Technical Information

MX100 Performance Specifications

TI 04M08B01-00E

TI 04M08B01-00E

2nd Edition Sep. 2005

MX100 Performance Specifications

TI 04M08B01-00E 2nd Edition

CONTENTS

Foreword 3
1. Resolution of A/D Conversion 4
2. Accuracy of Reference Junction Compensation when Measuring Negative Temperatures 5
3. Withstand Voltages (Insulation) of MX System 6
4. Noise Rejection Performance of the MX System 8
5. Time Stamping of MX System 17
6. Using Strain Conversion Cables (DV450-001) 27
7. Computing Functions of Yokogawa-Developed PC Software 31
   7.1 Examples: Controlling Operation Using Properties of Measurement Channels 33
   7.2 Examples: Counting Events 36
   7.3 Examples: Using Time 37
   7.4 Examples: Calculating Statistics 38
   7.5 Examples: Operation, Output (Release 2 or Later) 40
   7.6 Examples: Miscellaneous 42
   7.7 Notes When Writing Computational Expressions and Techniques to Avoid Problems (Description) 43
8. Comparison between Functions of MX100 and DARWIN DA100 Data Acquisition Units 50

Foreword

This technical information brochure addresses hard-to-understand and important items, among the general specifications of the MX100 PC-based data acquisition unit. When selecting these items, the editorial staff also referred to the content of FAQs it has received from users regarding the DARWIN series systems. Yokogawa Electric hopes this brochure will be helpful for readers in solving problems that may be encountered when using the MX100.

The editorial staff plans to update this brochure as necessary, in keeping with the expansion of functions provided the MX series of units, as well as according to user feedback. Your input of comments and requests to the Product Marketing Department of the Network Solutions Business Division of Yokogawa Electric Corporation regarding this brochure will be greatly appreciated.

For inquiries, please contact us at:

E-mail: daqmaster@cs.jp.yokogawa.com

1. Resolution of A/D Conversion

The A/D conversion resolution of the MX110-UNV-H04 high-speed universal input module and MX110-UNV-M10 medium-speed universal input module is specified as 16 bits (±20000/6000). This section briefly explains what this specification means.

Since 16 bits = 2^{16} = 65536, you may consider that the measured value has a resolution of 1/65536 for the given measuring range. In the MX system, however, the measured value is shown at a resolution of ±20000, i.e., 1/40000, or at a resolution of ±6000, i.e., 1/12000, for a given measuring range, in order to provide the highest resolution easily understandable to humans. A determination as to whether the input signal is measured at ±20000 or ±6000, is made depending on which of them is better suited for the given measuring range. For example, the data sheet states that the highest resolution is 100 μV (0.1 mV) (= 4 V/40000) for the 2 V range (-2 V to +2 V) and 1 mV (= 12 V/12000) for the 6 V range (-6 V to +6 V).

On the other hand, there are two particular ranges, among the measuring ranges of the MX system, at which the input signal is measured with a resolution of 1/60000. These are the 60 mV (0 to 60 mV) and 6 V (0 to 6 V) ranges indicated in the Special Input Range section of the specifications sheet. By limiting the input ranges to the positive side only, their resolution is thus made one order of magnitude higher than that of the standard 60 mV and 6 V ranges. Specifically, the highest resolution of the 6 V range is 100 μV (0.1 mV) (= 6 V/60000). Note that these special input ranges are only available with the optional software MXLOGGER, one of the genuine Yokogawa line of software products.

2. Accuracy of Reference Junction Compensation when Measuring Negative Temperatures

The Reference Junction Compensation Accuracy section of the specifications sheet indicates accuracy values, such as ±1°C and ±0.5°C, with the condition "when measuring 0°C or higher." However, what values would the reference junction compensation accuracy have if negative temperatures were measured?

The reference junction compensation accuracy deteriorates when negative temperatures are measured. (This statement is made based on the premise that the object of temperature measurement is subject to a negative temperature, rather than that the main module of the MX system is subject to a negative temperature. Note that the operating temperature range of the main module is specified as 0 to 50°C.)

In the MX system, transistors are used as the reference junction compensators. The transistor generates a voltage according to the temperature of the measuring instrument's terminals. From this voltage and the voltage (thermoelectromotive force) provided by a thermocouple used for actual measurement, a voltage (thermoelectromotive force) when the reference junction is at 0°C is calculated. This voltage is cross-checked with the thermoelectromotive force curve of the thermocouple being used, and output as a temperature value.

At this point, let us consider the question, "Why does the reference junction compensation accuracy deteriorate on the negative temperature side?"

The answer is that the slope indicating the relationship between the thermocouple's temperature and voltage is not linear (see technical reference books). Generally speaking, the thermoelectromotive force variation of a thermocouple for a temperature change is smaller on the negative temperature side than on the positive temperature side. Earlier in this section, we stated that in the MX system, a voltage is cross-checked with the thermoelectromotive force curve of the thermocouple being used, and output as a temperature value. This means that when the output voltage error of the transistor is converted into a temperature value, the error will result in a larger value for negative temperature measurement, compared with a value for positive temperature measurement. (For your reference, there are basically no temperature-based changes in the transistor's output voltage over the 0–50°C temperature range.)

The following table shows an example of the temperature-by-temperature accuracy of reference junction compensation.

Types of Thermocouple / Temperature being Measured	K	E	J	T	L	U
23°C	±0.5°C	±0.5°C	±0.5°C	±0.5°C	±0.5°C	±0.5°C
0°C	±0.5°C	±0.5°C	±0.5°C	±0.5°C	±0.5°C	±0.5°C
–50°C	±0.6°C	±0.6°C	±0.6°C	±0.6°C	±0.5°C	±0.5°C
–100°C	±0.7°C	±0.7°C	±0.7°C	±0.7°C	±0.7°C	±0.7°C
–150°C	±0.9°C	±0.9°C	±0.8°C	±0.9°C	±0.7°C	±1.0°C
–200°C	±1.3°C	±1.2°C	±1.2°C	±1.3°C	±0.9°C	±1.0°C

T0201.EPS

3. Withstand Voltages (Insulation) of MX System

The MX system features high withstand voltages. As it is provided with reinforced (double) insulation, the MX system is safe to use even when such voltage levels as shown in the catalog are being constantly applied.

The figure that appears in the catalog is shown below for your review.

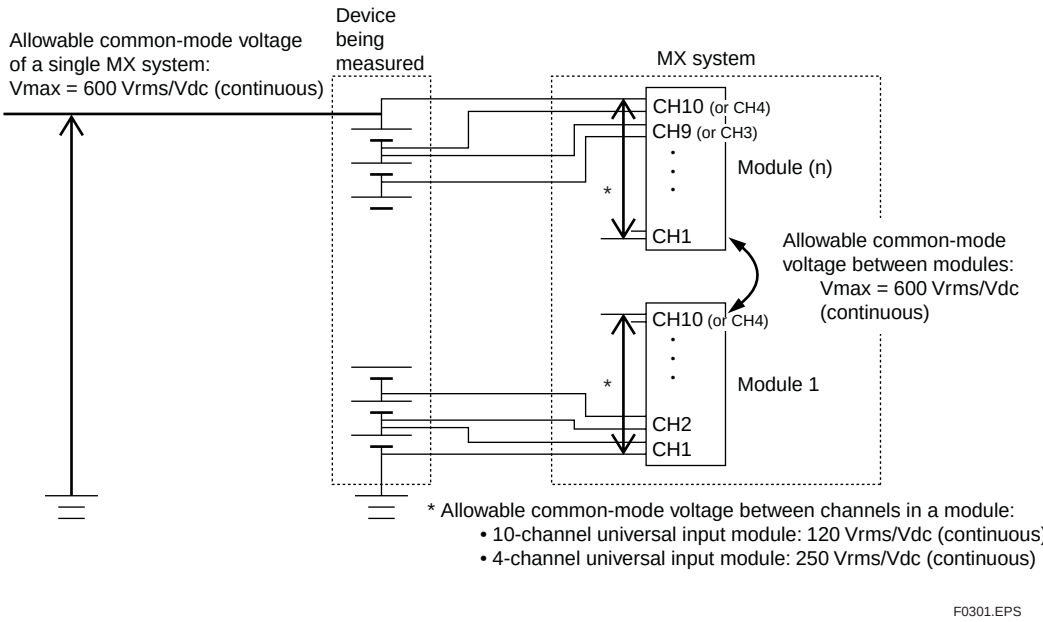


Figure 3-1

Figure 3-1 can be simplified as shown in Figure 3-2.

Assume that V1, V2 and V3 are the common-mode voltages applied to the locations noted below:

V1: between input terminal and case

V2: between channels within the same module

V3: between channels of different modules

Then, the respective common-mode voltages shown in Figure 3-1 are defined as:

V1: the common-mode voltage of a single MX system

V2: the common-mode voltage between channels in a module

V3: the common-mode voltage between modules

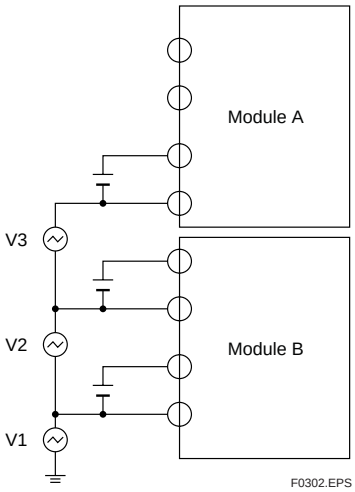


Figure 3-2

Now let us summarize the allowable common-mode voltages of the MX system with reference to Figure 3-2, for comparison with those of the DARWIN system.

		V1: between input terminal and case V3: between channels of different modules	V2: between channels within the same module
MX110-UNV-H04	Allowable common-mode voltage	600 Vrms/Vdc (continuous)	250 Vrms/Vdc (continuous)
	Test voltage (1 min)	3700 Vrms (50/60 Hz)	2300 Vrms (50/60 Hz)
MX110-UNV-M10	Allowable common-mode voltage	600 Vrms/Vdc (continuous)	120 Vrms/Vdc (continuous)
	Test voltage (1 min)	3700 Vrms (50/60 Hz)	1000 Vrms (50/60 Hz)
DARWIN	Allowable common-mode voltage	250 Vac noise voltage	150 Vac noise voltage
	Test voltage (1 min)	1500 Vrms (50/60 Hz)	1000 Vrms (50/60 Hz)

T0301.EPS

Figure 3-3

In the specifications of the DARWIN system, the allowable common-mode voltage is represented as a noise voltage. This is because the DARWIN system is measuring equipment intended for the Safety Extra Low Voltage (SELV) category covering equipment of low voltage measuring ranges. Measuring equipment designed for “SELV” does not require stating allowable continuous voltage levels (i.e., reinforced (double) insulation is assumed). Such representation as discussed above is therefore applied in the case of the DARWIN system. SELV refers to safe voltages pursuant to the IEC1010 standard.

Suppose the allowable common-mode voltage of the DARWIN system is described in the same way as that of the MX system, assuming the DARWIN system complies with requirements for reinforced (double) insulation. Then, the 250 Vac noise voltage is equivalent to 160 Vrms/Vdc (continuous) and the 150 Vac noise voltage is equivalent to 120 Vrms/Vdc (continuous).

For your reference, SELV is defined as noted below (according to IEC1010, 6.3):

Voltage: The voltage level must not exceed 30 Vrms, 42.4 Vpeak, or 60 Vdc.

Current: If the voltage level exceeds any of the abovementioned values, the current level must be no higher than 0.5 mArms for sinusoidal waveforms and 0.7 mApeak for non-sinusoidal waveforms or mixed frequencies.

Restrictions are also placed on the capacitance.

The 250 Vac and 150 Vac noise voltages stated as the allowable common-mode voltages of the DARWIN system are naturally subject to the SELV current limitations. In other words, current levels must be within the abovementioned limits in cases where 250 V or 150 V is applied.

4. Noise Rejection Performance of the MX System

The MX system includes a two-stage noise rejection function involving an A/D conversion method (integrating A/D) and a firmware-based first-order lag filter. The following is a description of the noise rejection function implemented through the A/D conversion method focusing on the normal mode rejection ratio (NMRR) and common rejection ratio (CMRR) used as indices of the noise resistance performance.

• Integrating A/D Converters and their Noise Rejection Performance

An integrating A/D converter integrates measured values over a specified length of time. It is possible to remove a frequency component you wish to eliminate if the specified time length agrees with the period of that frequency component.

An integral time of 20 ms is regarded as being equivalent to 1/50 Hz, which means that the 50 Hz supply frequency and its integer multiples can be removed. This logic is illustrated in Figure 4-1 below:

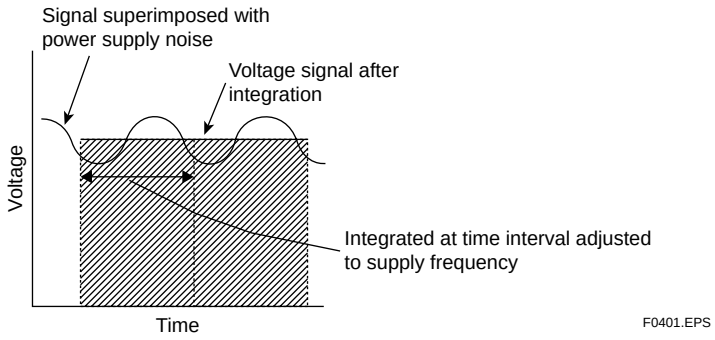


Figure 4-1 Example of Noise Rejection by Integrating A/D Converter

Likewise, an integral time of 16.67 ms means that 60 Hz and its integer multiples can be removed; 100 ms means 10 Hz and its integer multiples can be removed; and 1.67 ms means 600 Hz and its integer multiples can be removed. Accordingly, measurement with an integral time of 1.67 ms remains susceptible to the effects of the supply frequency.

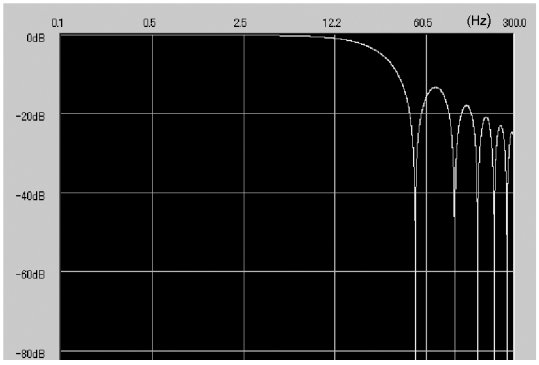
Although integral times of 36.67 ms and 200 ms differ slightly in the process of calculation from the integral times of such simple rectangular integration as discussed above, the 36.67 integral time permits removal of 50 Hz, 60 Hz, and their integer multiples. An integrating A/D converter with a 200 ms integral time, works as a low-pass filter with a cut-off frequency (Fc, which is discussed later) of 5 kHz.

The following chart shows the integral times of the integrating A/D converter included with the MX100 together with the noise rejection performance. Note that the applied integral time is automatically determined based on the type of input module and specified period of measurement. Please see the specifications of each module for more detailed information.

Integral Times of the Integrating A/D Converter and Rejected Noise Frequencies

Integral Time	Rejected Frequencies/Remarks
1.67 ms	600 Hz and its integer multiples
16.67 ms	60 Hz and its integer multiples
20 ms	50 Hz and its integer multiples
36.67 ms	50/60 Hz and their integer multiples
100 ms	10 Hz and its integer multiples
200 ms	Fc=5Hz low pass filter

Now let us take a look at a frequency characteristics graph to see how noise is actually removed. Figure 4-2 shows a frequency characteristics graph when the 20 ms integral time is applied.



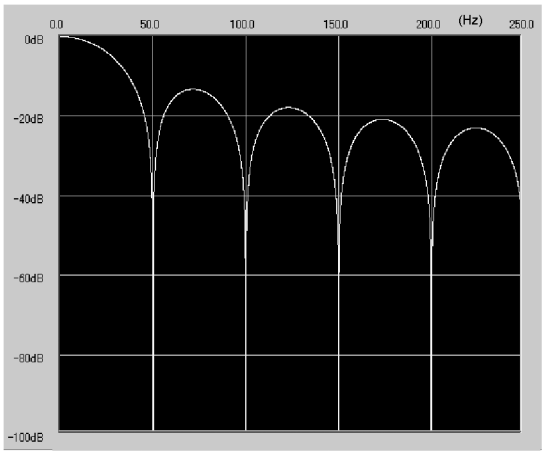
F0402.EPS

Figure 4-2 Graphical View of 20 ms Rectangular Integration with Logarithmic Representation of Vertical Axis (Theoretical Values)

The vertical axis represents the amplitude ratio (gain) and the horizontal axis represents the frequency. The amplitude ratio refers to the ratio of the output signal amplitude to the input signal amplitude and is indicated in decibels (dB). A negative dB value means that the input signal is attenuated before it is output (0 dB means the input signal is not attenuated). From the figure, it can be understood that there is virtually no decrease in the amplitude of signals up to several hertz, while the amplitude decreases significantly for signals of several tens of hertz or higher.

It remains uncertain, however, whether frequency components of 50 Hz and its integer multiples have actually been removed in the figure.

Therefore, we will cancel the logarithmic representation of the horizontal axis and plot frequency data points directly on the horizontal axis. Now we can clearly recognize that the frequency components of 50 Hz and its integer multiples have been sufficiently attenuated (i.e., removed).



F0403.EPS

Figure 4-3 Graphical View of 20 ms Rectangular Integration (Theoretical Values)

Then, what will degrees of attenuation of -20 dB and -40 dB, for example, be like?

The unit dB is expressed as “20log |output signal’s amplitude/input signal’s amplitude|”. A positive value of dB represents amplification, while a negative value means attenuation.

An attenuation of -20 dB is calculated as:

-20 dB = 20log |output signal’s amplitude/input signal’s amplitude|

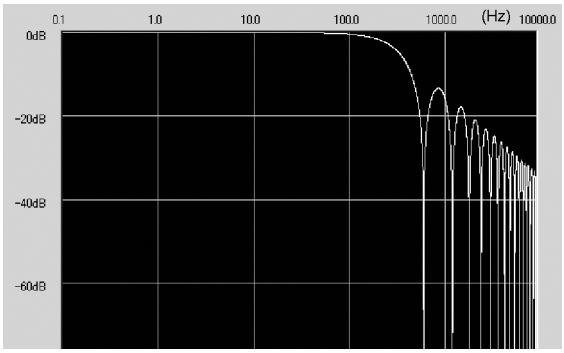
log |output signal’s amplitude/input signal’s amplitude| = -1

Output signal's amplitude/input signal's amplitude = 1/10

Likewise, -40 dB is calculated as an attenuation of 1/100, -6 dB as 1/2, and -3 dB as $1/\sqrt{2}$.

Now, let us take a look at the frequency characteristics of 1.67 ms integration.

As discussed earlier, this integration enables the frequency components of 600 Hz and its integer multiples to be removed. As is evident from Figure 4-4, however, this integration cannot remove 50 and 60 Hz components. In other words, measurement with this integral time is not immune to the effects of the supply frequency.

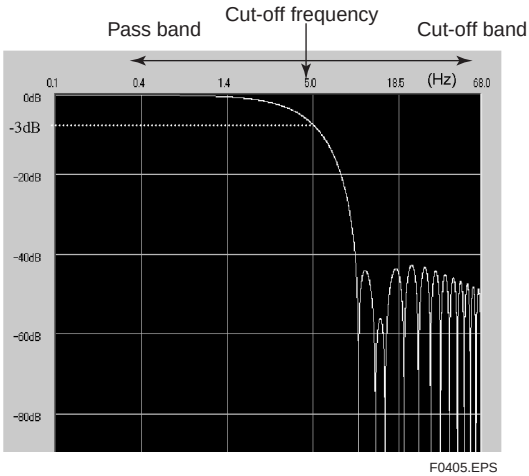


F0404.EPS

Figure 4-4 Graphical View of 1.67 ms Rectangular Integration (Theoretical Values)

In the case of 200 ms integration, the A/D converter serves as a low-pass filter (for letting low frequencies pass through) with a cut-off frequency of 5 Hz, as shown in Figure 4-5. The converter may also be described as a low-pass filer capable of rejecting noise components of 5 Hz and its integer multiples. Since the performance of rejection is greater for 5 Hz and higher frequencies, the specifications sheet describes the A/D converter as a low-pass filter with a cut-off frequency of 5 Hz.

A cut-off frequency refers to a frequency at which the amplitude ratio equals -3 dB (the point where the amplitude decreases by a factor of $\sqrt{2}$). A band of frequencies lower than the cut-off frequency is referred to as a pass band and a band of frequencies higher than the cut-off frequency is referred to as a cut-off band. The cut-off frequency is used as an index for representing filter performance.



F0405.EPS

Figure 4-5 Graphical View of 200 ms Rectangular Integration (Theoretical Values)

• Normal-Mode and Common-Mode Rejection Ratios

Now let us refer to the specifications of the MX main module, where the following statement is included.

Normal-mode rejection ratio (NMRR): 40 dB minimum for an integral time of 16.67 ms or longer (50/60 Hz±0.1%)

This statement means that if a normal-mode noise signal of 50/60 Hz is imposed when executing measurement with the integral time set to 16.67 ms or longer, such 50/60 Hz noise components can be removed at a rejection ratio of 40 dB or higher.

The term “normal-mode noise” refers to noise that is imposed across the High and Low terminals, as shown in Figure 4-6.

A rejection ratio of 40 dB indicates that as discussed earlier, the A/D converter has a noise rejection capability which is represented as “Vout/Vnm = 1/100.”

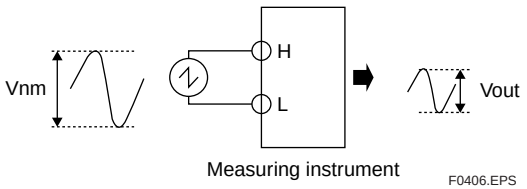


Figure 4-6

The graphs of frequency characteristics in Figures 4-2 to 4-5 that we have examined represent this normal-mode rejection performance.

The following statement is also included in the specifications.

Common-mode rejection ratio (CMRR): 120 dB minimum for an integral time of 16.67 ms or longer (50/60 Hz±0.1%)

This statement means that if a common-mode noise signal of 50/60 Hz is imposed when executing measurement with the integral time set to 16.67 ms or longer, such 50/60 Hz noise components can be removed at a rejection ratio of 120 dB or higher.

The term “common-mode noise” refers to noise that is imposed in an in-phase manner across the High and Low terminals, as shown in Figure 4-7. (Note that common-mode noise differs from normal-mode noise in the way it is imposed on the terminals.)

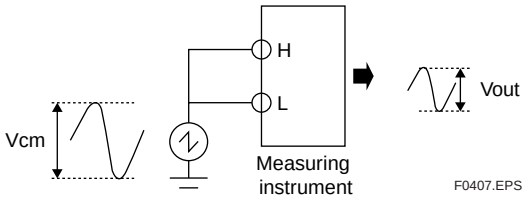


Figure 4-7

Since common-mode noise is imposed in an in-phase manner, it does not inherently affect measurement as long as an electrical difference between the High and Low terminals remains uniform. Even if noise is imposed on the High and Low sides in an in-phase manner, an unbalanced state internal to the measuring instrument will occur between the sides. This unbalanced state usually results in normal-mode noise, affecting the measured value. It is a common complaint that measurement cannot be conducted successfully because there is supply frequency noise interference in the signal lines. In most cases, this complaint is due to the abovementioned phenomenon.

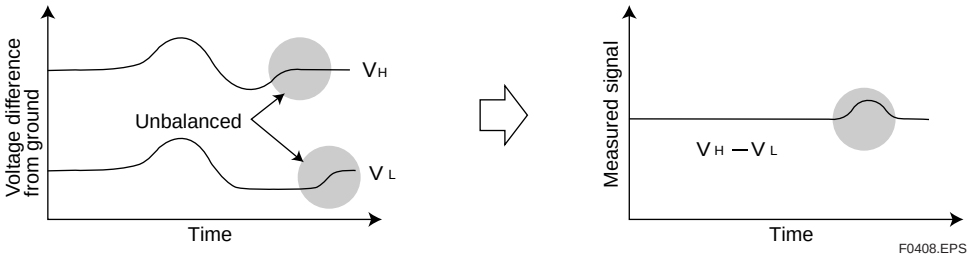


Figure 4-8 Effects of an Unbalanced State Caused by Common-Mode Noise (Conceptual View)

Then, what degree of error is a common-mode rejection ratio of 120 dB expected to cause?

Since 120 dB means $1/10^6$, interference by a common-mode noise voltage of 100 V results in an error of

$100\ V/10^6 = 100\ \mu V$

which is equivalent to an approximate temperature error of 2 to 3°C for a type-K thermocouple.

Likewise, interference by a common-mode noise voltage of 250 V results in an error of

$250\ V/10^6 = 250\ \mu V$

which is equivalent to an approximate temperature error of 6°C for a type-K thermocouple.

• Superiority of the MX110-UNV-H04 Four-Channel High-Speed Universal Input Module

The MX lineup includes two universal modules enabling measurement of voltage and temperature: the MX110-UNV-H04 Four-Channel High-Speed Universal Input Module, and the MX110-UNV-M10 Ten-Channel Medium-Speed Universal Input Module. The following should be noted regarding the relative performance of these two modules.

The specifications list the noise rejection performance (normal mode rejection ratio and common mode rejection ratio) as the same for both, but when the MX110-UNV-H04 Four-Channel High-Speed Universal Input Module is compared with the MX110-UNV-M10 Ten-Channel Medium-Speed Universal Input Module:

- At the same measurement interval, the integral time of the integrating A/D converter is longer (faster measurements can be taken at the same A/D integration period)
- Since an A/D converter is assigned independently on each channel, the measurement circuit is simpler, and the amount of conversion of high frequencies from common mode noise to normal mode noise is small.

These two factors mean that the MX110-UNV-H04 could be said to be superior in terms of noise resistance.

If noise resistance during high speed measurement and at high frequencies is called for, the MX110-UNV-H04 Four-Channel High-Speed Universal Input Module is certainly recommended.

• First-Order Lag Filters

The MX system is designed so that a first-order lag filter can be applied by means of firmware processing at a point where the measured value is finally output. This strategy has the same effect as the case where a low-pass filter comprising a resistor and a capacitor is provided in the final stage of the MX system.

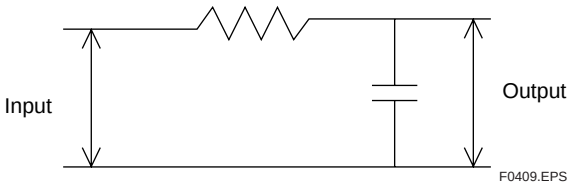


Figure 4-9

A first-order lag system refers to a system that presents such output characteristics (exponential characteristic curve) as shown in Figure 4-11 when such a step input as shown in Figure 4-10 is applied.

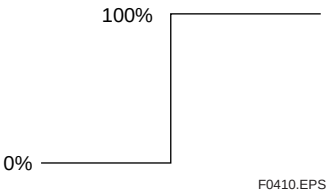


Figure 4-10

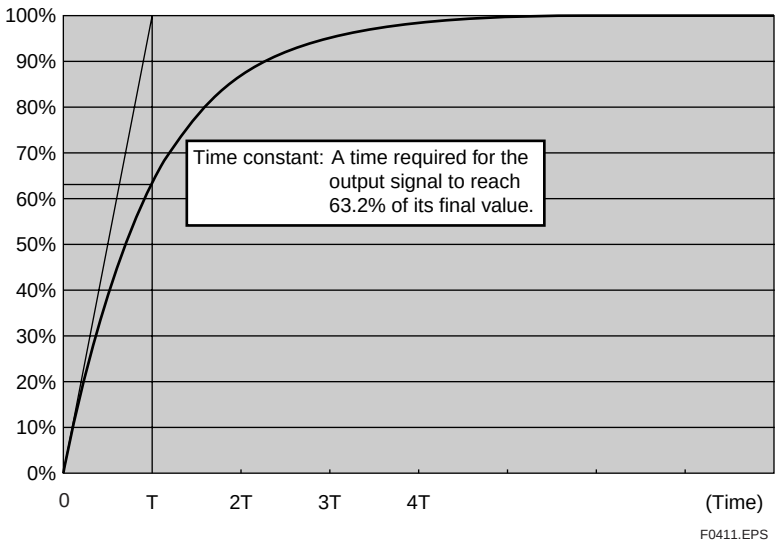


Figure 4-11

The time constant indicated in the specifications sheet refers to point T in the above graph. This means, longer time constants require longer periods of time before the 100% output value is reached, and the filtering function works to a higher extent (see Figure 4-12). The MX system provides a wide choice of time constants ranging from 5 to 100 times the measurement period, thus allowing users to set a filter according to the type of noise they want to remove.

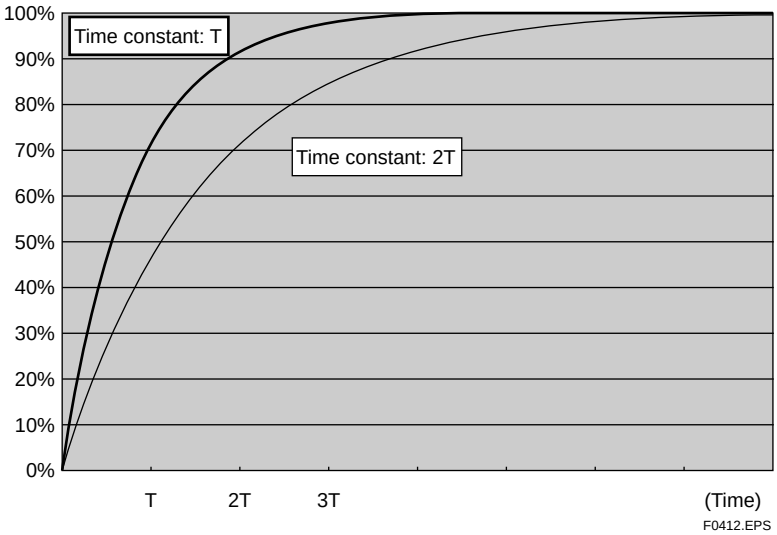


Figure 4-12

Now we examine the frequency response of a first-order lag system. As is apparent from Figure 4-13, the system does not have such a characteristic as to exclusively remove any specific frequency component. The system is characterized by a gradual increase in its rejection performance with an increase in the frequency.

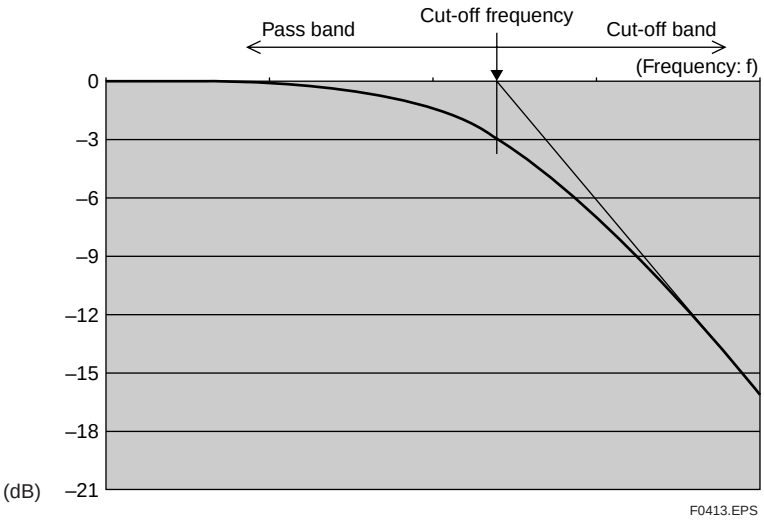


Figure 4-13

• Tip: Moving Average

In practical applications, Yokogawa considers it sufficiently effective to use both an integrating A/D converter and a first-order lag filter in order to achieve noise rejection. For this reason, the MX system has no moving average function.

Then, how does a moving average function differ from a first-order lag filter? For reference purposes, a graph of frequency characteristics is shown in Figure 4-14, where a moving average is taken at a 10 ms sampling interval from five measurement points. As is evident from the figure, a moving average method has the property of removing specific frequency components in the same manner as an integrating A/D converter. First-order lag filters (Figure 4-13) have no such property.

In general, however, noise frequencies of fixed values with regard to noise components have never been encountered, except for supply frequency noise. Although a first-order lag filter does not completely cover the functionality of the moving average method, the method of “removing supply frequency noise with the integrating A/D converter and other noise with the first-order lag filter” serves the primary purpose of noise rejection.

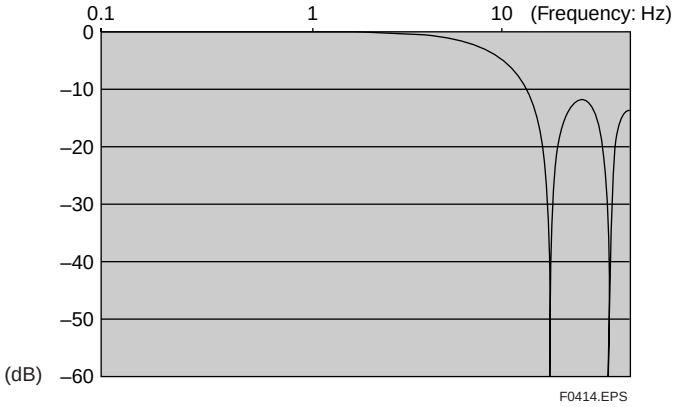


Figure 4-14

Finally, the practical effects (theoretical values) of the filter and the moving-average method are discussed below.

In Figures 4-15 and 4-16, the waveform “Input” plotted with ◆ denotes the noise waveform being input; the waveform “Filter Out” plotted with ■ denotes the filtered output waveform; and the waveform “Moving Average” plotted with ▲ denotes the moving-averaged output waveform.

From these figures, it is evident that there are no practical differences between the functions of the filter and the moving-average method.

Note that the vertical axis has no unit of measure. Use the scale values as indexes for comparing the magnitudes of input noise and filtered noise.

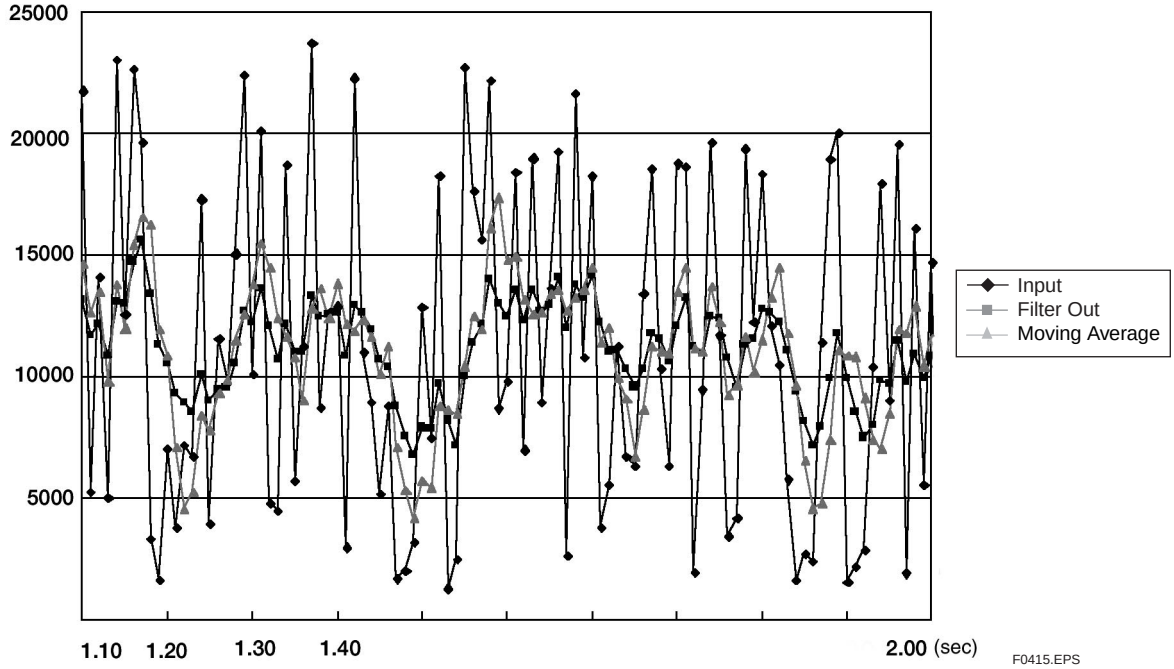


Figure 4-15 Filter: A Filter with a 10 ms Measurement Interval and a 0.05 s Time Constant Moving Average: 5-Point Moving Average

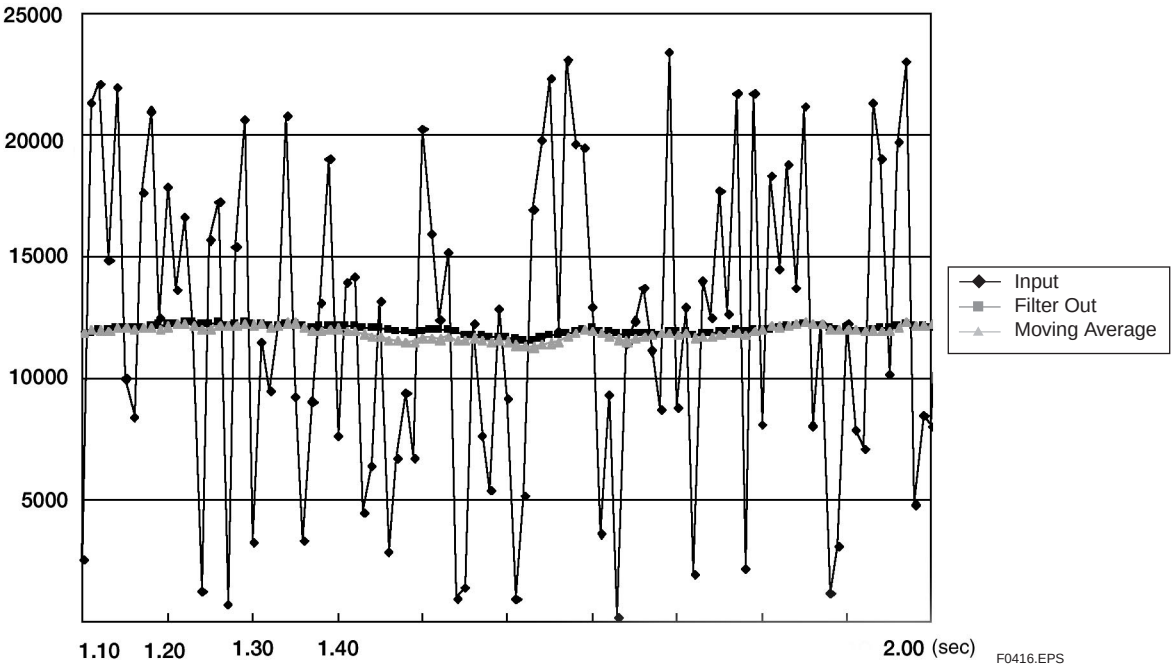
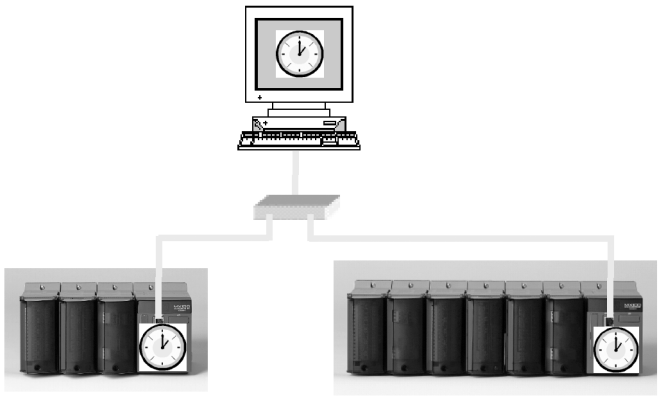


Figure 4-16 Filter: A Filter with a 10 ms Measurement Interval and a 1 s Time Constant Moving Average: 100-point Moving Average

5. Time Stamping of MX System

• MX System Clock

There are two types of time information appended to data provided by the MX system. One is the time information from the clock built into the MX system's main module, and the other is the time information from the PC. This section explains the time sequence of the MX system. Note that we assume here that Yokogawa-developed software is used as the PC software.



F0501.EPS

Figure 5-1

• Time Management within a Single Unit

F0502.EPS

Figure 5-2

The time information of a single MX-series unit is controlled by the clock of an MX100 main module. This clock ticks at 100 ms intervals, independent of the PC clock.

Each input module is set to start data acquisition separately according to the clock (ticking at 100 ms intervals) of the closest MX100 main module after a request from the PC to start measurement. Measured data is compiled by the main module on a measurement-interval basis and transferred to the PC upon an output request from the PC.

Now we examine the time sequence, focusing solely on an input module.

An input module is not capable of immediately starting measurement upon a request for same. A certain length of time is required for the input module to examine the A/D conversion error and RJC temperature before starting measurement. The time slot labelled “Initial Cal. (short for calibration)” in Figure 5-3 refers to this time length. This calibration is performed continuously even during measurement.

Although measurement begins after initial calibration, the input module is not designed to produce output-ready data on a measurement-interval basis. A certain length of time is also required to execute various processes within the main module, such as data transfer from the input module to the main module, calculation (inter-channel difference and scaling), alarm processing, and filtering.

The time when the input module is actually ready for outputting data arrives two to three measurement intervals after the given measurement time point. (The time required for initial calibration and data storage varies depending on the measurement interval.)

Specific time sequences of measurement are discussed below.

• Time Sequence of MX110-UNV-H04 Four-Channel High-Speed Universal Input Module

The individual channels of this input module take measurements independent of one another (because the module has four A/D converters). Accordingly, it could be said that the time stamps of these individual channels are synchronous with one another.

In cases where measurement intervals are 10 ms and 50 ms, the input module transfers a collection of data to the PC at the shortest clock interval of 100 ms. Figure 5-3 shows how one item of output data is produced when measured with a 10 ms sampling interval.

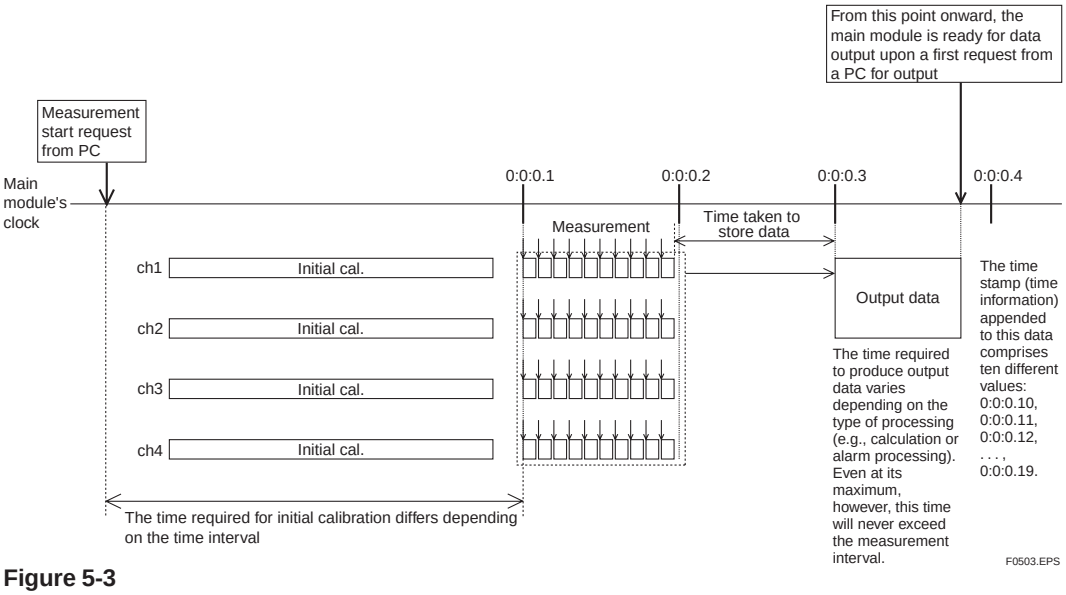


Figure 5-3

The input module performs measurement and calibration during an interval of 10 ms.

• Time Sequence of MX110-UNV-M10 Ten-Channel Medium-Speed Universal Input Module

This input module has only one A/D converter and scans the ten channels at a specified measurement interval.

Figure 5-4 shows how one item of output data is produced when measured at a 100 ms sampling interval. Since the input module scans from channel 1 to channel 10 at 100 ms, the channel-to-channel synchronization of time stamps of data can be interpreted as “synchronization within the measurement interval.”

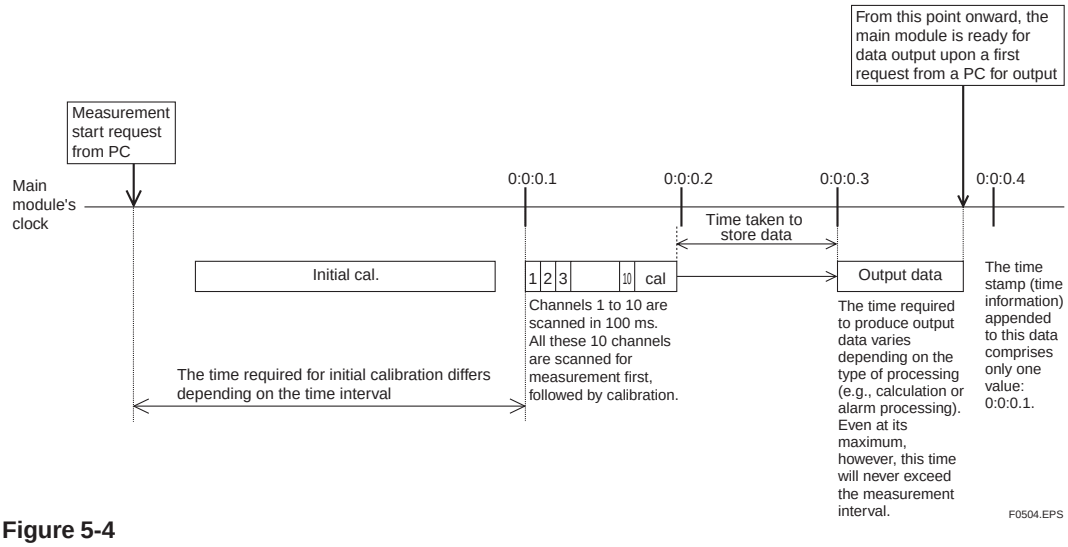


Figure 5-4

In the figure above, we discussed how one item of output data is produced. Now we will show an example of time sequence in actual continuous measurement.

In the case of analog input modules, the time sequence from the start of measurement to the completion of output data creation varies depending on the measurement interval and the combination of modules types. It should be noted here that depending on this combination, differences will occur in the presence/absence of initial calibration and the time required before output data is created. Also note that calibration during measurement is not mentioned in those figures; rather, only the word “measurement” is indicated to signify “measurement and calibration.”

Time Sequence for 10 ms and 50 ms Measurement Intervals of MX110-UNV-H04 Four-Channel High-Speed Universal Input Module and 100 ms Measurement Interval of MX110-UNV-M10 Ten-Channel Medium-Speed Universal Input Module

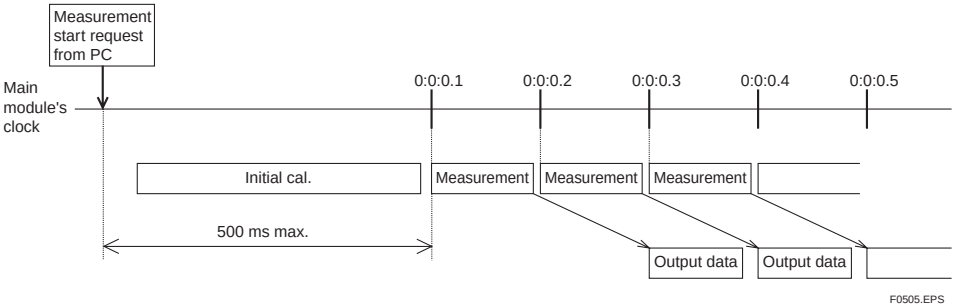


Figure 5-5

Time Sequence for 100 ms Measurement Interval of MX110-UNV-H04 Four-Channel High-Speed Universal Input Module

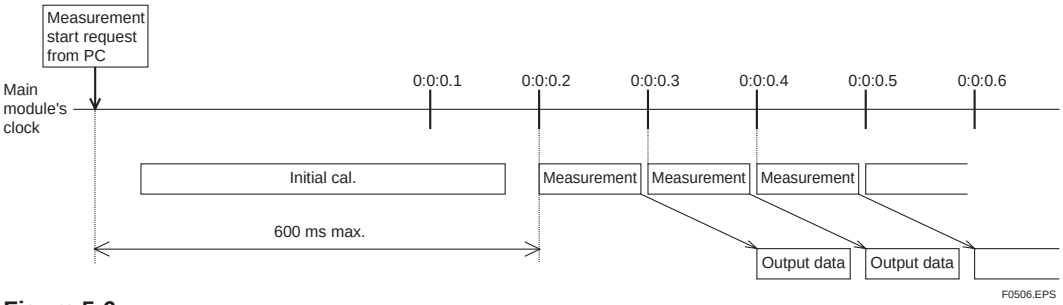


Figure 5-6

Time Sequence for 200 ms, 500 ms and 1 s Measurement Intervals of MX110-UNV-H04 Four-Channel High-Speed Universal Input Module and 200 ms, 500 ms and 2 s Measurement Intervals of MX110-UNV-M10 Ten-Channel Medium-Speed Universal Input Module

Measurement interval cal T(ms) / Module	200 ms	500 ms	1 s	2 s
MX110-UNV-H04	600	900	1300	200
MX110-UNV-M10	500	700	200	1700

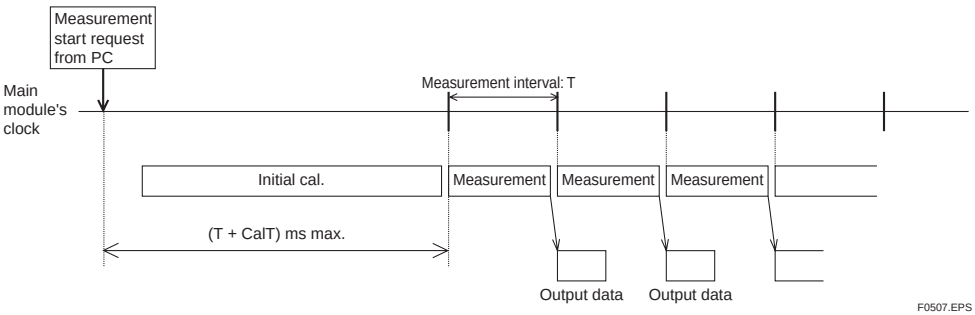


Figure 5-7

Time Sequence for 2 s and Longer Measurement Intervals of MX110-UNV-H04 Four-Channel High-Speed Universal Input Module and 1 s/5 s and Longer Measurement Intervals of MX110-UNV-M10 Ten-Channel Medium-Speed Universal Input Module

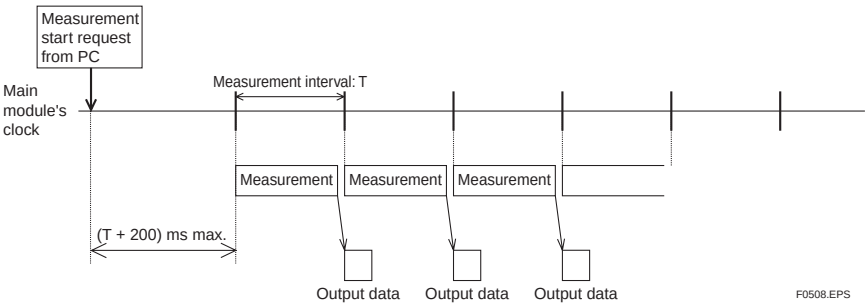


Figure 5-8

Time Sequence for 10 s, 20 s, 30 s and 60 s Measurement Intervals

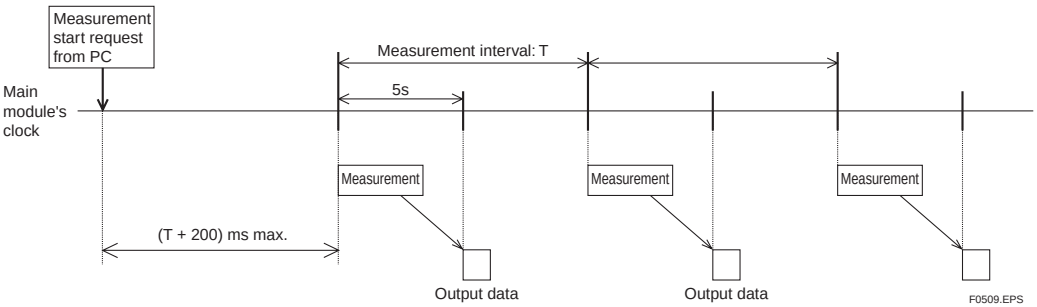


Figure 5-9

• Time Sequence of MX115-D05-H10 Ten-Channel High-Speed Digital Input Module

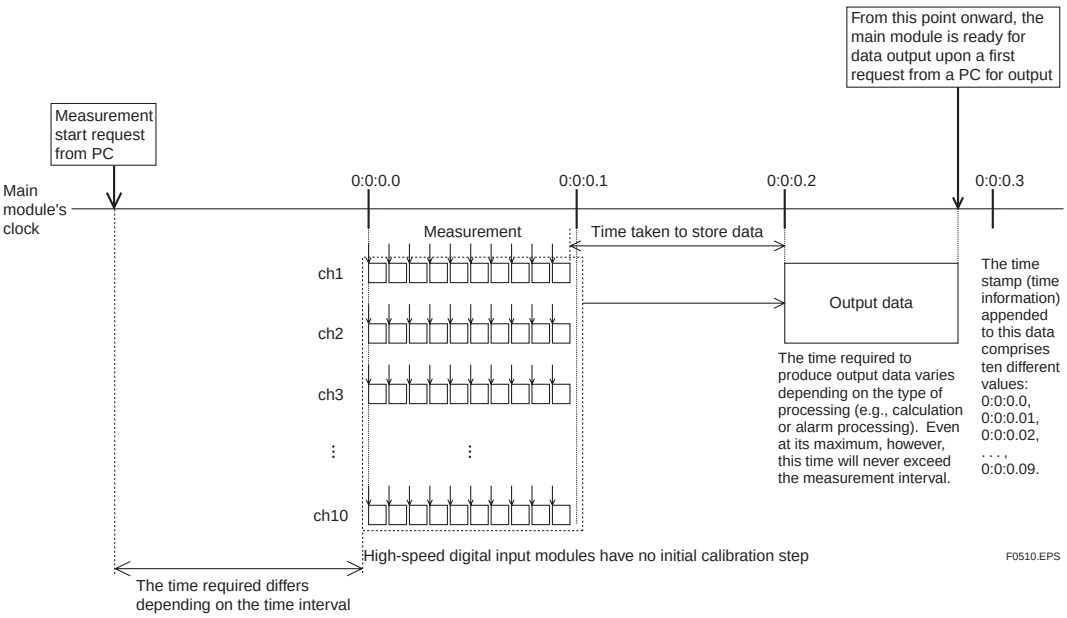
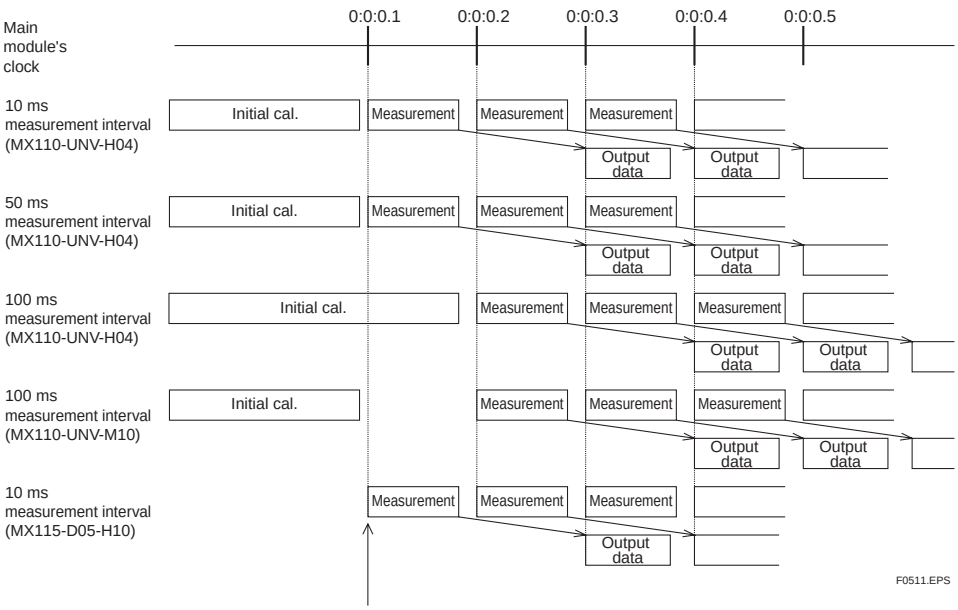


Figure 5-10

Until now, we have been discussing the flow of processing from measurement to output data creation performed at the same measurement interval within a given single module.

The same approach as that used for data acquisition with a single module also applies to data acquisition based on multiple modules. In this case, just think that the time sequences of individual modules are running concurrently.

Note: The measurement starting time for each module within one data acquisition unit lags behind the internal clock (ticking at 100 ms intervals) of a main module by 0.03 ms to 2.00 ms. In Figure 5-11, such a time lag is ignored.



A digital input module has no initial calibration step. If any analog input module having the same measurement interval as that of the digital input module is included in the MX system, the digital input module starts measurement upon completion of initial calibration at the analog input module.

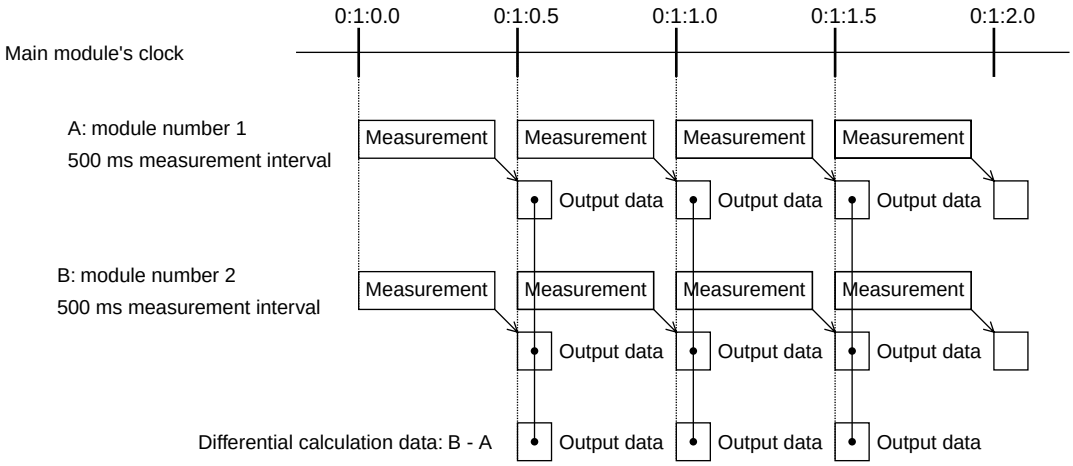
Figure 5-11

- **Inter-channel Calculation**

In order to make the time stamps of target channels strictly precise in inter-channel calculation, it is advisable that a channel of a lower number be assigned as the reference channel within the same module.

Earlier in this section, we explained that a certain length of time is required before any measured value can be turned into output-ready data. This conversion task is performed in increasing order of the modules' measurement intervals, in ascending order of module numbers, and in ascending order of channel numbers, in this order of priority. Consequently, a discrepancy occurs between the data values based on which a differential calculation is made, according to the conditions of the channels for which the differential calculation was made. This logic is illustrated in Figure 5-12. Note that the time stamp of the data based on which a calculation is made varies depending on which channel to assign as the reference channel.

<Cases when reference channel is defined as "A: module number 1">



<Cases when reference channel is defined as "B: module number 2">

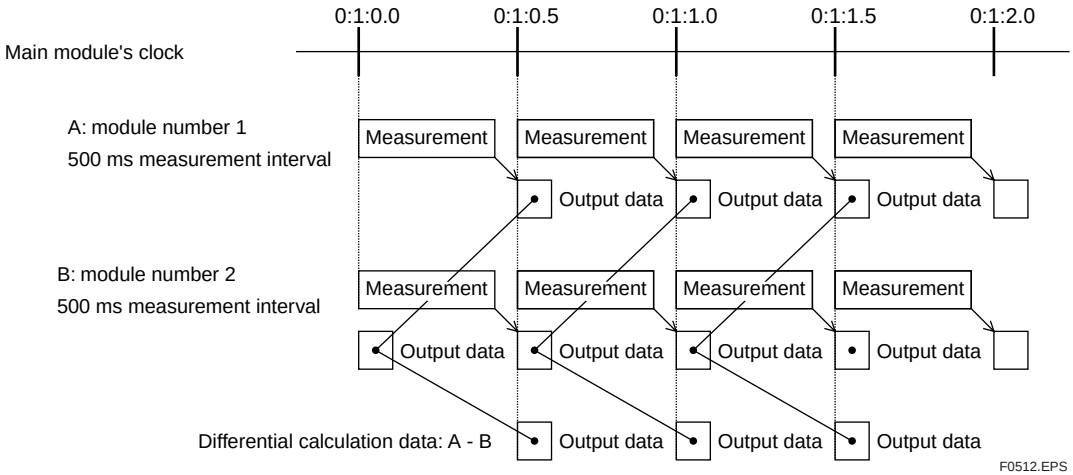


Figure 5-12

• Time Sequence of MX125-MKC-M10 Ten-Channel Medium-Speed Digital Output Module

This digital output module takes measurements at 100 ms intervals in synchronization with the measurement clock. At the time point of data output, the main module executes DO output requests that have arrived by that time.

Figure 5-13 shows that the universal input module has completed its initial calibration and entered a normal measurement step. Note that this digital output module has no initial calibration step.

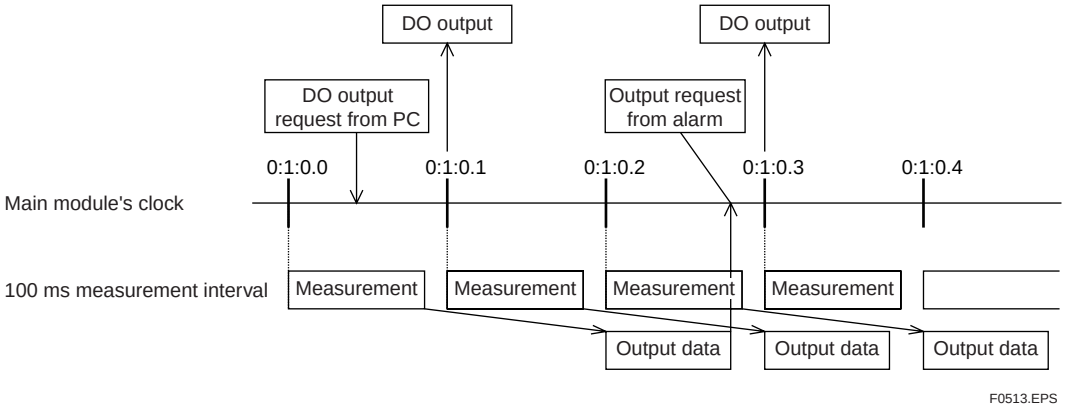


Figure 5-13

• Communication with PC (when Yokogawa-Developed Software is Used)

In response to a data output request from the PC, the MX system returns data available at the moment. At this point, the system also appends the PC's time information to the data before transmitting it. (The PC's time information is sent not only upon request to output data but also at each round of communication. This time information is appended to the most immediate measurement data to be output.)

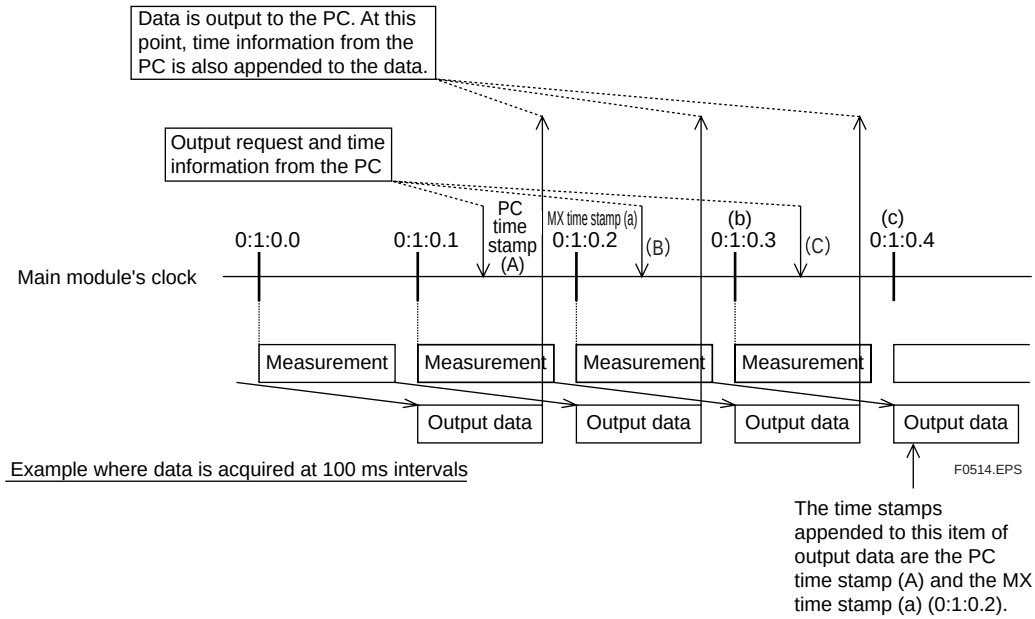


Figure 5-14

Although we have often referred to initial calibration in earlier discussions, users do not have to be conscious of this calibration as long as genuine Yokogawa software is used.

Once the MX main module is in an online state with the PC, the MX system automatically performs initial calibration and goes into a measurement (monitoring) status. Even if the current range is changed to another, the system automatically goes through the required steps up to initial calibration.

- **Realtime Monitoring Screen (when Yokogawa-Developed Software is Used)**

The Realtime Monitoring screen shows data in a visual form according to time information from the MX main module, rather than from the PC.

- **Viewer Screen (when Yokogawa-Developed Software is Used)**

In the case of stored data, however, the time stamp of either the MX main module or the PC can be chosen to visually replay the data. Then, why is it necessary to show the time information of the PC?

Let us consider the following case where data is obtained from two MX units using a single PC.

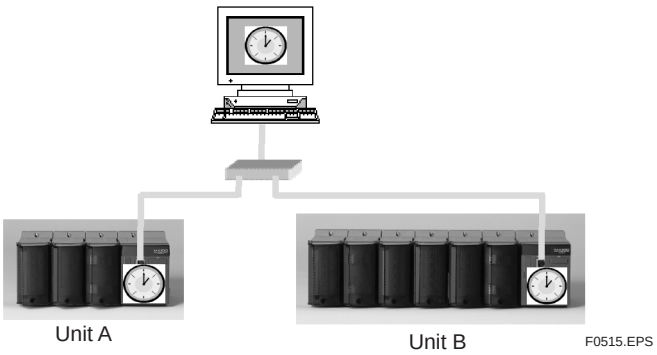


Figure 5-15

Suppose the clocks of units A and B are in perfect synchronization with each other. Then, data acquisition can be achieved by simply relying on time information from the MX units only. In practice, however, this synchronization is not possible.

The degree of time disagreement between the MX units depends on their clock accuracy. The clock accuracy of the MX main module is specified as ±100 ppm. This clock accuracy results in a weekly error of $(3600 \times 24 \times 7)/1000000 \times 100 \approx 60$ sec. In other words, in a case where data acquisition is performed continuously for one week at an interval of one second, the number of data items acquired may differ between units A and B to a maximum of 120 items (60×2) even though the data is acquired over the same period. In order to absorb this difference, the PC clock time is used to correct the number of data items before they are processed for visual replay.

Now we explain the idea of how data time stamps are corrected. (For easier understanding, the number of data items and the measurement time are represented in an exaggerated manner in the following example.)

Assume that data is acquired for five seconds based on the PC clock time, and seven data items are returned from the MX main module during that period. First, the measured values are interpolated using the seven data items to create a waveform. Next, the values at the given time points of the PC are defined as the measured values based on the PC clock time.

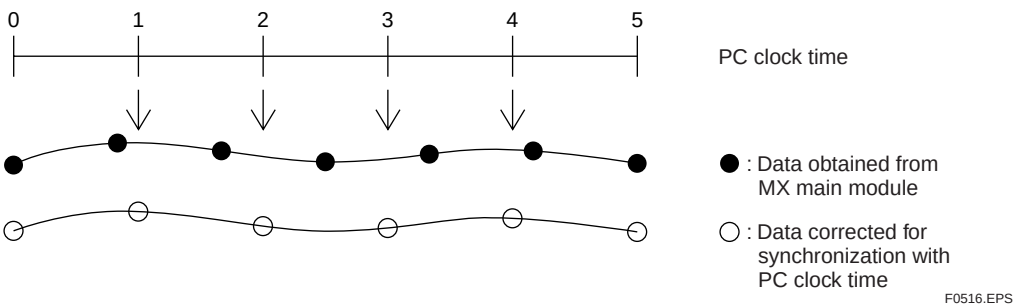


Figure 5-16

By applying this process to both units A and B, the data items of these units are caused to hold measured values corrected according to the PC clock time. As long as the time axes of these units agree with each other, it is possible to draw waveforms spanning the units by properly arranging these data items.

There is a difference no greater than one measurement interval (100 ms at the shortest) between the PC time stamp appended to the data of the MX units and the time at which the data was actually sampled. (See the example of Figure 5-14 “Difference between PC Time Stamp (A) and MX Time Stamp (a)”.) However, the time difference between the units is kept smaller than one measurement interval (100 ms at the shortest) at any point of time by means of the correction process noted above. Thus, it is possible to visualize data without allowing the time difference to accumulate.

6. Using Strain Conversion Cables (DV450-001)

The strain module section in the DAQMASTER catalog and user's manual cautions that when using strain gauge type sensors without remote sensing wires with the MX112-NDI-M04, the DV450-001 strain conversion cable (DARWIN accessory, sold separately) should also be used. The following explains why this is necessary.

The MX112-NDI-M04 supports strain gauge type sensors with remote sensing functionality. An overview of the wiring scheme is illustrated in figure 6-1. In this configuration, correct measurements can be obtained simply by connecting the sensor directly to the strain module.

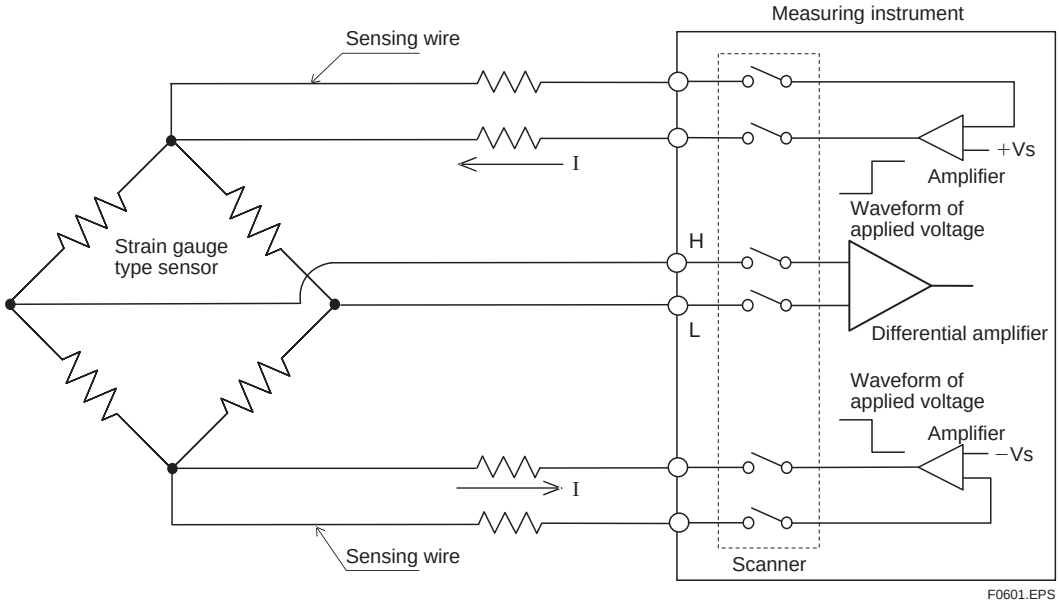


Figure 6-1

Incidentally, many strain gauge type sensors exist that do not have remote sensing functionality. For measurements in which error arising from wiring resistance need not be taken into consideration (when short wiring is used), it seems to be common to use the less expensive sensors without remote sensing wires.

If such sensors are wired to the MX112-NDI-M04 without using the DV450-001, the configuration that results is shown in figure 6-2.

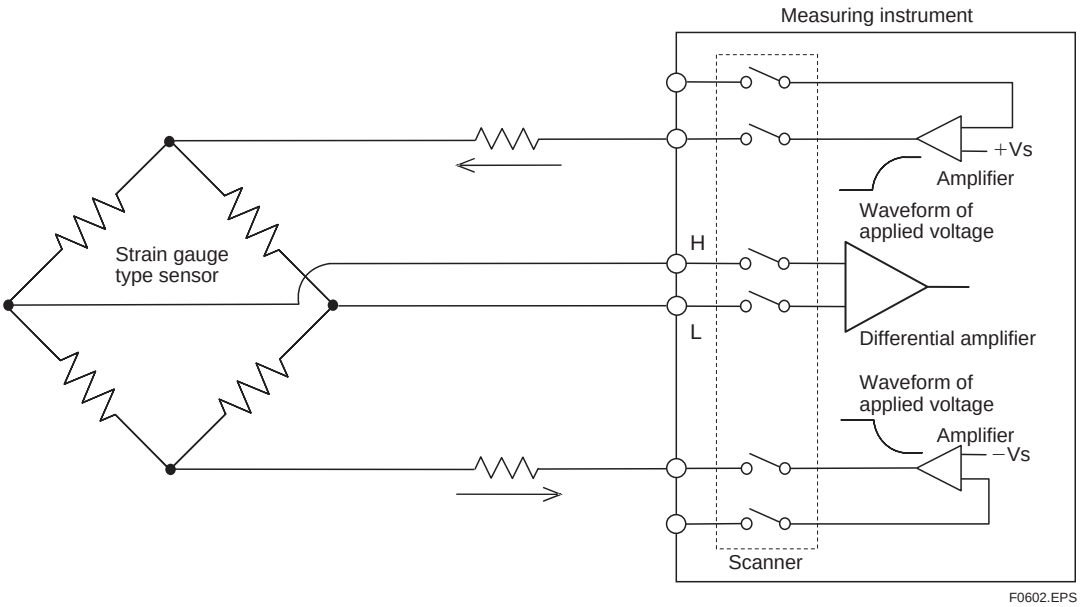


Figure 6-2

Under the conditions in figure 6-2, the MX112 cannot perform measurement. Let us compare the waveforms of applied voltage in figures 6-1 and 6-2. In figure 6-2, you can see that the response of the amp is slow. The reason for the slow response is that the MX112 is using a scanner. Since scanner-type measuring instruments like the MX112 place restrictions on the per-channel measurement time, the amp must respond in a short amount of time. In figure 6-1 the amp detects the bridge voltage from the sensing wire, making it possible to eliminate delay within the duration of a single measurement, but since the amp in figure 6-2 cannot detect the bridge voltage, response delays occur. As a result, in figure 6-2, measurement concludes without an appropriate applied voltage being reached, and the measurement is inaccurate.

The DV450-001 is used to resolve these response delays. Figure 6-3 shows the MX112 wired with a strain gauge type sensor without a remote sensing wire, but with the DV450-001 connected.

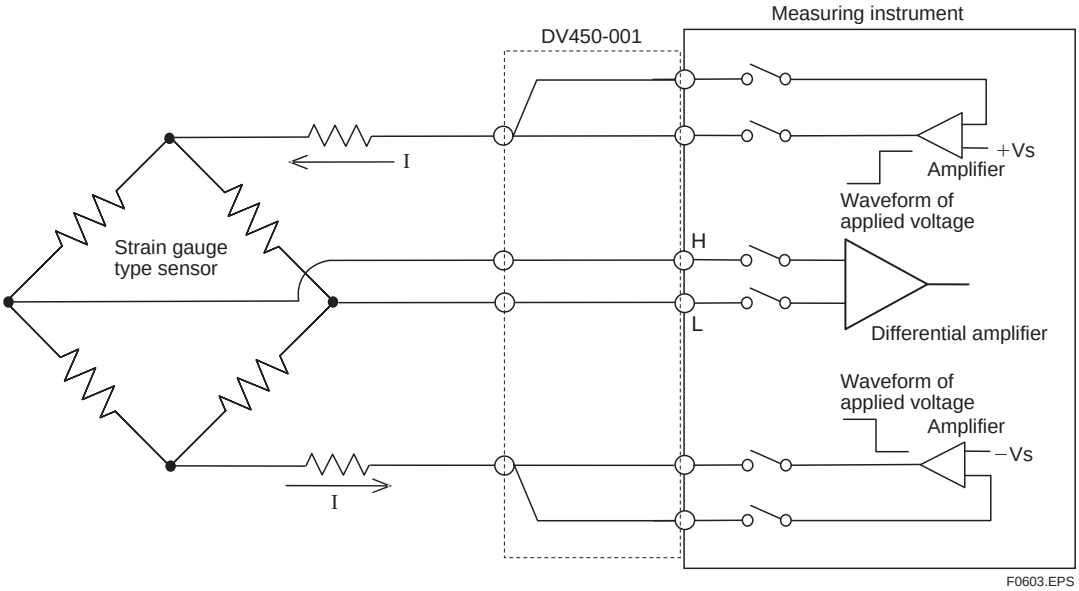


Figure 6-3

As shown in the figure, the DV450-001 shorts the sensing wire and the bridge voltage terminal from within the cable. By using the cable, the issue of delayed bridge voltage response is resolved, and stable measurements become possible. However, it is important to note that remote sensing is not taking place in figure 6-3. In other words, error due to wiring resistance is not compensated for by the measuring scheme in figure 6-3. Care must be taken if the wiring upstream from the DV450-001 is long.

Finally, I would like to give a simple review of remote sensing. For a detailed explanation, please refer to other technical publications or your sensor manufacturer's documentation.

Let us consider a measuring instrument in which voltage is applied to the bridge in a simple manner. In cases like figure 6-4 where wiring resistance cannot be ignored (the wiring is long), the voltage applied to the bridge is reduced by the amount of resistance in the wiring (the voltage applied to the bridge terminals by the measuring instrument is attenuated by the amount of wiring resistance).

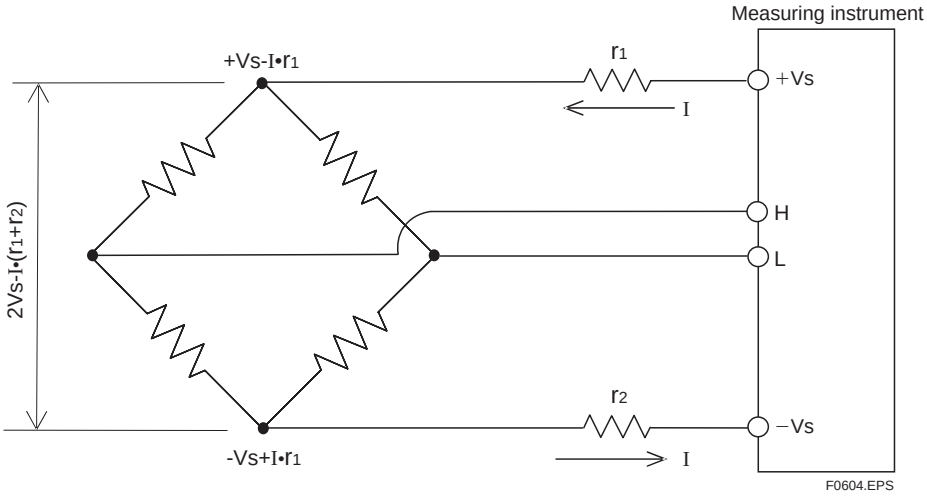


Figure 6-4

To resolve this issue, remote sensing is employed. A sensing wire is added in figure 6-5, and feedback is returned to the amp such that ±Vs is accurately obtained on both bridge terminals (the amp adds voltage equal to the voltage drop caused by the wiring resistance, and outputs the resulting voltage). This enables measurements that are not influenced by wiring resistance.

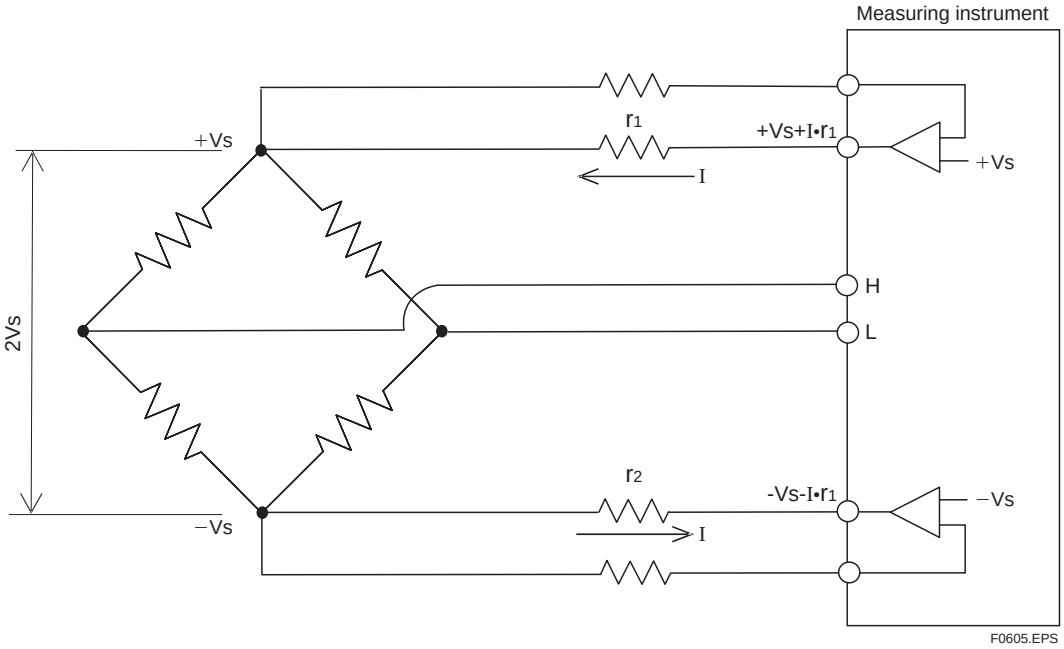


Figure 6-5

7. Computing Functions of Yokogawa-Developed PC Software

The MX100 system depends upon PC software for many of its computing functions. For this reason, proprietary PC software applications (MX100 Standard Software, MXLOGGER) provide a variety of operators. With these software programs, you can also perform most of the computing functions that Yokogawa's DARWIN series contains in its firmware. Moreover, the software has been designed with emphasis on improved convenience, including the capability to write constants directly into computational expressions. On the other hand, since communications intermediate between the MX100 performing the measurement and the PC performing the computation, some applications require certain special techniques to compensate for the lack of synchronization and other issues.

This chapter first introduces some techniques that take into account this intervening communication. The examples that follow thereafter occur frequently in real-world applications, and may be useful for reference. A more detailed explanation is provided at the end of the chapter in section 7.7, "Notes When Writing Computational Expressions and Techniques to Avoid Problems."

After addressing communication issues, key differences are described between the computation functions of Yokogawa's DARWIN and DAQSTATION series instruments. For readers who already have experience with the DARWIN and DAQSTATION series' computation functions, this material may provide a quick way to become familiar with the computation functions of Yokogawa's PC software for the MX100. In sections 7.1 through 7.6, computation examples for a broad range of measurements are provided. It is recommended that you start with those sections if you want to quickly find expressions that correspond to your particular application. The examples provided are as follows:

Section 7.1	Examples: Controlling Operations Using Properties of Measurement Channels
Section 7.2	Examples: Counting Events
Section 7.3	Examples: Using Time
Section 7.4	Examples: Calculating Statistics
Section 7.5	Examples: Operation, Output (Release 2 or Later)
Section 7.6	Miscellaneous

Most of the computational expressions introduced in these examples include techniques that take communication into account between the MX100 main unit and a PC. These techniques can generally be used for any real-world application, but requires sufficient understanding of their underlying principles. Detailed explanations are provided at the end of the chapter in section 7.7, "Notes When Writing Computational Expressions and Techniques to Avoid Problems." Please refer to this information as needed.

• Techniques That Take Communication into Account

When writing expressions using Yokogawa-developed PC software for the MX100, the following points must be considered since communication is involved between the MX100 performing the measurement and the PC performing the computation.

- The referenced result from the measurement channel can be "NaN" (an indefinite or erroneous value).
- The referenced result from the measurement channel of the prech() function is not necessarily the previous value of the ch() function.

When considering these points, the following techniques may be necessary according to your application. For a detailed explanation, see section 7.7, "Notes When Writing Computational Expressions and Techniques to Avoid Problems."

Table 7-1 Techniques (Methods) That Take Communication into Account

No.	Issue	Method	Sample Expression
1	Determination of operation conditions based on measured values	Use the IsNaN() function to guard against occurrence of NaN	IsNaN(ch(<M01>))?{[process when [NaN]]:{original process}
			"ch(<M01>)" →"(IsNaN(ch(<M01>))?{substitute value for [NaN]}: ch(<M01>))"
2	Summing of measured values without the TLOG function	Use the sum() function to avoid NaN	"({items to be summed}+ch(<M01>))" →"sum({items to be summed},ch(<M01>))"
3	Comparison with previous value of measurement channel	Compare after loading into the computation channel	<C91> = IsNaN(ch(<M01>))?prech(<C91>):ch(<M01>)
			<C92> = IsNaN(prech(<C91>))?{do not compare}: {comparison process}

<M01> = Measurement channel number <C91><C92> = computation channel numbers

T0701.EPS

• Comparison with Other Models

Among customers who have purchased the MX100, there are surely those who have used the computation functions of Yokogawa's DARWIN and DAQSTATION series' instruments. Therefore table 7-2 provides a comparison of the computation options for the DARWIN DA100 and DAQSTATION DX series, and the MX100 Standard Software.

Table 7-2 Comparison of the Computation Function with Other Models

Feature	DAQMASTER MX100 Standard Software	DARWIN DA100 Computation Option	DAQSTATION DX Series Computation Option
Absolute timer	None	Up to 6 settings based on abs. timer in format: "every 1 minute to 24 hours, daily at X hrs X min. standard"	Up to 3 settings based on abs. timer in format: "every 1 minute to 24 hours, daily at X hrs 00 min. standard"
Relative timer	Eight settings in units of seconds	Up to 6 settings based on relative timer in the format "every 0-31 days, X hrs. X min."	Up to 3 settings based on relative timer in the format: "every X hrs. X min."
Use of absolute timer/relative timer	Obtained in the expression by the timer() function	Works as event of the "event/action function"	TLOG computation reset interval only
Initializing of value of relative timer	Reset an arbitrary timer or all timers by ResetTimer() function	Works as an action of the "event/action function", all relative time timers reset together	None
Match time	None (instead, can be obtained directly with hourly(), daily() and other functions)	3 settings in the format "every month at 1-31 days, X hrs., X min.," or "daily at X hrs., X min."	1 setting in the format "every month at X day, X hrs/weekly at X day, X hr./daily at X hrs./hourly"
Use of match time	hourly(), daily() and other functions can be used in expressions	Works as event of "event/action function"	Memory time up only
Time series statistical calculations	Time series calculations of Max., Min, P-P, Sum, and Ave of the specified channel per the TLOG function	Time series calculations of Max., Min, P-P, Sum, and Ave of the specified channel per the TLOG function	Time series calculations of Max., Min, P-P, Sum, and Ave of the specified channel per the TLOG function
Initializing TLOG calculation	All TLOG calculations reset collectively per the ResetTlog() function	Arbitrary TLOG calculations reset per the RESET() function	Reset by timer only
Specification of integration units for TLOG integration	None	Same for all TLOG, select OFF, sec, min, hour	Select OFF, Sec, Min, Hour for each computation channel
Channel grouping	None	Assign arbitrarily to 7 groups	None (grouping on screen only)

T0702.EPS

Feature	DAQMASTER MX100 Standard Software	DARWIN DA100 Computation Option	DAQSTATION DX Series Computation Option
Statistical calculation between channels	None (instead, Max., Min, P-P, Sum, and Ave functions available enabling use of arbitrary parameters)	Max., Min, P-P, Sum, and Ave computation from within the specified group by the CLOG function	None
Display hold	None	Display hold for an arbitrary channel specified by the HOLD() function	None
Moving average of measurement channel	None	Set from 2-64 samples on each measurement channel	Set from 2-16 samples on each measurement channel (but only on medium speed model)
Moving average of computation channel (long-duration moving average)	None	None	Set 1 sec-1 hr interval and 1-64 samples on each measurement channel

T0702-2.EPS

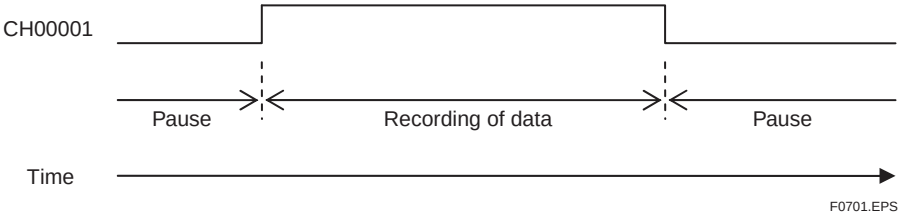
7.1 Examples: Controlling Operation Using Properties of Measurement Channels

With the Yokogawa-developed PC software for the MX100 you can control recording and the insertion of marks based on measured values and alarm statuses. Some examples of this are provided below. However these techniques should be used only with the understanding that the timing of the operation may be delayed or the operation may fail depending on communication errors and other occurrences, causing the operation to not be carried out. In particular, since the MX100 has a powerful communication recovery function, if the instrument recovers after a brief communication failure measured data will be recorded without loss. However, computation during the recovery may be performed with redundant use of measured data from before the failure, so you should take care not to perform computations on measured data that was recovered during a communication failure.

Furthermore, to make the StartRec() and StopRec() functions work (there are several examples of these), the user must set the record start and stop conditions to “Math” in the PC software’s recording/acquisition conditions screen, start recording, and then place recording in wait/pause mode. Also, the range of actually recorded data does not exactly match the timing at which the StartRec(), StopRec() and other event functions are evaluated (executed), but rather includes data slightly before and after (within one second).

• Controlling Recording Using External Contact Input (Level Operation)

This is an example in which a contact input signal is used to start and stop recording on CH00001 (measurement range is set to DI). Recording takes place while the contact on CH00001 is ON, and pauses when it is OFF.



F0701.EPS

Figure 7-1 Controlling Recording Using External Contact Input (Level Operation)

There are many approaches to this, but for example, the following expressions start recording if the value of CH00001 is larger than 0.5, and stops it otherwise.

CH99001 = IsNaN(ch(1))?0:ch(1)>0.5?StartRec():StopRec() (7.1)

or

CH99001 = (IsNaN(ch(1))?0:ch(1))>0.5?StartRec():StopRec() (7.2)

The IsNaN() function is used initially in both expressions 7.1 and 7.2; this is a technique that takes communication into account. It reflects the fact that in expression 7.1 the value of CH00001 can be NaN, and if it is NaN, nothing is done. In expression 7.2, if the value of CH00001 is NaN, it regards this as 0 and carries out the process under that assumption. When comparing these to table 7-1 in the beginning of this chapter "Techniques Taking Communication into Account," you can see that this is an example of one of the techniques introduced.

As an aside, both expressions 7.1 and 7.2 are continuously evaluated as StartRec() being ON while contact is ON and StopRec() being OFF while contact is OFF. In principle, you would need an expression that detects the change from ON to OFF or OFF to ON, but in actuality, if the instrument is already in the state indicated by the function (recording or not recording), that function is not executed. Therefore, it is acceptable to use the expressions in this way.

• Inserting Marks Using External Contact Input

This is an example in which a contact input signal is used on CH00001 to insert marks (measurement range is set to DI). Mark "a" is inserted when the contact on CH00001 is ON.

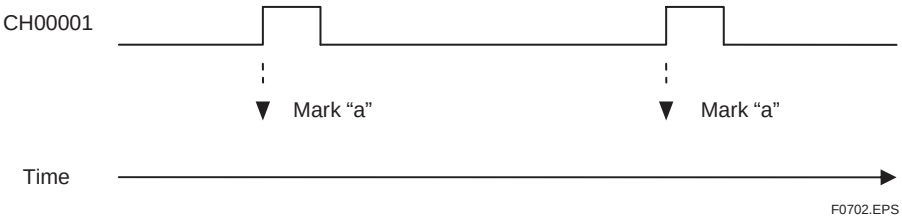


Figure 7-2 Inserting marks using external contact input

There are many approaches to setting this up, but at least we know that we can not derive the expression "CH99001 = ch(1)>0.5?Mark("a"):0 from a strategy of inserting "mark 'a' when the value of CH00001 exceeds 0.5." The thinking behind this strategy is not incorrect, but the expression results in "keep inserting mark 'a' as long as the value of CH00001 exceeds 0.5." Hence, "a" is continuously inserted while the contact is ON. To evaluate "if exceeds," you must detect the change between the previous value and the current value. For example, if you think of it as "insert mark 'a' if the value of CH00001 is larger than the previous value," you can implement the following.

CH99001 = IsNaN(ch(1))?prech(99001):ch(1) (7.3)

CH99002 = IsNaN(prech(99001))?0:prech(99001)<ch(99001)?Mark("a"):0 (7.4)

It may seem confusing, but this includes two techniques that take communication into account. One technique is to compare prech(99001) with ch(99001) after loading into the computation channel since changes are not necessarily detected with the comparison of prech(1) and ch(1). The other technique is the handling of NaN mentioned earlier. Two techniques were used, but if you examine this closely you will see that the methods introduced in table 7-1 at the beginning of this chapter "Techniques Taking Communications into Account" are used as-is.

Furthermore, if you work with expressions 7.3 and 7.4, you can combine them into a single expression.

CH99001 = IsNaN(ch(1))?prech(99001)

:(IsNaN(prech(99001))?0:prech(99001)<ch(1)?Mark("a"):0,ch(1)) (7.5)

• Controlling Recording Using External Contact Input (Edge Operation)

This is an example in which the contact input signal on CH00001/CH00002 is used to start and stop recording (measurement range is set to DI). Recording starts when the contact on CH00001 is ON, and stops when the contact on CH00002 is ON.

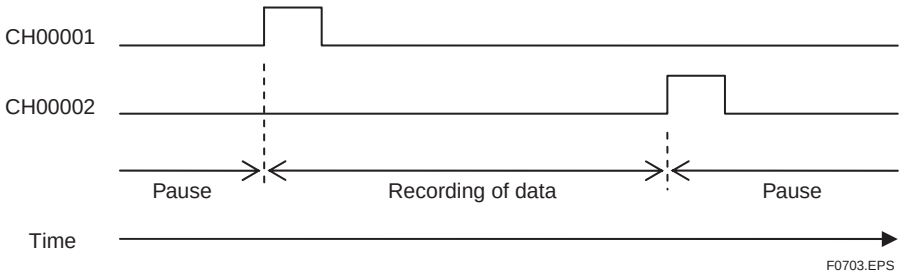


Figure 7-3 Controlling Recording Using Trigger Input (Edge Operation)

You can set this up using the same method introduced above in “Inserting Marks Using External Contact Input.”

If you start with expression 7.5, the following can be derived.

CH99001 = IsNaN(ch(1))?prech(99001)

:(IsNaN(prech(99001))?0:prech(99001)<ch(1)?StartRec():0,ch(1)) (7.6)

CH99002 = IsNaN(ch(2))?prech(99002)

:(IsNaN(prech(99002))?0:prech(99002)<ch(2)?StopRec():0,ch(2)) (7.7)

• Control Recording Using Levels of Measured Values

This is an example in which recording starts when the value of CH00001 exceeds 10.

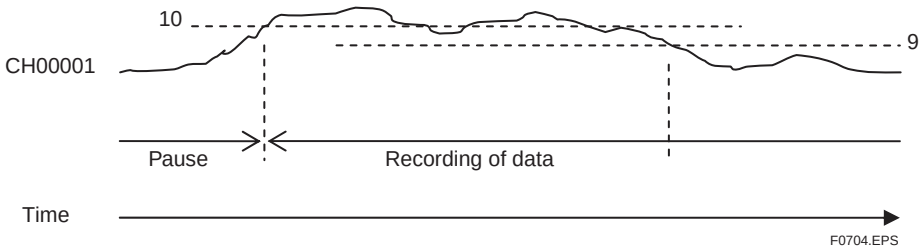


Figure7-4 Controlling Recording Using Levels of Measured Values

You can configure this using the same method introduced above in “Controlling Recording Using External Contact Input (Level Operation).”

Applying the expression in figure 7.1 results in the following.

CH99001 = IsNaN(ch(1))?0:ch(1)>10?StartRec():0 (7.8)

If you also want to stop recording, for example, you can add the StopRec() function to expression 7.8 and a condition judgment, and implement it as follows.

CH99001 = IsNaN(ch(1))?0:(ch(1)>10?StartRec():ch(1)<9?StopRec():0) (7.9)

In this example, recording stops when the measured value of CH00001 falls below 9.

• Inserting Marks When Levels of Measured Values are Exceeded

This is an example in which mark “a” is inserted when the value of CH00001 exceeds 10.

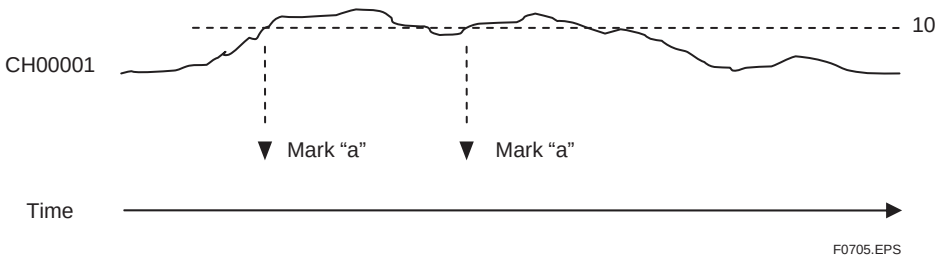


Figure7-5 Inserting marks when the level of the measured value is exceeded

The example is similar to recording control using levels of measured values introduced above, but unlike 7.8 you can apply the same usage as in the above, “Inserting Marks Using External Contact Input” (see the preceding explanation for the reasoning used). Apply expression 7-5 and you can set the condition as “previous reference value is under ten and current reference value is over 10” such that the following results.

```
CH99001 = IsNaN(ch(1))?prech(99001)
        :(IsNaN(prech(99001))?0:(prech(99001)<10)&&(ch(1)>=10)?Mark(“a”):0,ch(1))
```
(7.10)

• Inserting Marks Using the In/Out Condition of Measurement Channel Alarms

In this example, we watch the condition of alarm level 1 on CH00001, and have comments inserted when the alarm is In or Out.

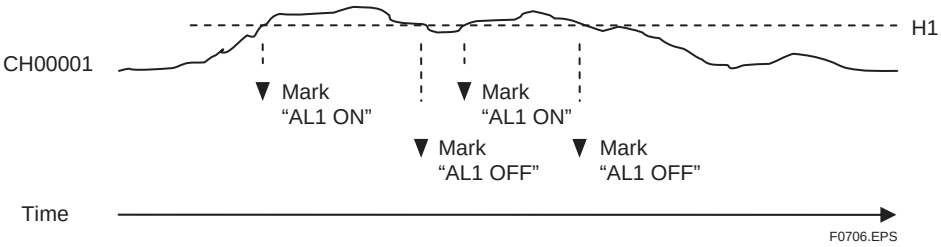


Figure 7-6 Inserting Marks Using the In/Out Condition of Measurement Channel Alarms

You can express this by applying the same method introduced above in “Inserting Marks Using External Contact Input.” You can use the order operator “,” to combine the processes in the same expression when the alarm is In or Out, as follows.

```
CH99001 = IsNaN(prech(99001))?0
        :prech(99001)<alarm(1,1)?Mark(“AL1 ON”)
        :prech(99001)>alarm(1,1)?Mark(“AL1 OFF”):0
        ,alarm(1,1)
```
(7.11)

Basically we apply expression 7.5, but since the alarm() function never returns NaN, a part of the processing by the IsNaN() function is omitted.

7.2 Examples: Counting Events

In section 7.1, “Examples: Controlling Operation Using Properties of Measurement Channels,” examples were given in which measured values and alarm conditions were detected. If these techniques can be used to count events, an even wider range of applications becomes available. Some examples of this are given below. In case of communication error, the counting timing may be delayed or counts may be missed, so please be aware of this before attempting these techniques.

• Counting External Contact Inputs (Edges)/Pulses

In this example CH00001 is used as a pulse signal (with the measuring range set to DI), and the number of rising pulses from the input on CH00001 is counted. You can express this by applying the same method introduced above in section 7.1, “Inserting Marks Using External Contact Input.” Using expressions 7.3 and 7.4 as a base, you can set CH99002 to “increment the value of CH99002 when CH99001 changes from 0 to 1, otherwise retain the current value,” and the following two expressions result.

```
CH99001 = IsNaN(ch(1))?prech(99001):ch(1)
```
(7.12)

```
CH99002 = IsNaN(prech(99001))?ch(99002):prech(99001)<ch(99001)?ch(99002)+1:ch(99002)
```
(7.13)

In addition to count loss that can occur due to communication errors, it is important to note the following.

- Even if the settings for the measurement interval and communication interval of DI match, sample loss can occur since the times are not synchronized, and the pulse width for both High and Low must be three times the computation interval.
- The integral value is reset to zero upon restart of the PC software

Furthermore, in this example, you cannot combine the expressions into a single expression as in 7.5.

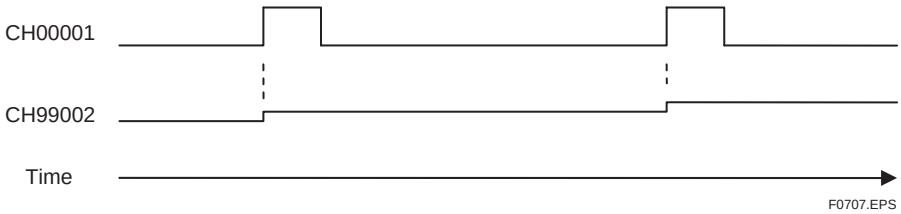


Figure 7-7 Counting External Contact Inputs (Edges)/Pulses

• Counting Measurement Channel Alarms

In this example, we watch the condition of alarm level 1 on CH00001, and count the number of times the alarm is In. This is the same thinking as in section 7.2, “Counting External Contact Inputs (Edges)/Pulses” above, and the following two expressions can be derived.

$$CH99001 = alarm(1,1) \quad (7.14)$$

$$CH99002 = IsNaN(prech(99001))?ch(99002):prech(99001)<ch(99001)?ch(99002)+1:ch(99002) \quad (7.15)$$

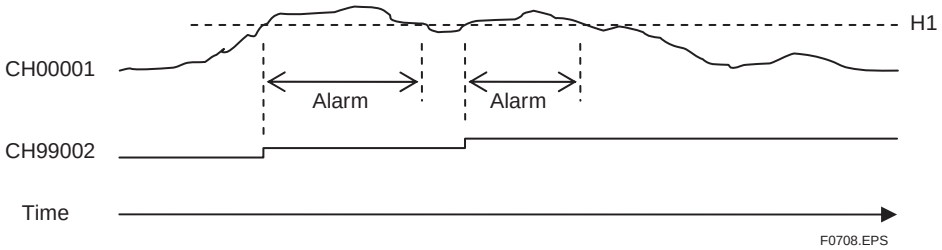


Figure 7-8 Counting Measurement Channel Alarms

7.3 Examples: Using Time

The Yokogawa-developed PC software for the MX100 offer a variety of time functions than are built in to the hardware of DARWIN, DAQSTATION, and the other conventional models. Below are some examples of how these functions are used. Furthermore, to make the StartRec() and StopRec() functions work (there are several examples of these), the user must set the record start and stop conditions to “Math” in the PC software’s acquisition conditions screen, start recording, and then place recording in wait or pause mode. Also, the range of actually recorded data does not exactly match the timing at which the StartRec(), StopRec(), and other event functions are evaluated, but includes data slightly before and after (within one second).

• Recording at Irregular Time Intervals

For example, assume that you want to automatically acquire data at irregular intervals, first for ten minutes from 10:00 to 10:10, next for fifteen minutes from 10:15 to 10:30, and finally for twenty minutes from 10:35 to 10:55. This task can be achieved using time functions and event functions. The computational expressions used in this example are:

$$CH99001 = daily(10,00)?StartRec():0 \quad (7.16)$$

$$CH99002 = daily(10,10)?StopRec():0 \quad (7.17)$$

CH99003 = daily(10,15)?StartRec():0 (7.18)

CH99004 = daily(10,30)?StopRec():0 (7.19)

CH99005 = daily(10,35)?StartRec():0 (7.20)

CH99006 = daily(10,55)?StopRec():0 (7.21)

These can be used as-is, but as explained in section 7.1, “Controlling Recording Using External Contact Input (Level Operation),” considering that it is acceptable to use the StartRec() function and StopRec() function for level operations, if you set it up as “start or stop recording when between 10:00-10:10, 10:15-10:30, or 10:35-10:55,” then you can reduce the number of expressions as follows.

CH99001 = daily(10,0,10,10)||daily(10,15,10,30)||daily(10,35,10,55)?StartRec():StopRec() (7.22)

• Split Files at Specified Times

In such cases as when you want to match up one file’s time range with that of a work shift, there are instances where you want to split continuously recorded data files at specified times. When using DAQWORX MXLOGGER, depending on the logging conditions that are set, you can specify “split files daily at XX hrs XX min XX sec” but this does not allow you to specify multiple times within a single day. And on the MX100 Standard Software, you cannot even make such a specification at all. You can overcome these limitations by using computational expressions. For example, you can split files daily at 7:00, 15:00, and 23:00 with the following expression.

CH99001 = daily(7,0)||daily(15,0)||daily(23,0)?SplitRec():0 (7.23)

• Recording Only during the Daytime from Monday to Friday, and in the Morning on Saturday

Let us assume that you want to record only on weekdays 8:00-17:00 and Saturdays 8:00-12:00. Unfortunately the system cannot identify holidays, so you would need to express this as “If Monday through Friday and 8:00 to 17:00, or Saturday and 8:00 to 12:00...”

CH99001 =
(weekly(1,0,0,6,0,0)&&daily(8,0,17,0))||(weekly(6,0,0,0,0,0)&&daily(8,0,12,0))?StartRec():StopRec()

(7.24)

7.4 Examples: Calculating Statistics

The statistical calculations built in to the hardware of conventional models differ from product to product, but generally time series and between-channel statistics are included. With the Yokogawa-developed PC software for the MX100, the TLOG function is available for implementing time series statistical calculations, but the way of using this differs slightly from the hardware based method. This section introduces several examples that take these differences into account.

• Summation Using the TLOG Function

Let us take a simple and practical example in which an amount of flow (L/min) on CH00001 is summed, and reset every hour on the hour. The differences from the conventional model hardware computation that need to be paid attention to are as follows.

- Since there is no absolute timer, the time function of the computation function is used.
- Since there is no processing function for the units of summation, a constant multiplier is used to account for the relationship to the computation interval.
- Reset is performed collectively for all TLOG computations, so when multiple time series statistical processes of differing reset intervals are required, the TLOG function cannot be used.

The resulting computation is shown below when the computation interval is 100 msec and there is only this one TLOG computation. In expression 7.26, the TLOG computation resets every hour on

the hour. In expression 7.27, after the summation of CH0001 by the tlogsum() function, the units of summation change from “each minute” to “every 100 msec.” CH99002 is the summed value.

CalcInt = 0.1 (7.25)

CH99001 = hourly(0)?ResetTLog():0 (7.26)

CH99002 = tlogsum(1)/60*CalcInt (7.27)

Summation can be performed in the manner above, but from a practical standpoint the “LowCut” line below is generally used to remove the effects of noise around zero of the output from the flow sensor (converter). In this example, the LowCut point is set at 4 L/min. CH99003 is the summed value.

CalcInt = 0.1 (7.28)

LowCut = 4 (7.29)

CH99001 = ch(1)<LowCut?0:ch(1) (7.30)

CH99002 = hourly(0)?ResetTLog():0 (7.31)

CH99003 = tlogsum(99001)/60*CalcInt (7.32)

Furthermore, since the TLOG computation is executed at the interval of the channel under consideration, if that channel’s interval differs, the statistics will be imprecise. In this example, if CH00001’s sampling interval is 10 msec, in expression 7.27 the TLOG computation executed at an interval of 10 msec is referenced at 100 msec, but in expression 7.32 the TLOG computation itself is executed at an interval of 100 msec. Please be aware of this.

• Summation without the TLOG Function

In the example above, “Summation Using the TLOG Function,” we can also perform the calculation without the TLOG function. Since we cannot rely on the ResetTLog() function, we set up the expressions such that they switch between adding to the previous value or adding to 0 depending on the condition of the time function. The CH99002 below is the summed value.

CalcInt = 0.1 (7.33)

LowCut = 4 (7.34)

CH99001 = ch(1)<LowCut?0:ch(1) (7.35)

CH99002 = sum(ch(99003)?0:ch(99002),ch(99001)/60*CalcInt) (7.36)

CH99003 = hourly(0) (7.37)

The use of the arithmetic function sum() in expression 7.36 is an example of a technique taking communication into account as introduced in table 7-1 in the beginning of this chapter. However, in expression 7.35, CH99001 can never be NaN, so actually it is acceptable to use a simple “+” operator for the addition without having to use the sum() function. However, if expression 7.35 is written as “CH99001 = ch(1) >= LowCut?ch(1):0”, even if it looks the same, CH99001 can sometimes be NaN, so problems can occur if you do not use the sum() function. Furthermore, the fact that the summation is written prior to the time function reference is because the reset timing is delayed by one interval relative to the operation of the time function, which means that it is reset immediately after every hour.

• Time Series Averaging without the TLOG Function

When the TLOG function can be used, it is applied in the same manner as in the example above, “Summation Using the TLOG Function.” However if the TLOG function cannot be used, you must write your own “averaging” expression. Specifically, you would calculate the sum and number of the time series data then perform division, but if the referenced value from the measurement channel is NaN, since the sum() function skips this value when performing the summation, the number of statistical data will differ from the number of actual computations, so you must be watchful of this.

An example in which the average temperature on CH00001 is calculated and reset every hour on the hour is worked out below. Furthermore, in expression 7.39, the logic of the IsNaN() function “If CH00001 is NaN, 1, otherwise 0” is reversed using the negation operator “!” (in other words, “if

NaN, 0, otherwise 1"), and by adding that to our expressions the number of time series is calculated. This is a minor technique.

CH99001 = sum(ch(99004)?0:ch(99001),ch(1)) (7.38)

CH99002 = sum(ch(99004)?0:ch(99002),!IsNaN(ch(1))) (7.39)

CH99003 = ch(99001)/ch(99002) (7.40)

CH99004 = hourly(0) (7.41)

7.5 Examples: Operation, Output (Release 2 or Later)

Style number S2 of the MX100 includes an analog output function, and on Yokogawa-developed PC software for the MX100 Release 2 or later, a function has been added that supports the analog function. Also, the ManualDO() and ManualAO() functions have been added for manual operation, improving usability. The following are some examples of how these processes are used.

• Using the PC to Hold Numerical Values

There are times when you want to perform an operation to hold measured or computed values. The simplest example would be one in which you hold the value of CH00001 whenever the contact input on CH00011 is ON; this can even be done with style number S1 of the MX100 and Release 1 of the Yokogawa-developed PC software for the MX100.

CH99001 = !ch(11)?ch(1):ch(99001) (7.42)

This may be a slightly crude expression, but this is based on the reasoning that "if the value of CH00011 is NaN (indefinite) or 0 (OFF), load the value of CH00001, and if 1 (ON), hold the value." This deliberately avoids the use of the IsNaN() function.

What we actually want to introduce here is not expression 7.42 but the technique "while Manual DO 1 on the PC software is ON, ...". To express this we use the ManualDO() function that was added as of Release 2, but if manual DO is not set up in the system the Manual DO button does not appear. Therefore, with the MX100 standard software, this can only be performed when using hardware on which the DO module is actually installed. On the other hand, in the case of DAQWORX MXLOGGER, since you can register a virtual "dummy" unit, if there is room left in the registered number of units, you can use this technique even if an actual DO module is not present.

CH99001 = ManualDO(1)?ch(99001):ch(1) (7.43)

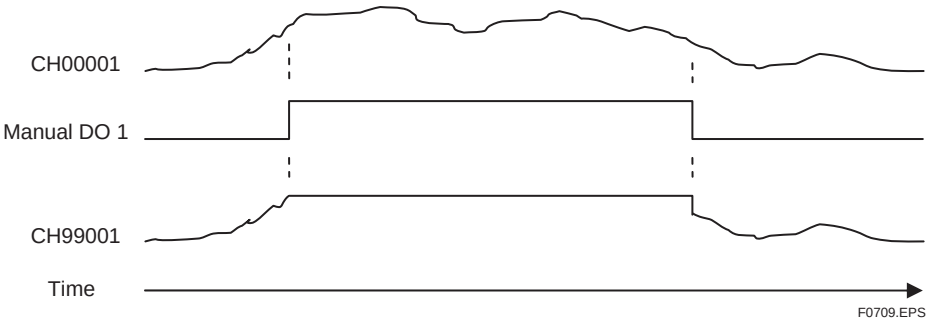


Figure 7-9 Using the PC to Hold Numerical Values

• Emergency Shut Down of Analog Output

In applications where the computed result is output to an AO module, there are cases where you want to be able to manually stop the output (set the output to a certain specified value) in an emergency. In this example as in the above "Using the PC to Hold Numerical Values," you can achieve this using contact input or a Manual DO operation. For example, if the analog output value from CH00021 is calculated on CH99001, you can set Manual DO 1 for emergency shut down (by setting it to 0% for example) in the following manner.

CH00021 : action = transmission output, reference channel = CH9902

CH99002 : span = 0-100

CH99001 = { expression for normal output } (7.44)

CH99002 = ManualDO(1)?0:ch(99001) (7.45)

Also, if we slightly alter expression 7.45, we can assign a preset value for the output values as shown in expression 7.46 below.

CH00021 : action = transmission output, reference channel = CH9902,

output upon error = preset value

CH99002 = ManualDO(1)?NaN:ch(99001) (7.46)

The manual method in expression 7.47 is also possible, in which the output value can be operated manually.

CH99002 = ManualDO(1)?ManualAO(1):ch(99001) (7.47)

Furthermore, for information on the conditions for use of Manual DO button and Manual AO panel, see “Using the PC to Hold Numerical Values,” above.

• Pattern Output of Analog Values

DAQWORX MXLOGGER Release 2 offers a pattern output function for AO modules, and you can output in analog combinations of fixed value output and ramp output (linear output with a positive or negative slope). We definitely recommend DAQWORX MXLOGGER for complex pattern output, but for simple pattern output, you can use the MX100 Standard Software’s computation functions.

For example, the following expressions are used to have the analog output on CH00021 output 0% for the first 5 minutes after computation starts, then step up the output to 25%, 50%, 75%, and 100% every five minutes, returning to 0% after 25 minutes.

CH00021 : operation = transmission output, reference channel = CH9905

CH99003 : span = 0-100

CalcInt = 0.1 (7.48)

CH99001 = ch(99001)+1 (7.49)

CH99002 = ch(99001)*CalcInt (7.50)

CH99003 = 300<=ch(99002)&&ch(99002)<600?25:600<=ch(99002)&&ch(99002)<900?50

:900<=ch(99002)&&ch(99002)<1200?75:0 (7.51)

CH99004 = 1200<=ch(99002)&&ch(99002)<1500?100:0 (7.52)

CH99005 = ch(99003)+ch(99004) (7.53)

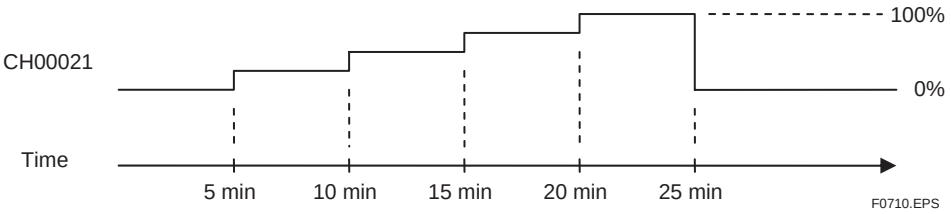


Figure 7-10 Pattern Output of Analog Values

The three computational expressions in 7.51-7.53 are used for calculation of the output patterns; they could not be rendered in a single expression due to the character limit for computational expressions (up to 127 characters). If the pattern is simpler (for example if the pattern stopped after 20 minutes), only one expression need be used.

7.6 Examples: Miscellaneous

• Decoding Eight-Bit DI Input (Numerical Interpretation)

There are cases where numerical values that express the condition of the device are output in binary using multiple contact signals (generally open collector). For example, this would include error numbers that indicate device abnormalities, or pattern/segment numbers occurring in test patterns for testing equipment. By loading these contact signals on the MX and employing the computation function, you can keep a record of the original numerical values. For example, given 8-bit binary values, the basic expression for decoding the signals input to CH00001-CH00008 as "2 when L/L/L/L/L/L/H/L, and 77 when L/H/L/L/H/H/L/H" would be as follows.

CH99001 = poly(2,ch(1),ch(2),ch(3),ch(4),ch(5),ch(6),ch(7),ch(8)) (7.54)

There are plenty of situations where this could be used, but to take it even further, if the measured value is NaN, you can hold the computed result (CH99002 in expressions 7.55 and 7.56), or avoid the bit error (CH99003 in expressions 7.55 through 7.57).

CH99001 = ch(1)+ch(2)+ch(3)+ch(4)+ch(5)+ch(6)+ch(7)+ch(8) (7.55)

CH99002 = IsNaN(ch(99001))?ch(99002):poly(2,ch(1),ch(2),ch(3),ch(4),ch(5),ch(6),ch(7),ch(8)) (7.56)

CH99003 = IsNaN(prech(99002))?ch(99003):ch(99002)==prech(99002)?ch(99002):ch(99003) (7.57)

• Moving Average (Long-Duration Moving Average)

The moving average functions built in to the hardware of the conventional models include a moving average for suppressing noise in the input signal, and a long-duration moving average (in the DAQSTATION series, and others) that reduces fluctuations in the computed results. As for the former, the aim is resolved in general through digital filtering by the hardware, as is the case with the MX100. However, for the latter, the MX100 is not equipped to handle long-duration moving averages (see table 7-2, "Comparison with Other Models" at the beginning of this chapter), so if necessary computational expressions should be used to reduce fluctuations.

Expressions 7.58-7.63 are examples in which the moving average of the computed results from CH99001 is taken every minute for a period of ten minutes (yielding ten data items). As you can see, a large number of computational expressions is required.

Timer 1: Action = edge, interval [sec] = 60, On-period [sec] = 0

CH99001 = { target computation} (7.58)

CH99011 = timer(1)?prech(99012):prech(99011) (7.59)

CH99012 = timer(1)?prech(99013):prech(99012) (7.60)

⋮

CH99019 = timer(1)?prech(99020):prech(99019) (7.61)

CH99020 = timer(1)?ch(99001):prech(99020) (7.62)

CH99021 = ave(ch(99011),ch(99012),...,ch(99019),ch(99020)) (7.63)

• Statistical Calculation between Channels

The Yokogawa-developed PC software for the MX100 does not come with functions for carrying out statistical calculations between channels (see table 7-2 at the beginning of this chapter, "Comparison with Other Models"). However, there are plenty of arithmetic operations available making it easy to create an equivalent set of expressions. Expression 7.64 below calculates the average of CH00001-CH00007, and expression 7.65 calculates the maximum difference between CH00001, CH00011, and CH00021.

CH99001 = ave(ch(1),ch(2),ch(3),ch(4),ch(5),ch(6),ch(7)) (7.64)

CH99002 = pp(ch(1),ch(11),ch(21)) (7.65)

• OR Alarm on Arbitrary Channels

There are times when you want to output alarms using one DO when one of several alarm conditions becomes true. On MXLOGGER, you can specify a range of reference channels for the alarm as a digital output condition, but you are otherwise limited in the selection of channels: for example, you cannot specify multiple non-consecutive channels. And on the MX Standard Software, you cannot even select a range at all. In these cases, you can implement OR alarms with expressions. In this example, we have contact output on CH00031 turn ON when either the condition for alarm level 1 on CH00001 or alarm level 2 on CH00011 is reached.

CH00031 : Action = alarm, reference channel = CH99001, level 1, hold = non-hold

CH99001 : alarm 1 type = High, alarm 1 setting = 0.5, alarm 1 hysteresis = 0

$$\text{CH99001} = \text{alarm}(1,1)||\text{alarm}(11,2) \tag{7.66}$$

However, since this is a result computed by the PC software, the operation occurs more slowly compared to hardware based alarms, and you must be aware that operations can fail during communication abnormalities. Also, if you attempt to hold output using this method, the fact that troubles such as failure in detection of newly occurring alarms can occur means that you need a more accurate technique (not provided herein).

7.7 Notes When Writing Computational Expressions and Techniques to Avoid Problems (Description)

In “Techniques That Take Communication into Account” at the beginning of this chapter, there is a brief explanation of points to consider regarding the communication between the MX100 which is performing measurement and the PC which is performing computation. Techniques for resolving related issues were also touched on briefly. In this section, detailed examples and explanations are provided to help you understand these points and techniques in greater depth.

The table below gives an overview of the points that should be noted regarding the writing of computational expressions.

Table 7-3 Points to Note When Writing Computational Expressions

No.	Issue	Cause	Key Points	Method
1	It is possible for the referenced result from the measurement channel to be NaN.	Communication broken, or data not received immediately after monitoring starts	Determination of operation conditions based on measured values, measured value summing without TLOG function, other	Use the IsNaN() or sum() function
2	The referenced result from the measurement channel of the prech() function is not necessarily the previous value of the ch() function.	Measured data via communications received too late	Comparison with previous value of measurement channel (detection of changes in contact or other signals), other	Load into the computation channel first

T0703.EPS

To understand these points better, we will first briefly explain how to incorporate measured values into computational expressions, and the conditions under which referenced results from the measurement channel can be NaN, and then give specific examples and techniques.

• Incorporating Measured Values and Other Parameters into Computational Expressions

The ch() and prech() functions are available as a means of incorporating measured values into computational expressions. However, unlike computational expressions which are executed on the PC at fixed intervals, measured values are not necessarily obtained at fixed intervals. This is because communication takes place between the MX main unit and PC, and the measurements on the MX and computations on the PC are not synchronized. Therefore, you must be aware that the time series of the data measured on the MX main unit does not necessarily match the time series incorporated into the computational expression by the ch() function. Figure 7-11 and Table 7-4 show how the ch() and prech() functions are affected depending on the delay of arrival of measured data.

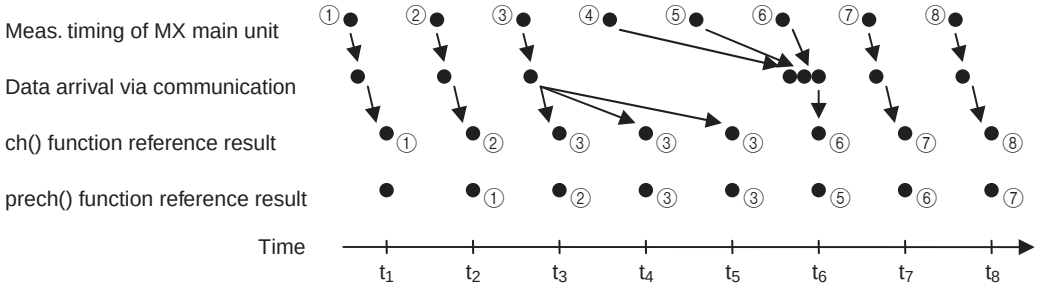


Figure 7.11 Effect on the ch() and prec() functions by the delay of measured data

Table 7.4 Effect on the ch() and prec() functions by the delay of measured data (explanation)

Time	Arrived data	Reference of ch() function	Reference of prech() function
t_2	①②	② nearest time t_2 is referenced	① nearest time t_1 is referenced
t_3	①②③	③ nearest time t_3 is referenced	② nearest time t_2 is referenced
t_4	①②③	③ nearest time t_4 is referenced (④ has not arrived)	③ nearest time t_3 is referenced
t_5	①②③	③ nearest time t_5 is referenced (④ and ⑤ have not arrived)	③ nearest time t_4 is referenced (④ has not arrived)
t_6	①②③④⑤⑥	⑥ nearest time t_6 is referenced	⑤ nearest time t_5 is referenced
t_7	①②③④⑤⑥⑦	⑦ nearest time t_7 is referenced	⑥ nearest time t_6 is referenced

T0704.EPS

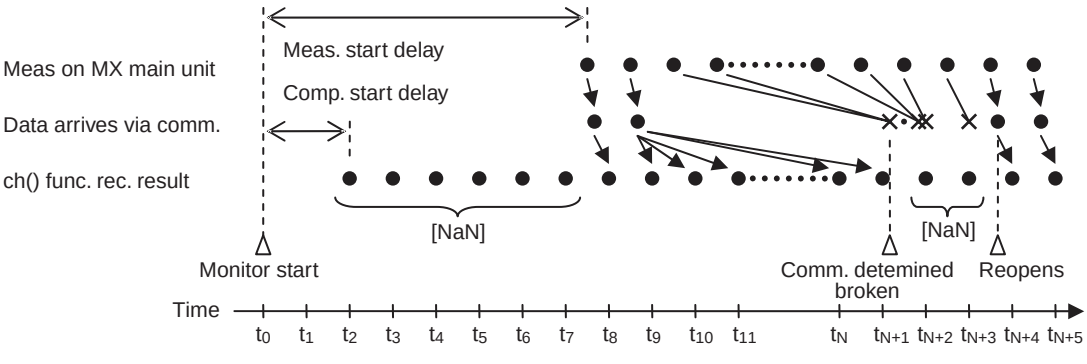
As table 7-4 shows, the ch() function references the data nearest the time the computation is executed, and the prech() function references the data nearest the previous execution of the computation. Nevertheless, since the timing at which data arrives varies depending on the conditions of the communication, the referenced result from the measurement channel of the prech() function is not necessarily the previous value of the ch() function. This does not pose a problem in every case, but when you need to accurately detect changes in measured values, special measures are required. The basic idea underlying such measures is that:

- Measured values are first loaded on computation channels, and then changes in the computation channel are detected. A specific example is given below in “Example 3) Insert mark ‘a’ when the contact on CH00001 is ON.”

• Conditions in Which the Referenced Result from the Measurement Channel Is NaN

When measured data that must be referenced by the ch() and prech() functions is absent, NaN (an indefinite value, or error value) is substituted. This occurs of course when communications with the PC software are cut, but actually also occurs directly after monitoring starts-since measurement start is late by comparison with computation start, remember that computation takes place when there is no measured data and NaN is used instead.

Figure 7-12 and Table 7-5 show how referencing by the ch() and prech() functions is affected immediately after start of monitoring and by a broken communication.



F0712.EPS

Figure 7.12 Referencing NaN immediately after monitoring starts and when communication is broken

Table 7.5 Referencing NaN immediately after monitoring starts and when communication is broken (explanation)

Time	Condition of Referencing of Measured Data in Computations
t_0 to t_1	Computation fails to start immediately after monitoring starts
t_2 to t_8	Computation is started, but NaN is used since no measured data has been received yet
t_9 to t_{10}	Measurement is also started, and the received measured data is referenced
t_{11} to t_{N+1}	Measured data does not arrive, and the measured data nearest t_{10} is referenced
t_{N+2} to t_{N+3}	Communication is determined to be broken, and NaN is used since no measured data has been received yet
t_{N+4} to t_{N+5}	Communication is restarted, and the received measured data is referenced

T0705.EPS

As figure 7-5 shows, NaN is the referenced result from the measurement channel in the following cases.

- When a break in communications is detected
- When immediately after monitoring starts no measured data exists

The measured value becomes NaN, and for the most part, the computed result is also NaN. This does not pose a problem in every case, but if your application makes it prohibitive to have a computed result of NaN, special measures are required. Two specific examples are given below for a detailed explanation.

• Example 1) Recording Starts When the Value on CH00001 Exceeds 10

To prevent problems when NaN is the referenced value of the measurement channel, we use the example given in the previous section, “Controlling Recording Using Levels of Measured Values,” and show the problematic computational expressions and their fixes in table 7-6.

Table 7-6 Fixes for NaN Being the Referenced Result from the Measurement Channel

Recording starts when the value on CH00001 exceeds 10		
Expression before fix (problematic)	CH99001 = ch(1)>10?StartRec():0	(7.67)
Expression 1 after fix	CH99001 = (IsNaN(ch(1))?0:ch(1))>10?StartRec():0	(7.68)
Expression 2 after fix	CH99001 = IsNaN(ch(1))?0:ch(1)>10?StartRec():0	(7.8)

T0706.EPS

If you use the sentence “Recording starts when the value of CH00001 exceeds 10” as-is, the conditional operator “?” is used and the expression looks like the one in 7.67. However, in this expression, there are practical problems. Namely, the value of ch(1) can be NaN (see “Conditions in Which the Referenced Result from the Measurement Channel is NaN.”) When the value of ch(1) is NaN, the result of the comparison operation “ch(1)>10” is also NaN. Logical and conditional operators only handle 0 as “false,” and all other values including NaN are considered “True.”

Therefore, given the syntax before any special measures are taken, recording is also started when ch(1) is NaN.

There are two methods for resolving this issue.

1. When the value of ch(1) is NaN, overwrite ch(1) with a substitute value and compare it to 10.
2. When the value of ch(1) is NaN, do not carry out the comparison computation.

The fixes suggested in table 7-6 are based on this reasoning. With method 1, in the problematic expression in 7.67, if we overwrite ch(1) with IsNaN(ch(1))?0:ch(1), we derive expression 7.68 With method 2, in the problematic expression in 7.67, if we rewrite the entire expression with IsNaN(ch(1))?0:{original expression}, we derive expression 7.8.

Table 7-7 shows the flow of computational expression evaluation as to how CH99001 is calculated when the reference value of CH00001 is NaN, giving an overview of expressions before and after fixes based on methods (1) and (2). With the computational expression before the fix, recording starts with the StartRec() function being evaluated, but in the computational expression after the fix, you can see that evaluation of StartRec() is deferred.

Table 7-7 Fixes for NaN Being the Referenced Result from the Measurement Channel (Flow of Evaluation of Computational Expression)

Expression before fix	Expression 1 after fix	Expression 2 after fix
CH99001	CH99001	CH99001
= ch(1)>10?StartRec():0 = ***[NaN]*** >10?StartRec():0	= (IsNaN(ch(1))?0:ch(1))>10?StartRec():0 = (IsNaN(***[NaN]***)?0:ch(1))>10?StartRec():0	= IsNaN(ch(1))?0:ch(1)>10?StartRec():0 = IsNaN(***[NaN]***)?0:ch(1)>10?StartRec():0
= [NaN]>10?StartRec():0 = ***[NaN]*** ?StartRec():0	= (IsNaN([NaN])?0:ch(1))>10?StartRec():0 = (***1*** ?0:ch(1))>10?StartRec():0	= IsNaN([NaN])?0:ch(1)>10?StartRec():0 = ***1*** ?0:ch(1)>10?StartRec():0
= [NaN]?StartRec():0 = ***StartRec()***	= (1?0:ch(1))>10?StartRec():0 = ***0*** >10?StartRec():0	= 1?0:ch(1)>10?StartRec():0 = ***0***
= StartRec() = ***1***	= 0>10?StartRec():0 = ***0*** ?StartRec():0	
	= 0?StartRec():0 = ***0***	

* Double-underlined items are evaluated, and the bolded italic items are for substitution.

T0707.EPS

• Example 2) Summation of Values on CH00001

Table 7-8 shows one more example of a fix used when NaN is the referenced result from the measurement channel.

Table 7-8 Fix When NaN is the Referenced Result from the Measurement Channel

Summing the Values on CH00001		
Expression before fix (problematic)	CH99001 = ch(99001)+ch(1)	(7.69)
Expression after fix	CH99001 = sum(ch(99001),ch(1))	(7.70)

T0708.EPS

This example generally uses the tlogsum() function, but depending on the application the TLOG function may not be appropriate (see section 7.4). If you do not use the tlogsum() function, the expression will be rendered as “add values of CH00001 on CH99001,” but in this expression the arithmetical operator “+” is used to derive an expression like the one in 7.69. However, this expression has the limitation that, again, the value of ch(1) can be NaN (see “Conditions in Which the Referenced Result from the Measurement Channel is NaN.”) When the value of ch(1) is NaN, the result of the arithmetic operation “ch(99001)+ch(1)” is also NaN (displayed as “INVALID”). Once the value of ch(99001) is NaN, the results of the arithmetic operation “CH99001 = ch(99001)+ch(1)” will always be NaN.

To resolve this issue, you can use the IsNaN() function as demonstrated in example 1 above, “Recording Starts When the Value of CH00001 Exceeds 10,” but if you take advantage of the fact that the sum() function ignores NaN, you can solve the problem easily as shown in the table 7-8. Table 7-9 shows the flow of computational expression evaluation for expressions before and after the fix, as to how CH99001 is calculated when the reference value of CH00001 is 2.7, NaN, and 3.2. Before the fix, even if the value of CH00001 returns from NaN the value of CH99001 remains at NaN, but you can see that after the fix, summation continues without the value ever becoming NaN.

Table 7-9 Fix for NaN being the Referenced Result from the Measurement Channel, part 2 (flow of evaluation of computational expression)

Time	Expression before fix	Expression after fix
t_{N-1}	CH99001 = ch(99001)+ch(1) = ***0 +2.7*** = 0+2.7 = ***2.7***	CH99001 = sum(ch(99001),ch(1)) = sum(***0 ,2.7***) = sum(0,2.7) = ***2.7***
t_N	CH99001 = ch(99001)+ch(1) = ***2.7 +[NaN]*** = 2.7+[NaN] = ***[NaN]***	CH99001 = sum(ch(99001),ch(1)) = sum(***2.7 ,[NaN]***) = sum(2.7,[NaN]) = ***2.7***
t_{N+1}	CH99001 = ch(99001)+ch(1) = ***[NaN] +3.2*** = [NaN]+3.2 = ***[NaN]***	CH99001 = sum(ch(99001),ch(1)) = sum(***2.7 ,3.2***) = sum(2.7,3.2) = ***5.9***

* Double-underlined items are evaluated, and the bolded italic items are for substitution.

T0709.EPS

As an aside, the initial value of the computation channel is 0. Therefore, with the expression “CH99001 = ch(99001)+.....” and “CH99001 = sum(ch(99001),.....)”, this is the summation on and after recording, or on and after monitoring.

• Example 3) Insert Mark “a” When Contact on CH00001 Is ON

As a fix for the referenced result of the prech() function not necessarily being the previous value of the ch() function, we use the example given in the previous section, “Inserting Marks Using External Contact Input,” and show the problematic computational expressions and their fixes in table 7-10.

Table 7-10 Fixes preventing the referenced result for the prech() function not necessarily being the previous value of the ch() function.

Insert mark “a” when the contact on CH00001 is ON		
Expression before fix (problematic)	CH99001 = IsNaN(prech(1)+ch(1))?0:prech(1)<ch(1)?Mark(“a”):0	(7.71)
Expression 1 after fix	CH99001 = IsNaN(ch(1))?prech(99001):ch(1)	(7.3)
	CH99002 = IsNaN(prech(99001))?0:prech(99001)<ch(99001)?Mark("a"):0	(7.4)
Expression 2 after fix	CH99001 = IsNaN(ch(1))?prech(99001) :(IsNaN(prech(99001))?0:prech(99001)<ch(1)?Mark(“a”):0,ch(1))	(7.5)

T0710.EPS

In the above explanation, we think of it as “insert mark ‘a’ if the value of CH00001 is larger than the previous value.” Converting that sentence as-is into a computational expression, if you use the “NaN fix,” you can derive the expression as the one in 7.71. However, in this expression, even though the value on CH00001 changes from 0 to 1, since the measurement on the MX main unit is not synchronized with the execution of computation on the PC software, the conditions prech(1)=0 and ch(1)=1 do not necessarily arise (see above, “Incorporating Measured Values into Computational Expressions”). If prech(1)=0 and ch(1)=0 when certain computations are executed, then if prech(1)=1 and ch(1)=1 upon execution of the next computation, the change in value from 0 to 1 on CH00001 will not be detected.

We have already explained one of the fundamentals of solving this problem:

- Measured values are first loaded on computation channels, then changes in the computation channel are detected

However, it is not simply a matter of loading. If you consider the types of transitions that must be detected as shown in table 7-11, the following sort of loading method is required:

- Only the NaN immediately after measurement start is handled as NaN
- Once it becomes a real number, any NaN appearing thereafter is ignored and the previous value is held.

Table 7-11 Types of transitions that must be detected

ch(1) transitions	Item must be detected?	ch(1) transitions	Item must be detected?
Meas. start -> NaN -> 0	No	Meas start → [NaN] → 1	No
0 → 0	No	1 → 1	No
1 → 0	No	0 → 1	Yes
0 → [NaN] → 0	No	1 → [NaN] → 1	No
1 → [NaN] → 0	No	0 → [NaN] → 1	Yes

T0711.EPS

The fixes suggested in table 7-10 are based on this thinking. First, expression 7.3 is used to load the value of CH0001 to CH99001. The reference value of the prech() function for the computation channel right after monitoring start always is NaN, so this expression will definitely accomplish its loading task. With problematic expression 7.71, if we overwrite prech(1) and ch(1) with prech(99001) and ch(99001) and remove ch(99001) as the target of the IsNaN() function, we derive expression 7.4. Expression 2 (after fix) is exactly as explained above.

Regarding the expressions before and after the fix, in figure 7-11 above in the section entitled, “Incorporating Measured Values into Computational Expressions,” between ③ and ④ in which the value changed from 0 to 1, an example was given. The flow of computational evaluation for this is shown in table 7-12 below (however, evaluation of the IsNaN() function is omitted). In the expression before the fix, from time t_5 to t_6, prech(1) and ch(1) change from 0 to 1 at the same time, and change in the contact could not be detected, but in the expressions after being fixed, you can see that the change is clearly detected.

Table 7-12 Fixes for the referenced result from the measurement channel of the prech() function not necessarily being the previous value of the ch() function (flow of evaluation of expression)

Time	Expression before fix	Expression 1 after fix	Expression 2 after fix
t_5	CH99001 = ... = prech(1)<ch(1)?Mark(“a”):0 = ***0*** <***0*** ?Mark(“a”):0 = ***0***	CH99001 = ch(1) = ***0*** CH99002 = ... = prech(99001)<ch(99001)? Mark(“a”):0 = ***0*** <***0*** ?Mark(“a”):0 = ***0***	CH99001 =... = prech(99001)<ch(1)? Mark(“a”):0,ch(1) = ***0*** <***0*** ?Mark(“a”):0,ch(1) = ***0*** , ch(1) = ch(1) = ***0***
t_6	CH99001 = ... = prech(1)<ch(1)?Mark(“a”):0 = ***1*** <***1*** ?Mark(“a”):0 = ***0***	CH99001 = ch(1) = ***1*** CH99002 = ... = prech(99001)<ch(99001)? Mark(“a”):0 = ***0*** <***1*** ?Mark(“a”):0 = ***Mark(“a”)*** = ***1***	CH99001 =... = prech(99001)<ch(1)? Mark(“a”):0,ch(1) = ***0*** <***1*** ?Mark(“a”):0,ch(1) = ***Mark(“a”)*** ,ch(1) = ***1*** ,ch(1) = ch(1) = ***1***
t_7	CH99001 = ... = prech(1)<ch(1)?Mark(“a”):0 = ***1*** <***1*** ?Mark(“a”):0 = ***0***	CH99001 = ch(1) = ***1*** CH99002 = ... = prech(99001)<ch(99001)? Mark(“a”):0 = ***1*** <***1*** ?Mark(“a”):0 = ***0***	CH99001 =... = prech(99001)<ch(1)? Mark(“a”):0,ch(1) = ***1*** <***1*** ?Mark(“a”):0,ch(1) = ***0*** ,ch(1) = ch(1) = ***1***

* Double-underlined items are evaluated, and the bolded italic items are for substitution.

T0712.EPS

8. Comparison between Functions of MX100 and DARWIN DA100 Data Acquisition Units

	MX100	Superiority	DA100	Remarks
System configuration	• PC • Base plates • Control units • I/O modules	>	• PC • Basic/extension units • Sub-units • I/O modules • Communication modules	The MX100 permits simpler system configuration, covering from a small to large number of channels.
Module lineup	10-ms AI, 100-ms AI, DI, 24V DI, Strain, 4-Wire RTD, Resistance, Digital output, Analog output and PWM output	≈	10/20/30-channel universal (mV/TC), power monitor, strain, DI, mA, pulse, and analog output communication modules	Both support a wider choice of modules.
Measurement interval	• 10 ms at the shortest • Multi-interval (combined use of three types of interval is allowed)	>>	• 500 ms at the shortest Synchronized with the interval of the slowest module	The MX100 could be said to be better in performance for its high-speed and multi-interval measurement capabilities.
Withstand voltage	3700 Vrms (60/60 Hz), for one minute, between input and case	>>	1500 Vrms (60/60 Hz), for one minute, between input and case	The MX100 can withstand higher voltages.
Number of input channels	60 channels max./unit 1200 channels per system (when MXLOGGER is used)	≈	60 channels max./sub-unit 300 channels max.; the DAQLOGGER is required to add more channels	Both support multi-channel measurement.
Alarm processing	Shared between the main unit and PC software. Main unit: 2 levels/channel - upper/lower limit and differential upper/lower limit alarms PC software: "calculation plus command DO" alarm and rate-of-change alarm	<	Entirely undertaken by the main unit only. Four levels/channel - upper/lower limit, differential upper/lower limit, and rate-of-change alarms	The DARWIN is superior in alarm processing based on the main unit.
Computing functions	Shared between the main unit and PC software. Main unit: differential calculation, scaling and software filtering PC software: Provided with computing functions as standard feature; computational functions equivalent to the DARWIN's M1 option are available from this software	>	Entirely undertaken by the main unit only. - Differential calculation, scaling and moving-average processing	The MX100 is superior in computing flexibility.
Report functions	None. Requires use of, for example, Excel macros.	<	Supported by the /M3 option	The DA100 is superior in reporting.
Communication functions	Ethernet only. Simplified initial setup by means of automatic detection, automatic IP address setting, etc.	≈	Ethernet, GP-IB, RS-232C and RS-422/485	The MX100 is superior in the convenience of use, though the DA100 provides a wider choice of communication means.
CF card	• Provided with data backup function in case of communication shutdown • /DS option	>	None	The MX100 is superior in reliability

F0801.EPS