RSLogix™ 5000 Programmer's Guide for Integrated Condition Monitoring Data Collection

Purpose and Scope

This guide is intended for the programmer or software engineer responsible for programming the Logix family of controllers (Programmable Automation Controllers - PACs) to get data from any supported I/O device. This guide contains an explanation of some of the settings needed to move data from a Logix controller into Emonitor. This guide uses the Allen-Bradley® XM-120 Dynamic Vibration Module as the example; however a PAC can accept condition monitoring data from other sources as well.

This guide does not cover using any of the software or other hardware components of an Emonitor system. For information on using an Emonitor program, refer to the online help for that program.

Introduction

The new Logix family of controllers provides a way to retrieve data from XM modules using programmable automation controllers (PACs). The PAC reads data from the XM module, and makes that data available to RSLinx Classic (which functions as an OPC server). The Emonitor software imports the data from RSLinx Classic into the Emonitor database, where it can be analyzed in Emonitor.

This guide is divided into several sections. The first section defines the UDTs required to get data from an XM module using a PAC. The remaining sections describe the handshaking routine to protect data integrity, the sample project file, and the details of reading data from XM modules.

Terms used in this guide

Emonitor Program: Refers to the software programs in the Emonitor family, such as those in the Emonitor Workstation, Emonitor Factory, and Emonitor Enterprise. Emonitor includes a database for storing condition monitoring data and powerful tools for alarming, reporting, and data analysis.

PAC: A programmable automation controller; specifically one of the Logix family of controllers.

UDT: User defined data type; a data structure that lets you create customized memory records that consolidate multiple fields (members) of data into a single contiguous group with a hierarchical lay-out.

ICM: Integrated Condition Monitoring, which includes capturing, analyzing and effectively using machine condition information to reduce maintenance costs and increase uptime and productivity.

Emonitor Online Data Management Console: An Emonitor software component that starts and stops data acquisition through the XM/DYN Online Data and Logix Online Data data sources.

Logix Online Data data source: An Emonitor software component that gets data from a Logix PAC.

Sample project file

There is a sample project file that shows how the information in this guide can be used in an RSLogix 5000 project. The project is located on the Emonitor version 3.50 software installation disk: **\extras\RSLogix Application\ICM DATA APP.ACD**

RSLogix 5000 programming manuals

For more information on the topics covered in this guide, refer to the RSLogix 5000 Programming manuals, found on the Rockwell Literature Library, http://www.rockwellautomation.com/literature/.

Publication Title	Publication Number
Logix5000 Controllers Add-on Instructions Programming Manual	1756-PM010C-EN-P
Logix5000 Controllers ASCII Strings Programming Manual	1756-PM013B-EN-P
Logix5000 Controllers Common Procedures Programming Manual	1756-PM001M-EN-E
Logix5000 Controllers Controller Information and Status Programming Manual	1756-PM015C-EN-P
Logix5000 Controllers Function Block Diagram Programming Manual	1756-PM009C-EN-P
Logix5000 Controllers I/O and Tag Data Programming Manual	1756-PM004C-EN-P
Logix5000 Controllers IEC 61131-3 Compliance Programming Manual	1756-PM018B-EN-P
Logix5000 Controllers Ladder Diagram Programming Manual	1756-PM008C-EN-P
Logix5000 Controllers Major, Minor, and I/O Faults Programming Manual	1756-PM014D-EN-P
Logix5000 Controllers Messages Programming Manual	1756-PM012C-EN-P
Logix5000 Controllers Nonvolatile Memory Programming Manual	1756-PM017D-EN-P
Logix5000 Controllers Produced and Consumed Tags Programming Manual	1756-PM011C-EN-P

Overview of ICM data types in RSLogix 5000

In the integrated condition monitoring world, each piece of data that is collected contains numerous elements. Some of the elements are time stamps, data units, signal detection, actual values and more. To ensure data integrity, each element of the data must be read before any updates in the controller.

ICM data types

Based on the data structure and the need for data integrity, it is necessary to create user defined data types to hold the different types of data. There are seven different data types that are used in integrated condition monitoring.

Data Type	Description	UDT
Values	This data is normal single value data that can be numeric, magnitude, or enumerated (Emonitor data types)	ICM_Data_Value
Vectors	This data is considered single value data that contains a magnitude and phase. It can only be of type magnitude (Emonitor data type).	ICM_Data_Vector
Spectrum – Normalized	This data is considered multi-value data and it contains the spectrum data in a normalized form. The normalized form indicates that there is a de-normalizing factor and a set of normalized integer values.	ICM_Data_Spectrum_Normalized
Spectrum – Actual	This data is considered multi-value data and it contains the spectrum data in a de-normalized form. Instead of a de-normalizing factor and a set of normalized integers, there is a set of real values.	ICM_Data_Spectrum_Actual
Time Waveforms – Normalized	This data is considered multi-value data and it contains the time waveform data in a normalized form. The normalized form indicates that there is a de-normalizing factor and a set of normalized integer values.	ICM_Data_TimeWaveform_Normalized
Time Waveforms - Actual	This data is considered multi-value data and it contains the time waveform data in a de-normalized form. Instead of a de-normalizing factor and a set of normalized integers, there is a set of real values.	ICM_Data_ TimeWaveform _Actual
Date and Time Stamp	This data consists of the date and time information in UTC time. It is used in each of the data types listed in this table as a nested UDT	ICM_WallClockType

The names of the user defined types must match the information in the table above. If you need to define multiple UDTs for each data type, a suffix can be attached to the names. Multiple UDTs would be required if you want different sizes of arrays for more efficient memory allocation.

Also there must be two user defined string types created:

Data Type	Description
String_16	This string is a 16 character string. In the ICM data UDT elements, this is used as the data type for the units. Emonitor can have unit names of up to 16 characters.
String_32	This string is a 32 character string. In the ICM data UDT elements, this is used as the data type for the Item Name. Emonitor can have item names of up to 32 characters.

ICM_Data_Value UDT

The ICM_Data_Value user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
NumberOfDataValues	DINT	Decimal	NUMBER OF VALUES PRESENT IN DATA ARRAY	Read/Write
ItemName	String_32[xx]		EMONITOR DATA TAG NAME FOR THIS DATA ITEM	Read/Write
Units	String_16[xx]		DATA VALUE UNITS	Read/Write
DataValueType	SINT[xx]	Decimal	0=NONE, 1=MAGNITUDE, 2=NUMERIC, 3=ENUMERATED.	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF DATA COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF DATA COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SignalDetection	SINT[xx]	Decimal	0=NONE, 1=RMS, 2=CALCULATED PK, 3=CALCULATED PK-PK, 4=TRUE PK, 5=TRUE PK-PK	Read/Write
DataValues	REAL[xx]	Float	DATA ITEM VALUE	Read/Write

A deviation to this UDT can be the following as indicated by the bold text in the Data Type column of the last row. This replaces the REAL array with DINT array.

DataValues	**DINT[xx]**	Float	DATA ITEM VALUE	Read/Write

In the table above the "[xx]" will be replaced with the actual array size. Array sizes in Logix are static, meaning that memory is allocated, whether it is actually used or not.

ICM_Data_Vector UDT

The ICM_Data_Vector user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
NumberOfDataValues	DINT	Decimal	NUMBER OF DATA VALUES PRESENT IN DATA ARRAY	Read/Write
ItemName	String_32[xx]		EMONITOR DATA TAG NAME FOR THIS DATA ITEM	Read/Write
MagnitudeUnits	String_16[xx]		DATA VALUE MAGNITUDE UNITS	Read/Write
PhaseUnits	BOOL[32]	Decimal	DATA VALUE PHASE UNITS (0=DEGREES, 1= RADIANS) FOR DATA ALIGN ON RSLOGIX5000, BOOL ARRAY SIZE MAY BE GREATER THAN NUMBEROFDATAVALUES. (IF NUMBEROFDATAVALUES IS 50, BOOL ARRAY SIZE MUST BE 64 ON RSLOGIX5000.)	Read/Write
DataValueType	SINT[xx]	Decimal	ONLY CAN BE 1=MAGNITUDE	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF DATA COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF DATA COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SignalDetection	SINT[xx]	Decimal	0=NONE, 1=RMS, 2=CALCULATED PK, 3=CALCULATED PK-PK, 4=TRUE PK, 5=TRUE PK-PK	Read/Write
MagnitudeValues	REAL[xx]	Float	MAGNITUDE DATA VALUES	Read/Write
PhaseValues	REAL[xx]	Float	PHASE DATA VALUES	Read/Write

In the table above the "[xx]" will be replaced with the actual array size. Array sizes in Logix are static, meaning that memory is allocated, whether it is actually used or not.

ICM_Data_Spectrum_Normalized

The ICM_Data_Spectrum_Normalized user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
ItemName	String_32		EMONITOR DATA TAG NAME FOR THIS SPECTRUM	Read/Write
AmplitudeUnits	String_16		SPECTRUM AMPLITUDE UNITS	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF SPECTRUM COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF SPECTRUM COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SpeedUnits	BOOL	Decimal	SPEED FEEDBACK UNITS (0=HZ, 1=CPM)	Read/Write
OrdersFlag	BOOL	Decimal	FREQUENCY UNITS AS HERTZ OR ORDERS (1=ORDERS, 0=HZ)	Read/Write
CPBFlag	BOOL	Decimal	CPB OR NONE (1=CPB, 0=NONE)	Read/Write
PowerFlag	BOOL	Decimal	NORMALIZED SPECTRUM DATA FORMAT (1=POWER, 0=NONE)	Read/Write
ComplexFlag	BOOL	Decimal	NORMALIZED SPECTRUM DATA (1=COMPLEX,0=REAL/POWER)	Read/Write
SignalDetection	SINT	Decimal	0=NONE, 1=RMS, 2=CALCULATED PK, 3=CALCULATED PK-PK,4=TRUE PK, 5=TRUE PK-PK	Read/Write
WindowType	DINT	Decimal	WINDOWTYPE: (1=RECTANGULAR, 2=HANNING, 3=FLAT TOP, 4=HAMMING, 5=KAISER-BESSEL, 6=COS).	Read/Write
WindowLineshapeFactor	REAL	Float	WINDOW LINE TYPE FACTOR	Read/Write
AverageTypeEnumeration	DINT	Decimal	AVERAGETYPEENUMERATION: (0 = USED IN XM ONLINE DATA, 1=LINEAR, 2=TIME SYNCHRONOUS, 3=EXPONENTIAL, 4=PEAK HOLD).	Read/Write
Speed	REAL	Float	SPEED FEEDBACK AT TIME OF SPECTRUM COLLECTION	Read/Write
NumberOfLines	DINT	Decimal	NUMBER OF SPECTRUM LINES	Read/Write
Fmax	REAL	Float	MAXIMUM FREQUENCY OR ORDERS OF SPECTRUM DATA.	Read/Write
Fmin	REAL	Float	MINIMUM FREQUENCY OR ORDERS OF SPECTRUM DATA	Read/Write
AmplitudeReference	REAL	Float	AMPLITUDE NORMALIZATION REFERENCE	Read/Write
NormalizingConversionFactor	DINT	Decimal	AMPLITUDE CONVERSION FACTOR	Read/Write
DataValues	INT[xxxx]	Decimal	NORMALIZED SPECTRUM DATA, IF COMPLEXFLAG IS 1, ARRAYSIZE = 2 * NUMBEROFLINES, IF COMPLEXFLAG IS 0, ARRAYSIZE = NUMBEROFLINES.	Read/Write

In the table above the "[xxxx]" will be replaced with the actual array size that will hold either the quantity of lines or two times the number of lines. Array sizes in Logix are static, meaning that memory is allocated, whether it is actually used or not.

ICM_Data_Spectrum_Actual

The ICM_Data_Spectrum_Actual user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
ItemName	String_32		EMONITOR DATA TAG NAME FOR THIS SPECTRUM	Read/Write
AmplitudeUnits	String_16		SPECTRUM AMPLITUDE UNITS	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF SPECTRUM COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF SPECTRUM COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SpeedUnits	BOOL	Decimal	SPEED FEEDBACK UNITS (0=HZ, 1=CPM)	Read/Write
OrdersFlag	BOOL	Decimal	FREQUENCY UNITS AS HERTZ OR ORDERS (1=ORDERS, 0=HZ)	Read/Write
CPBFlag	BOOL	Decimal	CPB OR NONE (1=CPB, 0=NONE)	Read/Write
PowerFlag	BOOL	Decimal	ACTUAL SPECTRUM DATA FORMAT (1=POWER, 0=NONE)	Read/Write
ComplexFlag	BOOL	Decimal	ACTUAL SPECTRUM DATA (1=COMPLEX,0=REAL/POWER)	Read/Write
SignalDetection	SINT	Decimal	0=NONE, 1=RMS, 2=CALCULATED PK, 3=CALCULATED PK-PK,4=TRUE PK, 5=TRUE PK-PK	Read/Write
WindowType	DINT	Decimal	WINDOWTYPE: (1=RECTANGULAR, 2=HANNING, 3=FLAT TOP, 4=HAMMING, 5=KAISER-BESSEL, 6=COS).	Read/Write
WindowLineshapeFactor	REAL	Float	WINDOW LINE TYPE FACTOR	Read/Write
AverageTypeEnumeration	DINT	Decimal	AVERAGETYPEENUMERATION: (0 = USED IN XM ONLINE DATA, 1=LINEAR, 2=TIME SYNCHRONOUS, 3=EXPONENTIAL, 4=PEAK HOLD).	Read/Write
Speed	REAL	Float	SPEED FEEDBACK AT TIME OF SPECTRUM COLLECTION	Read/Write
NumberOfLines	DINT	Decimal	NUMBER OF SPECTRUM LINES	Read/Write
Fmax	REAL	Float	MAXIMUM FREQUENCY OR ORDERS OF SPECTRUM DATA.	Read/Write
Fmin	REAL	Float	MINIMUM FREQUENCY OR ORDERS OF SPECTRUM DATA	Read/Write
DataValues	REAL[xxxx]	Decimal	ACTUAL SPECTRUM DATA, IF COMPLEXFLAG IS 1, ARRAYSIZE = 2 * NUMBEROFLINES, IF COMPLEXFLAG IS 0, ARRAYSIZE = NUMBEROFLINES.	Read/Write

In the table above the "[xxxx]" will be replaced with the actual array size that will hold either the quantity of lines or two times the number of lines. Array sizes in Logix are static, meaning that memory is allocated, whether it is actually used or not.

ICM_Data_TimeWaveform_Normalized

The ICM_Data_TimeWaveform_Normalized user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
ItemName	String_32		EMONITOR DATA TAG NAME FOR THIS TIME WAVEFORM	Read/Write
AmplitudeUnits	String_16		WAVEFORM AMPLITUDE UNITS	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF WAVEFORM COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF WAVEFORM COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SpeedUnits	BOOL	Decimal	SPEED FEEDBACK UNITS (0=HZ, 1=CPM)	Read/Write
PeriodFlag	BOOL	Decimal	0=SECONDS, 1=CYCLES	Read/Write
SamplingMode	BOOL	Decimal	0=ASYNCHRONOUS, 1=SYNCHRONOUS	Read/Write
AverageTypeEnumeration	DINT	Decimal	AVERAGETYPEENUMERATION: (0 = USED IN XM ONLINE DATA, 1=LINEAR, 2=TIME SYNCHRONOUS, 3=EXPONENTIAL, 4=PEAK HOLD).	Read/Write
Speed	REAL	Float	SPEED FEEDBACK AT TIME OF WAVEFORM COLLECTION	Read/Write
NumberOfPoints	DINT	Decimal	NUMBER OF TIME WAVEFORM DATA POINTS	Read/Write
Period	REAL	Float	TIME WAVEFORM PERIOD	Read/Write
AmplitudeReference	REAL	Float	AMPLITUDE NORMALIZATION REFERENCE	Read/Write
NormalizingConversionFactor	DINT	Decimal	AMPLITUDE CONVERSION FACTOR	Read/Write
DataValues	INT[xxxx]	Decimal	NORMALIZED TIME WAVEFORM DATA	Read/Write

In the table above the "[xxxx]" will be replaced with the actual array size that will hold either the quantity of points in the waveform. Array sizes in Logix are static, meaning that memory is allocated, whether it is actually used or not.

ICM_Data_TimeWaveform_Actual

The ICM_Data_TimeWaveform_Actual user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
ItemName	String_32		EMONITOR DATA TAG NAME FOR THIS TIME WAVEFORM	Read/Write
AmplitudeUnits	String_16		WAVEFORM AMPLITUDE UNITS	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF WAVEFORM COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF WAVEFORM COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SpeedUnits	BOOL	Decimal	SPEED FEEDBACK UNITS (0=HZ, 1=CPM)	Read/Write
PeriodFlag	BOOL	Decimal	(0=SECONDS 1=CYCLES)	Read/Write
SamplingMode	BOOL	Decimal	(0=ASYNCHRONOUS, 1=SYNCHRONOUS)	Read/Write
AverageTypeEnumeration	DINT	Decimal	AVERAGETYPEENUMERATION: (0 = USED IN XM ONLINE DATA, 1=LINEAR, 2=TIME SYNCHRONOUS, 3=EXPONENTIAL, 4=PEAK HOLD).	Read/Write
Speed	REAL	Float	SPEED FEEDBACK AT TIME OF WAVEFORM COLLECTION	Read/Write
NumberOfPoints	DINT	Decimal	NUMBER OF TIME WAVEFORM DATA POINTS	Read/Write
Period	REAL	Float	TIME WAVEFORM PERIOD	Read/Write
DataValues	REAL[xxxx]	Float	WAVEFORM DATA ACTUAL VALUES	Read/Write

In the table above the "[xxxx]" will be replaced with the actual array size that will hold either the quantity of points in the waveform. Array sizes in Logix are static, meaning that memory is allocated, whether it is actually used or not.

ICM_WallClockTime

The ICM_WallClockTime user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
Year	DINT	Decimal	YEAR (FOUR DIGIT NUMBER)	Read/Write
Month	DINT	Decimal	MONTH (1-12)	Read/Write
Day	DINT	Decimal	DAY (1-31)	Read/Write
Hour	DINT	Decimal	HOUR (0-23)	Read/Write
Minute	DINT	Decimal	MINUTE (0-59)	Read/Write
Second	DINT	Decimal	SECOND (0-59)	Read/Write
Microseconds	DINT	Decimal	MICROSECONDS (0-999,999)	Read/Write

Handshaking mechanism and ICM data values

ICM handshaking mechanism

Since the data contains multiple elements, there needs to be a mechanism in place to ensure that the data does not change as it is being read from the controller. This is the purpose of the handshaking elements (RTR and OTR) in the user defined types.

There is also a need to time out the hand shaking if there is a communication loss. By default the time out should be twenty (20) seconds. This is the default on the Emonitor side.

The normal handshake process between Emonitor and Logix for scheduled data collection is as follows:

1. Emonitor (or other application) sets the ready to read (RTR) flag.
2. The Logix application then verifies that data is not being writte, or finishes writing to the data points.
3. Logix sets the ok to read (OTR) flag.
4. Emonitor reads the data.
5. When Emonitor is done reading the data, it clears the ready to read (RTR) flag.
6. When Logix sees the ready to read (RTR) flag cleared, it clears the ok to read (OTR) flag.
7. Once the ok to read (OTR) flag is cleared, Logix can update data.

The normal handshake process between Emonitor and Logix for triggered data collection is as follows:

1. Logix sets the trigger.
2. When trigger is true, Emonitor (or other application) sets the ready to read (RTR) flag.
3. Logix application then verifies that data is not being writte, or finishes writing to the data points.
4. Logix sets the ok to read (OTR) flag.
5. Emonitor reads the data.
6. When Emonitor is done reading the data, it clears the ready to read (RTR) flag
7. When Logix sees the ready to read (RTR) flag cleared, it clears the ok to read (OTR) flag.
8. Once the ok to read (OTR) flag is cleared, Logix can reset the trigger if necessary and update the data.

Timeout functionality:

1. If the RTR flag is set true, and in twenty seconds, Emonitor does not see the OTR flag, the RTR and OTR flags are cleared.
2. If the RTR flag is set true, and in twenty seconds, Logix does not see the OTR flag, the RTR and OTR flags are cleared.

Each tag defined as an ICM data type must have a handshake set up for it. In the sample ACD file, there is an Add On Instruction (AOI) that is defined for this purpose. The add-on instruction functions as described above, but has an added mechanism to handle the asynchronous multi-tasking environment on the Logix side.

Hand Shaking AOI - ICM_Handshaking

The name of the instruction is ICM_Handshaking. The parameters for the instruction are defined as:

Name	Usage	Data Type	Description
EnableIn	Input	BOOL	Enable Input - System Defined Parameter
EnableOut	Output	BOOL	Enable Output - System Defined Parameter
BlockDataUpdates	Output	BOOL	FLAG TO BLOCK THE DATA UPDATES FOR THE DATA THAT IS TO BE READ
DataUpdatesBlocked	Input	BOOL	FLAG THAT THE DATA UPDATE HAS BEEN BLOCKED.
TimedOut	Output	BOOL	FLAG TO INDICATE THAT THE HANDSHAKE TIMED OUT.
TimeoutTime	Input	DINT	TIME OUT PRESET (DEFAULTS TO 20 SECONDS (20000 ms) IF NOT CHANGED)
ReadyToRead	InOut	BOOL	READY TO READ FLAG FROM DEVICE THAT IS TO READ THE DATA
OkToRead	InOut	BOOL	OK TO READ DATA FROM THIS CONTROLLER SENT TO THE DEVICE THAT IS TO READ THE DATA

When using the instruction, there will be a backing tag created for it. This backing tag will be created as data type “ICM_Handshaking” which is the same name as the instruction.

The elements of the backing tag will consist of the parameters that are either input or output but not the InOut type.

Name	Usage	Data Type	Description
EnableIn	Input	BOOL	Enable Input - System Defined Parameter
EnableOut	Output	BOOL	Enable Output - System Defined Parameter
BlockDataUpdates	Output	BOOL	FLAG TO BLOCK THE DATA UPDATES FOR THE DATA THAT IS TO BE READ
DataUpdatesBlocked	Input	BOOL	FLAG THAT THE DATA UPDATE HAS BEEN BLOCKED.
TimedOut	Output	BOOL	FLAG TO INDICATE THAT THE HANDSHAKE TIMED OUT.
TimeoutTime	Input	DINT	TIME OUT PRESET (DEFAULTS TO 20 SECONDS (20000 ms) IF NOT CHANGED)

The purpose of the handshaking instruction is to isolate the code and functionality so the programmer does not have to repeat all of it for each tag.

A structured text call to the instruction looks like the following:

ICM_Handshaking(handShaking[index1], icmNode_01_XM120_Spectrum[index1].RTR, icmNode_01_XM120_Spectrum[index1].OTR);

Where

- handshaking[index1] is an indexed array element of the type ICM_Handshaking
- icmNode_01_XM120_Spectrum[index1] is an indexed array element of the type ICM_Data_Sprectrum_Normalized.

When the instruction is executed, the following happens:

- If the ready to read parameter transitions from true to false,
    - The ok to read parameter is cleared
    - BlockDataUpdates parameter is cleared
    - The DataUpdatesBlocked parameter is cleared
- If the ready to read parameter is false (no transition)
    - Nothing happens
- If the ready to read parameter is true
    - Enable the time out timer
    - If the time out timer is not done timing
        - If the BlockDataUpdates parameter is false then set the BlockDataUpdates parameter
        - If the BlockDataUpdates parameter is true and the DataUpdatesBlocked parameter is true, set the ok to read parameter.
    - If the time out timer is done timing
        - Clear the ready to read parameter
        - Clear the ok to read parameter
        - Clear the BlockDataUpdates parameter
        - Clear the DataUpdatesBlocked parameter
        - Set the time out parameter
        - Reset the time out timer.

The BlockDataUpdates parameter allows for the programmer to place the handshake call in any task and handle the asynchronous behavior of multi-tasking environment. So when the BlockDataUpdates parameter is set, the user routine finishes updating the data and/or makes sure that the data is not updated again, and then sets the DataUpdatesBlocked parameter. See the sample Logix project file for details.

ICM data values

Data in the ICM_Data_Values

The ICM_Data_Value user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
NumberOfDataValues	DINT	Decimal	NUMBER OF VALUES PRESENT IN DATA ARRAY	Read/Write
ItemName	String_32[xx]		EMONITOR DATA TAG NAME FOR THIS DATA ITEM	Read/Write
Units	String_16[xx]		DATA VALUE UNITS	Read/Write
DataValueType	SINT[xx]	Decimal	0=NONE, 1=MAGNITUDE, 2=NUMERIC, 3=ENUMERATED.	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF DATA COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF DATA COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SignalDetection	SINT[xx]	Decimal	0=NONE, 1=RMS, 2=CALCULATED PK, 3=CALCULATED PK-PK, 4=TRUE PK, 5=TRUE PK-PK	Read/Write
DataValues	REAL[xx]	Float	DATA ITEM VALUE	Read/Write

The ICM_Data_Value user defined type consists of the following elements:

- The value of the xx in the elements that are arrays must be greater than or equal to the number of data values to use.
- When a tag of type ICM_Data_Value is to be used to collect data, certain elements must be initialized, and some are optional.

NumberOfDataValues

The initialization of the NumberOfDataValues is required. When attached to an Emonitor Online System, the number of data values to use controls what the engineer can map. If not initialized, defaults to zero in the controller. Emonitor Online System does not display items associated with this tag unless it contains a value greater than zero.

ItemName

The initialization of the ItemName array elements is optional. When used, the item name array elements can help identify the data. When attached to an Emonitor online system and item name array elements contain values, Emonitor uses the item name to create measurement definitions during auto-mapping.

Units

The initialization of the Units array elements is required. When attached to an Emonitor Online System, Emonitor uses the units in the mapping rules.

DataValueType

The initialization of the DataValueType array elements is required. When attached to an Emonitor Online System, the data value types are used in the mapping rules. During runtime, the data value type is used to determine the value to store in the database based on signal detection settings.

Controller_UTC_DateTime

The initialization of the Controller_UTC_DateTime elements is not required. During runtime the elements of this field must be updated. This structure contains the time (UTC) that the data was collected on the controller side. Use the GSV command for WallClockTime to get the elements from the controller.

Device_UTC_DateTime

The initialization of the Device_UTC_DateTime elements is not required. During runtime the elements of this field are updated, depending on the device that is collecting the data. This structure contains the time (UTC) that the data was collected on the device side.

RTR (Ready to Read)

The initialization of the RTR (Ready to Read) element is not required. When attached to an Emonitor Online System, the ready to read element is written to by the Logix Online Data driver. It is only written to by the controller when there is a time-out in the handshaking mechanism.

OTR (Ok to Read)

The initialization of the OTR (Ok to Read) element is not required. When attached to an Emonitor Online System, the ok to read element is written to by the controller during the handshaking to ensure data is being written to while it is being read.

SignalDetection

The initialization of the SignalDetection array elements is required. When attached to an Emonitor Online System, the data is stored in the database as RMS. Signal Detection setting and data type will determine how the value is transformed into an RMS value. Signal Detection is used for data type magnitude only, being ignored when the data type is numeric or enumerated.

DataValues

The initialization of the DataValues element is not required. During runtime, the data collected from the device must be placed into this array.

Data in the ICM_Data_Vectors

The ICM_Data_Vector user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
NumberOfDataValues	DINT	Decimal	NUMBER OF DATA VALUES PRESENT IN DATA ARRAY	Read/Write
ItemName	String_32[xx]		EMONITOR DATA TAG NAME FOR THIS DATA ITEM	Read/Write
MagnitudeUnits	String_16[xx]		DATA VALUE MAGNITUDE UNITS	Read/Write
PhaseUnits	BOOL[32]	Decimal	DATA VALUE PHASE UNITS (0=DEGREES, 1= RADIANS) FOR DATA ALIGN ON RSLOGIX5000, BOOL ARRAY SIZE MAY BE GREATER THAN NUMBEROFDATAVALUES. (IF NUMBEROFDATAVALUES IS 50, BOOL ARRAY SIZE MUST BE 64 ON RSLOGIX5000.)	Read/Write
DataValueType	SINT[xx]	Decimal	ONLY CAN BE 1=MAGNITUDE	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF DATA COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF DATA COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SignalDetection	SINT[xx]	Decimal	0=NONE, 1=RMS, 2=CALCULATED PK, 3=CALCULATED PK-PK, 4=TRUE PK, 5=TRUE PK-PK	Read/Write
MagnitudeValues	REAL[xx]	Float	MAGNITUDE DATA VALUES	Read/Write
PhaseValues	REAL[xx]	Float	PHASE DATA VALUES	Read/Write

- The value of the xx in the elements that are arrays must be greater than or equal to the number of data values to use.
- When a tag of type ICM_Data_Vector is to be used to collect data, certain elements must be initialized, and some are optional.

NumberOfDataValues

The initialization of the NumberOfDataValues is required. When attached to an Emonitor Online System, the number of data values to use controls what the engineer can map. If not initialized, defaults to zero in the controller. Emonitor Online System does not display items associated with this tag unless it contains a value greater than zero.

ItemName

The initialization of the ItemName array elements is optional. When used, the item name array elements can help identify the data. When attached to an Emonitor online system and item name array elements contain values, Emonitor uses the item name to create measurement definitions during auto-mapping.

MagnitudeUnits

The initialization of the MagnitudeUnits array elements is required. When attached to an Emonitor Online System, the units are used in the mapping rules.

PhaseUnits

The initialization of the PhaseUnits array elements is required. When attached to an Emonitor Online System, the phase data is stored in the Emonitor database in units of radians. Devices may give the phase in degrees or radians.

DataValueType

The initialization of the DataValueType array elements is required. For the vector data, the data value type must be magnitude.

SignalDetection

The initialization of the SignalDetection array elements is required. When attached to an Emonitor Online System, the data is stored in the database as RMS. Signal Detection setting will determine how the value is transformed into an RMS value.

Controller_UTC_DateTime

The initialization of the Controller_UTC_DateTime elements is not required. During runtime the elements of this field must be updated. This structure contains the time (UTC) that the data was collected on the controller side.

Use the GSV command for WallClockTime to get the elements from the controller.

Device_UTC_DateTime

The initialization of the Device_UTC_DateTime elements is not required. During runtime the elements of this field will be updated, depending on the device that is collecting the data. This structure contains the time (UTC) that the data was collected on the device side.

RTR (Ready to Read)

The initialization of the RTR (Ready to Read) element is not required. When attached to an Emonitor Online System, the ready to read element is written to by the Logix Online Data driver. It is only written to by the controller when there is a time out in the handshaking mechanism.

OTR (Ok to Read)

The initialization of the OTR (Ok to Read) element is not required. When attached to an Emonitor Online System, the ok to read element is written to by the controller during the handshaking to insure data is being written to while it is being read.

MagnitudeValues

The initialization of the MagnitudeValues element is not required. During runtime, the magnitude data collected from the device must be placed into this array.

PhaseValues

The initialization of the PhaseValues element is not required. During runtime, the phase data collected from the device must be placed into this array.

ICM_Data_Spectrum_Normalized and ICM_Data_Spectrum_Actual

The ICM_Data_Spectrum_Normalized user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
ItemName	String_32		EMONITOR DATA TAG NAME FOR THIS SPECTRUM	Read/Write
AmplitudeUnits	String_16		SPECTRUM AMPLITUDE UNITS	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF SPECTRUM COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF SPECTRUM COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SpeedUnits	BOOL	Decimal	SPEED FEEDBACK UNITS (0=HZ, 1=CPM)	Read/Write
OrdersFlag	BOOL	Decimal	FREQUENCY UNITS AS HERTZ OR ORDERS (1=ORDERS, 0=HZ)	Read/Write
CPBFlag	BOOL	Decimal	CPB OR NONE (1=CPB, 0=NONE)	Read/Write
PowerFlag	BOOL	Decimal	NORMALIZED SPECTRUM DATA FORMAT (1=POWER, 0=NONE)	Read/Write
ComplexFlag	BOOL	Decimal	NORMALIZED SPECTRUM DATA (1=COMPLEX,0=REAL/POWER)	Read/Write
SignalDetection	SINT	Decimal	0=NONE, 1=RMS, 2=CALCULATED PK, 3=CALCULATED PK-PK,4=TRUE PK, 5=TRUE PK-PK	Read/Write
WindowType	DINT	Decimal	WINDOWTYPE: (1=RECTANGULAR, 2=HANNING, 3=FLAT TOP, 4=HAMMING, 5=KAISER-BESSEL, 6=COS).	Read/Write
WindowLineshapeFactor	REAL	Float	WINDOW LINE TYPE FACTOR	Read/Write
AverageTypeEnumeration	DINT	Decimal	AVERAGETYPEENUMERATION: (0 = USED IN XM ONLINE DATA, 1=LINEAR, 2=TIME SYNCHRONOUS, 3=EXPONENTIAL, 4=PEAK HOLD).	Read/Write
Speed	REAL	Float	SPEED FEEDBACK AT TIME OF SPECTRUM COLLECTION	Read/Write
NumberOfLines	DINT	Decimal	NUMBER OF SPECTRUM LINES	Read/Write
Fmax	REAL	Float	MAXIMUM FREQUENCY OR ORDERS OF SPECTRUM DATA.	Read/Write
Fmin	REAL	Float	MINIMUM FREQUENCY OR ORDERS OF SPECTRUM DATA	Read/Write
AmplitudeReference	REAL	Float	AMPLITUDE NORMALIZATION REFERENCE	Read/Write
NormalizingConversionFactor	DINT	Decimal	AMPLITUDE CONVERSION FACTOR	Read/Write
DataValues	INT[xxxx]	Decimal	NORMALIZED SPECTRUM DATA, IF COMPLEXFLAG IS 1, ARRAYSIZE = 2 * NUMBEROFLINES, IF COMPLEXFLAG IS 0, ARRAYSIZE = NUMBEROFLINES.	Read/Write

The ICM_Data_Spectrum_Actual user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
ItemName	String_32		EMONITOR DATA TAG NAME FOR THIS SPECTRUM	Read/Write
AmplitudeUnits	String_16		SPECTRUM AMPLITUDE UNITS	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF SPECTRUM COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF SPECTRUM COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write

SpeedUnits	BOOL	Decimal	SPEED FEEDBACK UNITS (0=HZ, 1=CPM)	Read/Write
OrdersFlag	BOOL	Decimal	FREQUENCY UNITS AS HERTZ OR ORDERS (1=ORDERS, 0=HZ)	Read/Write
CPBFlag	BOOL	Decimal	CPB OR NONE (1=CPB, 0=NONE)	Read/Write
PowerFlag	BOOL	Decimal	ACTUAL SPECTRUM DATA FORMAT (1=POWER, 0=NONE)	Read/Write
ComplexFlag	BOOL	Decimal	ACTUAL SPECTRUM DATA (1=COMPLEX,0=REAL/POWER)	Read/Write
SignalDetection	SINT	Decimal	0=NONE, 1=RMS, 2=CALCULATED PK, 3=CALCULATED PK-PK,4=TRUE PK, 5=TRUE PK-PK	Read/Write
WindowType	DINT	Decimal	WINDOWTYPE: (1=RECTANGULAR, 2=HANNING, 3=FLAT TOP, 4=HAMMING, 5=KAISER-BESSEL, 6=COS).	Read/Write
WindowLineshapeFactor	REAL	Float	WINDOW LINE TYPE FACTOR	Read/Write
AverageTypeEnumeration	DINT	Decimal	AVERAGETYPEENUMERATION: (0 = USED IN XM ONLINE DATA, 1=LINEAR, 2=TIME SYNCHRONOUS, 3=EXPONENTIAL, 4=PEAK HOLD).	Read/Write
Speed	REAL	Float	SPEED FEEDBACK AT TIME OF SPECTRUM COLLECTION	Read/Write
NumberOfLines	DINT	Decimal	NUMBER OF SPECTRUM LINES	Read/Write
Fmax	REAL	Float	MAXIMUM FREQUENCY OR ORDERS OF SPECTRUM DATA.	Read/Write
Fmin	REAL	Float	MINIMUM FREQUENCY OR ORDERS OF SPECTRUM DATA	Read/Write
DataValues	REAL[xxxx]	Decimal	ACTUAL SPECTRUM DATA, IF COMPLEXFLAG IS 1, ARRAYSIZE = 2 * NUMBEROFLINES, IF COMPLEXFLAG IS 0, ARRAYSIZE = NUMBEROFLINES.	Read/Write

In the table above the “[xxxx]” will be replaced with the actual array size that will hold either the quantity of lines or two times the number of lines. Array sizes in Logix are static, meaning that memory is allocated, whether it is actually used or not.

ItemName

The initialization of the ItemName element is optional. When used, the item name element can help identify the data. When attached to an Emonitor online system and item name element contains a value, Emonitor uses the item name to create the measurement definition during auto-mapping.

AmplitudeUnits

The initialization of the AmplitudeUnits element is required. When attached to an Emonitor Online System, the amplitude units are used in the mapping rules.

Controller_UTC_DateTime

The initialization of the Controller_UTC_DateTime elements is not required. During runtime the elements of this field must be updated. This structure contains the time (UTC) that the data was collected on the controller side. Use the GSV command for WallClockTime to get the elements from the controller.

Device_UTC_DateTime

The initialization of the Device_UTC_DateTime elements is not required. During runtime the elements of this field will be updated, depending on the device that is collecting the data. This structure contains the time (UTC) that the data was collected on the device side.

RTR (Ready to Read)

The initialization of the RTR (Ready to Read) element is not required. When attached to an Emonitor Online System, the ready to read element is written to by the Logix Online Data driver. It is only written to by the controller when there is a time out in the handshaking mechanism.

OTR (Ok to Read)

The initialization of the OTR (Ok to Read) element is not required. When attached to an Emonitor Online System, the ok to read element is written to by the controller during the handshaking to insure data is being written to while it is being read.

SpeedUnits

The initialization of the SpeedUnits element is required. When attached to an Emonitor Online System, the speed associated at the time the spectrum was collected will be stored in the database in units of Hz. When collected from the device the speed may be in units of Cycles per minute or RPM. This field is used to indicate how the speed is to be transformed for storage.

OrdersFlag

The initialization of the OrderFlag element is required. Spectrums can be collected either asynchronously or synchronously to the tachometer signal (equipment speed). When the orders flag is cleared, the spectrum frequency is in units of Hz. When set the spectrum frequency is in orders of equipment speed.

CPBFlag (constant percentage bandwidth)

The initialization of the OrderFlag element is required. If the device is using CPB to collect the data, then this flag must be set (set to 1). If the device is not using CPB, then this flag should be cleared (set to 0).

ComplexFlag (spectrum is composed of complex data)

The initialization of the ComplexFlag element is required. If the spectrum collected from the device is a complex spectrum (real, imaginary) then this flag is set (set to one). When the spectrum collected from the device is not complex but is either a power or real spectrum, this flag is cleared (set to zero).

PowerFlag

The initialization of the ComplexFlag element is required. If the spectrum collected from the device is a complex spectrum or if the spectrum is not complex but is real spectrum, this flag is cleared (set to zero). If the spectrum is not complex but is a power spectrum, this flag is set (set to one).

SignalDetection

The initialization of the SignalDetection element is required. When attached to an Emonitor Online System, the data is stored in the database as RMS. Signal Detection setting will determine how the value is transformed into an RMS value.

WindowType

The initialization of the WindowType element is required. When the device calculates the spectrum (Fast Fourier Transform) there is a windowing mechanism used. This element is used to indicate what that windowing mechanism was.

WindowLineShape Factor

The initialization of the WindowLineShape Factor element is required. The window line shape affects the amplitude resolution of spectral lines. This value must be greater than zero (0).

AverageType Enumeration

The initialization of the AverageType Enumeration element is required. This value is based on the average type in the device that is collecting the data.

Speed

The initialization of the Speed element is optional. During runtime, this is the speed value at the time the spectrum was collected at the device.

NumberOfLines

The initialization of the NumberOfLines element is optional. During runtime, this value is the number of lines in the collected spectrum. If the spectrum is complex then the amount of data collected is two times the number of lines, else the amount of data is the number of lines. This value must be greater than 0.

Fmax

The initialization of the Fmax element is optional. During runtime, this value is the maximum frequency in the collected spectrum. This value must be set to a value greater than zero (0). It is used to determine the bandwidth (bin width) frequency.

bandwidth = (Fmax-Fmin) / (numberOfLines)

Fmin

The initialization of the Fmax element is optional. During runtime, this value is the maximum frequency in the collected spectrum. It is used to determine the bandwidth (bin width) frequency.

bandwidth = (Fmax-Fmin) / (numberOfLines)

AmplitudeReference (Normalized Spectrum Only)

The initialization of the AmplitudeReference element is required. During runtime, this value is needed to de-normalize the data values. This value is based on the device that data is collected with (if the spectrum is normalized at the device).

$Float_n$ = AmplitudeReference * ($dataValue_n$ / NormalizingConversionFactor)

NormalizingConversionFactor (Normalized Spectrum Only)

The initialization of the NormalizingConversionFactor element is required. During runtime, this value is needed to de-normalize the data values. This value is based on the device that data is collected with (if the spectrum is normalized at the device).

$Float_n$ = AmplitudeReference * ($dataValue_n$ / NormalizingConversionFactor)

DataValues

The initialization of the DataValues elements is optional. At runtime, the data values array elements will contain the amplitude values (normalized or actual based).

ICM_Data_TimeWaveform_Normalized

The ICM_Data_TimeWaveform_Normalized user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
ItemName	String_32		EMONITOR DATA TAG NAME FOR THIS TIME WAVEFORM	Read/Write
AmplitudeUnits	String_16		WAVEFORM AMPLITUDE UNITS	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF WAVEFORM COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF WAVEFORM COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SpeedUnits	BOOL	Decimal	SPEED FEEDBACK UNITS (0=HZ, 1=CPM)	Read/Write
PeriodFlag	BOOL	Decimal	0=SECONDS, 1=CYCLES	Read/Write
SamplingMode	BOOL	Decimal	0=ASYNCHRONOUS, 1=SYNCHRONOUS	Read/Write
AverageTypeEnumeration	DINT	Decimal	AVERAGETYPEENUMERATION: (0 = USED IN XM ONLINE DATA, 1=LINEAR, 2=TIME SYNCHRONOUS, 3=EXPONENTIAL, 4=PEAK HOLD).	Read/Write
Speed	REAL	Float	SPEED FEEDBACK AT TIME OF WAVEFORM COLLECTION	Read/Write
NumberOfPoints	DINT	Decimal	NUMBER OF TIME WAVEFORM DATA POINTS	Read/Write
Period	REAL	Float	TIME WAVEFORM PERIOD	Read/Write
AmplitudeReference	REAL	Float	AMPLITUDE NORMALIZATION REFERENCE	Read/Write
NormalizingConversionFactor	DINT	Decimal	AMPLITUDE CONVERSION FACTOR	Read/Write
DataValues	INT[xxxx]	Decimal	NORMALIZED TIME WAVEFORM DATA	Read/Write

The ICM_Data_TimeWaveform_Actual user defined type consists of the following elements:

Name	Data Type	Style	Description	External Access
ItemName	String_32		EMONITOR DATA TAG NAME FOR THIS TIME WAVEFORM	Read/Write
AmplitudeUnits	String_16		WAVEFORM AMPLITUDE UNITS	Read/Write
Controller_UTC_DateTime	ICM_WallClockTime		CONTROLLER'S DATE/TIME IN UTC AT THE TIME OF WAVEFORM COLLECTION	Read/Write
Device_UTC_DateTime	ICM_WallClockTime		DEVICE'S DATE/TIME IN UTC AT THE TIME OF WAVEFORM COLLECTION	Read/Write
RTR	BOOL	Decimal	REQUEST TO READ HANDSHAKE BIT	Read/Write
OTR	BOOL	Decimal	OK TO READ HANDSHAKE BIT	Read/Write
SpeedUnits	BOOL	Decimal	SPEED FEEDBACK UNITS (0=HZ, 1=CPM)	Read/Write
PeriodFlag	BOOL	Decimal	(0=SECONDS 1=CYCLES)	Read/Write
SamplingMode	BOOL	Decimal	(0=ASYNCHRONOUS, 1=SYNCHRONOUS)	Read/Write
AverageTypeEnumeration	DINT	Decimal	AVERAGETYPEENUMERATION: (0 = USED IN XM ONLINE DATA, 1=LINEAR, 2=TIME SYNCHRONOUS, 3=EXPONENTIAL, 4=PEAK HOLD).	Read/Write
Speed	REAL	Float	SPEED FEEDBACK AT TIME OF WAVEFORM COLLECTION	Read/Write
NumberOfPoints	DINT	Decimal	NUMBER OF TIME WAVEFORM DATA POINTS	Read/Write
Period	REAL	Float	TIME WAVEFORM PERIOD	Read/Write
DataValues	REAL[xxxx]	Float	WAVEFORM DATA ACTUAL VALUES	Read/Write

In the table above the “[xxxx]” will be replaced with the actual array size that will hold either the quantity of points in the waveform. Array sizes in Logix are static. Array sizes in Logix are static, meaning that memory is allocated, whether it is actually used or not.

ItemName

The initialization of the ItemName element is optional. When used, the item name element can help identify what the actual data is. When attached to an Emonitor online system and item name element contains a value, the item name will be used to create the measurement definition during auto-mapping.

AmplitudeUnits

The initialization of the AmplitudeUnits element is required. When attached to an Emonitor Online System, the amplitude units are used in the mapping rules.

Controller_UTC_DateTime

The initialization of the Controller_UTC_DateTime elements is not required. During runtime the elements of this field must be updated. This structure contains the time (UTC) that the data was collected on the controller side.

Use the GSV command for WallClockTime to get the elements from the controller.

Device_UTC_DateTime

The initialization of the Device_UTC_DateTime elements is not required. During runtime the elements of this field will be updated, depending on the device that is collecting the data. This structure contains the time (UTC) that the data was collected on the device side.

RTR (Ready to Read)

The initialization of the RTR (Ready to Read) element is not required. When attached to an Emonitor Online System, the ready to read element is written to by the Logix Online Data driver. It is only written to by the controller when there is a time out in the handshaking mechanism.

OTR (Ok to Read)

The initialization of the OTR (Ok to Read) element is not required. When attached to an Emonitor Online System, the ok to read element is written to by the controller during the handshaking to insure data is being written to while it is being read.

SpeedUnits

The initialization of the SpeedUnits element is required. When attached to an Emonitor Online System, the speed associated at the time the time waveform was collected will be stored in the database in units of Hz. When collected from the device the speed may be in units of Cycles per minute or RPM. This field is used to indicate how the speed is to be transformed for storage.

PeriodFlag

The initialization of the PeriodFlag element is required. Time waveforms can be collected either asynchronously or synchronously to the tachometer signal (equipment speed). When the period flag is cleared, the period is in seconds (asynchronous to tachometer signal). When the period flag is set, the period is in cycles (synchronous to tachometer signal).

SamplingMode

The initialization of the SamplingMode element is required. Time waveforms can be collected either asynchronously or synchronously to the tachometer signal (equipment speed). When the sampling mode flag is cleared, the time waveform is being collected asynchronous to tachometer signal. When the sampling mode flag is set, the time waveform is being collected synchronous to tachometer signal.

AverageType Enumeration

The initialization of the AverageType Enumeration element is required. This value is based on the average type in the device that is collecting the data.

Speed

The initialization of the Speed element is optional. During runtime, this is the speed value at the time the spectrum was collected at the device.

NumberOfPoints

The initialization of the NumberOfPoints element is optional. During runtime, this value is the number of points in the collected time waveform. This value must be greater than 0.

Period

The initialization of the Period element is optional. During runtime, this value is the overall time of the time waveform sample. This value must be greater than zero (0). This value is used to calculate the time between samples using the following formula.

Time Between Samples = (Period) / (numberOfPoints)

AmplitudeReference (NormalizedTime Waveform Only)

The initialization of the AmplitudeReference element is required. During runtime, this value is needed to de-normalize the data values. This value is based on the device that data is collected with (if the time waveform is normalized at the device).

$Float_n$ = AmplitudeReference * ($dataValue_n$ / NormalizingConversionFactor)

NormalizingConversionFactor (NormalizedTime Waveform Only)

The initialization of the NormalizingConversionFactor element is required. During runtime, this value is needed to de-normalize the data values. This value is based on the device that data is collected with (if the time waveform is normalized at the device).

$Float_n$ = AmplitudeReference * ($dataValue_n$ / NormalizingConversionFactor)

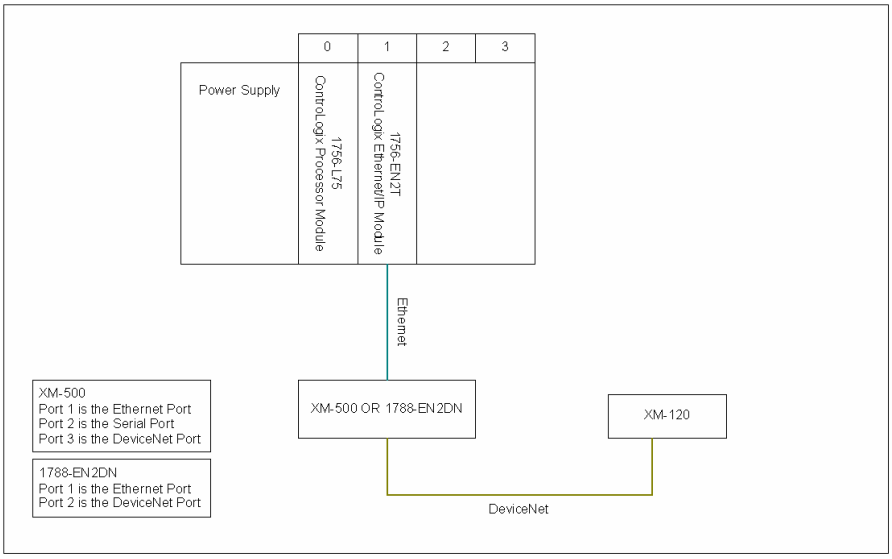
DataValues

The initialization of the DataValues elements is optional. At runtime, the data values array elements will contain the amplitude values (normalized or actual based).

SAMPLE PROJECT FILE

The sample Logix project file is used to read data from an XM-120 module. It then loads the data into the tags defined as the ICM user defined types. The project is named **ICM DATA APP.ACD** and located on the Emonitor software installation disk in **\extras\RSLogix Application** folder.

The project uses the following topology:



It uses an XM-500 module for the converter from Ethernet to DeviceNet. Even though the XM-500 can be a scanner module on DeviceNet it is not used as a scanner.

All data read from the XM-120 is done through explicit messaging.

The path used for the explicit messaging is :

Processor module (slot 0) to backplane

Backplane to Ethernet module (Slot 1)

Ethernet Module to Ethernet Port

Ethernet Port to XM500 Ethernet IP Address

XM-500 Ethernet Port to XM-500 DeviceNet Port (3)

XM-500 DeviceNet Port (3) to XM-120 (Node 1)

So the communication path will look like this

1, 1, 2, {ip address of XM-500}, 3, {Node Address of XM-120}

Setting the number of unconnected buffers

Since all data from the XM modules in this sample will be obtained through explicit messaging, the number of unconnected buffers needs to be increased. The default number of unconnected buffers is ten (10) but can be increased to forty (40). For more details see publication: ***Logix5000 Controllers Messages Programming Manual (1756-PM012C-EN-P)***.

This is accomplished in the task taskInitialization_01.

During execution, the current configured number of unconnected buffers is read. If the number of unconnected buffers is not set to the maximum of forty (40), an attempt is made to change the number of buffers. After the attempt is made then the configured number of buffers is read back. At this point, the configured number should be forty (40). If it is, then all is well and a flag is set accordingly to allow the XM module read commands. If it is not set to forty, then the flag is not set and no XM read commands can be executed.

Reading the assembly data, spectrum data, and time waveform data from the XM-120 module

The task that contains all the programs that read the XM-120 data is “task_XM120_DataCollection1”. This task contains three (3) programs that handle reading different information from the XM-120.

Task	Program	Routines
task_XM120_DataCollection1	program_Read_XM120_Assemblies	routine_Main1
		routine_DecodeMove
		routine_Initialize
	program_Read_XM120_Spectrum	routine_Main1
		routine_DecodeMove
		routine_Initialize
	program_Read_XM120_TimeWaveform	routine_Main1
		routine_DecodeMove
		routine_Initialize

The “program_Read_XM120_Assemblies” is the program that is used to read the assembly data. The “routine_Main1” routine is the main routine. It contains the state machine (7-states) that is used to control the reading of the assembly data. The “routine_Initialize” routine is called from the main routine to initialize data. The “routine_DecodeMove” routine is used to handle state errors and process data associated with many of the states.

State	Description
State 0	This state is used to reset all the buffer data
State 1	This state is used to read the assembly 100 data from the XM-120 module
State 2	This state is used to process the assembly 100 data that was read in state 0

State 3	This state is used to read the assembly 101 data from the XM-120 module
State 4	This state is used to process the assembly 101 data that was read in state 3
State 5	Update the ICM tags for assembly 100 if not being read from Emonitor
State 6	Update the ICM tags for assembly 101 if not being read from Emonitor

The “program_Read_XM120_Spectrum” is the program that is used to read the spectrum data. The “routine_Main1” routine is the main routine. It contains the state machine (10-states) that is used to control the reading of the spectrum. The “routine_Initialize” routine is called from the main routine to initialize data. The “routine_DecodeMove” routine is used to handle state errors and process data associated with many of the states.

To read the spectrum from both channels of the XM module, the state logic has to be executed two times. This is accomplished with first reading from channel 1. When channel 1 is read in then switch the instance numbers in the message command to read channel 2.

State	Description
State 0	This state is used to reset all the buffer data
State 1	This state is used to read the first 120 words from the XM-120 spectrum/waveform buffer and to read in the current speed from the tachometer. This read contains the data for a 100 line spectrum.
State 2	This state is used to process the data read in state 1
State 3	This state is used to read the next 120 words from the XM-120 spectrum/waveform. This read combined with the read from state 1 contains the data for a 200 line spectrum.
State 4	This state is used to process the data read in state 3
State 5	This state is used to read the next 180 words from the XM-120 spectrum/waveform. This read combined with the read from states 1 and 3 contains the data for a 400 line spectrum.
State 6	This state is used to process the data read in state 5
State 7	This state is used to read the next 420 words from the XM-120 spectrum/waveform. This read combined with the read from states 1, 3, and 5 contains the data for a800 line spectrum.
State 8	This state is used to process the data read in state 7
State 9	This state will perform 2 actions. Action 1 modifies the index pointer for the channel index and changes the instance number in the message command. Action 2 updates the ICM tags for both channel 1 and channel 2 spectrums if not being read from Emonitor

The “program_Read_XM120_TimeWaveform” is the program that is used to read the time waveform data. The “routine_Main1” routine is the main routine. It contains the state machine (10-states) that is used to control the reading of the time waveform. The “routine_Initialize” routine is called from the main routine to initialize data. The “routine_DecodeMove” routine is used to handle state errors and process data associated with many of the states.

State	Description
State 0	This state is used to reset all the buffer data
State 1	This state is used to read the assembly 100 data from the XM-120 module
State 2	This state is used to process the assembly 100 data that was read in state 0
State 3	This state is used to read the assembly 101 data from the XM-120 module
State 4	This state is used to process the assembly 101 data that was read in state 3
State 5	Update the ICM tags for assembly 100 if not being read from Emonitor
State 6	Update the ICM tags for assembly 101 if not being read from Emonitor

The “program_Read_XM120_Spectrum” is the program that is used to read the spectrum data. The “routine_Main1” routine is the main routine. It contains the state machine (10-states) that is used to control the reading of the spectrum. The “routine_Initialize” routine is called from the main routine to initialize data. The “routine_DecodeMove” routine is used to handle state errors and process data associated with many of the states.

To read the spectrum from both channels of the XM module, the state logic has to be executed two times. This is accomplished with first reading from channel 1. When channel 1 is read in then switch the instance numbers in the message command to read channel 2.

State	Description
State 0	This state is used to reset all the buffer data
State 1	This state is used to read the first 120 words from the XM-120 spectrum/waveform buffer and to read in the current speed from the tachometer. This read contains the data for a 100 line spectrum.
State 2	This state is used to process the data read in state 1
State 3	This state is used to read the next 120 words from the XM-120 spectrum/waveform. This read combined with the read from state 1 contains the data for a 200 line spectrum.
State 4	This state is used to process the data read in state 3
State 5	This state is used to read the next 180 words from the XM-120 spectrum/waveform. This read combined with the read from states 1 and 3 contains the data for a 400 line spectrum.
State 6	This state is used to process the data read in state 5

State	Description
State 7	This state is used to read the next 420 words from the XM-120 spectrum/waveform. This read combined with the read from states 1, 3, and 5 contains the data for a800 line spectrum.
State 8	This state is used to process the data read in state 7
State 9	This state will perform 2 actions. Action 1 modifies the index pointer for the channel index and changes the instance number in the message command. Action 2 updates the ICM tags for both channel 1 and channel 2 spectrums if not being read from Emonitor

The “program_Read_XM120_TimeWaveform” is the program that is used to read the time waveform data. The “routine_Main1” routine is the main routine. It contains the state machine (10-states) that is used to control the reading of the time waveform. The “routine_Initialize” routine is called from the main routine to initialize data. The “routine_DecodeMove” routine is used to handle state errors and process data associated with many of the states.

State	Description
State 0	This state is used to reset all the buffer data
State 1	This state is used to read the first 180 words from the XM-120 spectrum/waveform buffer and to read in the current speed from the tachometer. This read contains the data for 360 points in the time waveform.
State 2	This state is used to process the data read in state 1
State 3	This state is used to read the next 120 words from the XM-120 spectrum/waveform. This read combined with the read from state 1 contains 600 data points.
State 4	This state is used to process the data read in state 3
State 5	This state is used to read the next 240 words from the XM-120 spectrum/waveform. This read combined with the read from states 1 and 3 contains 1080 data points.
State 6	This state is used to process the data read in state 5
State 7	This state is used to read the next 540 words from the XM-120 spectrum/waveform. This read combined with the read from states 1, 3, and 5 contains 2160 data points. (2048 data points for the time waveform)*
State 8	This state is used to process the data read in state 7
State 9	This state performs 2 actions. Action 1 modifies the index pointer for the channel index and changes the instance number in the message command. Action 2 updates the ICM tags for both channel 1 and channel 2 time waveforms if not being read from Emonitor

* Maximum allowed in XM, must be power of two to allow FFT calculation.

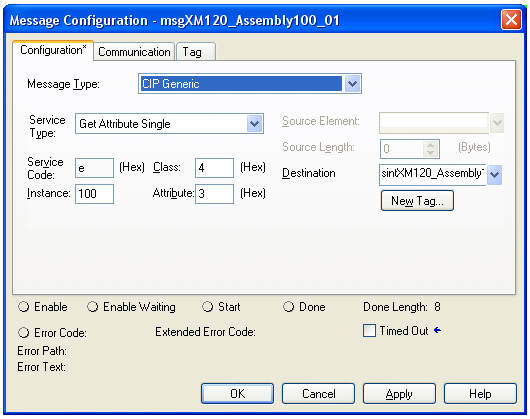
Reading data from the XM module

Reading the assembly data from the XM module

In the XM-120 module, there are two assemblies that are read. Assembly 100 contains the default COS (change of state) data for the DeviceNet Communications. Assembly 101 is the default Poll Response Message for the DeviceNet Communications.

To read the assembly 100 data from the XM-120 module, a message instruction must be executed. The information for the message instruction can be found in the XM-120 module user manual, in appendix C.

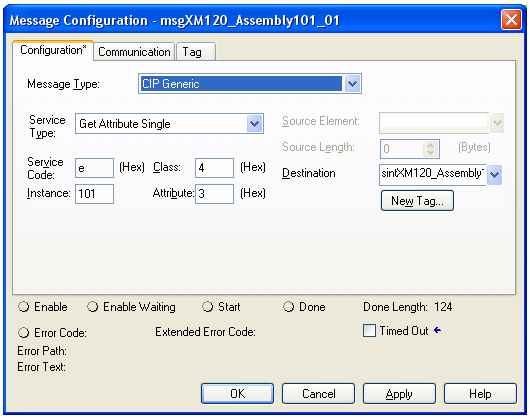
Below is the message instruction's configuration tab for assembly 100 read.



The tag in the destination field is "sintXM120_Assembly100_01". This is a SINT array tag.

To read the assembly 101 data from the XM-120 module, a message instruction must be executed. The information for the message instruction can be found in the XM-120 module user manual, in appendix C.

Below is the message instruction's configuration tab for assembly 101 read.



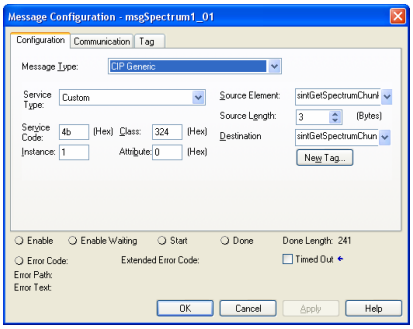
The tag in the destination field is "sintXM120_Assembly101_01". This is a SINT array tag.

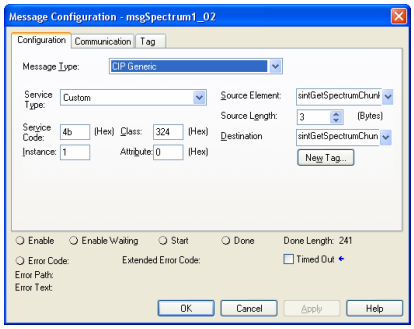
Reading the spectrum data from the XM module

In the XM-120 module, the spectrum for each channel is read. The XM-120 spectrum is a complex spectrum only. To read the spectrum data from the XM-120 module, message instructions must be executed. The information for the message instruction can be found in the XM-120 module user manual, in appendix C.

The source element data for each execution of a message instruction is loaded prior to the execution and this is done in the routineDecodeMode.

Based on the state logic, the configuration tabs of State 1 message instructions contain the following:





The message instructions for the other states have similar configurations. The information that is loaded into the source element data changes per message instruction.

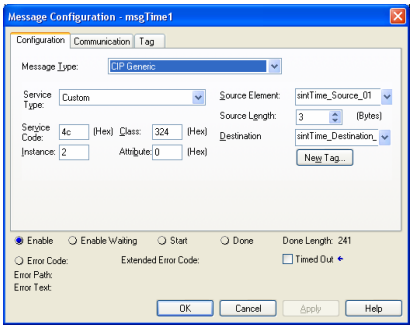
If accessing channel 1 data the Instance number is 1. If accessing channel 2 data the instance is 2.

Reading the time waveform data from the XM module

In the XM-120 module, the time waveform for each channel is read. To read the time waveform data from the XM-120 module, message instructions must be executed. The information for the message instruction can be found in the XM-120 module user manual, in appendix C.

The source element data for each execution of a message instruction is loaded prior to the execution and this is done in the routineDecodeMode.

Based on the state logic, the configuration tabs of one of the State 1 message instructions contain the following:



The rest of the message instructions for the states have similar configurations. The source element and destination element change tag names. The information that is loaded into the source element data changes per message instruction.

If accessing channel 1 data the Instance number is 1. If accessing channel 2 data the instance is 2.

www.rockwellautomation.com

Power, Control and Information Solutions Headquarters

Americas: Rockwell Automation, 1201 South Second Street, Milwaukee, WI 53204-2496 USA, Tel: (1) 414.382.2000, Fax: (1) 414.382.4444
Europe/Middle East/Africa: Rockwell Automation NV, Pegasus Park, De Kleetlaan 12a, 1831 Diegem, Belgium, Tel: (32) 2 663 0600, Fax: (32) 2 663 0640
Asia Pacific: Rockwell Automation, Level 14, Core F, Cyberport 3, 100 Cyberport Road, Hong Kong, Tel: (852) 2887 4788, Fax: (852) 2508 1846

Publication EMONTR-AT002A-EN-E – September 2011