

MX100仕様解説

TI 04M08B01-00 2版

目 次

はじめに.....	3
1. A/D分解能について	4
2. マイナス温度測定時の基準接点補償確度について	5
3. MXの耐圧（絶縁）性能について	6
4. MXのノイズ除去性能について	8
5. MXの測定値時刻について	17
6. ひずみ変換ケーブル（DV450-001）の使用について	27
7. 横河純正PCソフトウェアによる演算機能について	31
7.1 事例：測定チャンネルの状態による動作制御	
7.2 事例：イベントを数える	
7.3 事例：時間の使い方	
7.4 事例：統計演算	
7.5 事例：操作・出力関係（Release 2以降）	
7.6 事例：その他	
7.7 演算構築上の注意点および回避テクニック（解説）	
8. MX100とDARWINの機能比較	51

はじめに

本TIでは、MX仕様の内、分かりにくい仕様、重要な仕様を選び解説をしています。テーマ選定には、弊社DARWINシリーズで繰り返しお問い合わせいただいた内容も参考にしております。お客様がMXをご使用になられる際、ふっと湧く疑問の解決に本TIがお役に立てればと考えております。

今後、MXシリーズの機能拡張に合わせ、また、お客様の声に合わせて内容は順次更新してまいります。本TIへのご意見、ご要望は、横河電機(株)IA事業部 プロダクト事業センター NetSoL PMK部までお寄せください。

お問い合わせ先

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

TEL : 055-243-0309

FAX : 055-243-0397

1. A / D分解能について

高速ユニバーサル入力モジュール (MX110-UNV-H04) と
中速ユニバーサル入力モジュール (MX110-UNV-M10) の
A/D分解能仕様は、16ビット ($\pm 20000/\pm 6000$) とあります。
この意味について簡単に説明します。

16ビット = $2^{16} = 65536$ ですから、測定値は、測定レンジに対し1/65536の分解能を持つはずだと思われるかもしれませんが、しかし、MXでは、人間の理解しやすい最高分解能にするために、 ± 20000 つまり測定レンジに対し1/40000、もしくは ± 6000 つまり測定レンジに対し1/12000の分解能で測定値を表示しています。 ± 20000 で測定するか ± 6000 で測定するかは、レンジによってきりの良い方を選択しています。

例えば、2 V レンジ (- 2 V ~ 2 V) は、最高分解能 $100 \mu V (0.1 mV) = 4 V \div 40000$
6 Vレンジ (- 6 V ~ 6 V) は、最高分解能 $1 mV = 12 V \div 12000$
となっています。

ところで、MXのレンジの中で2レンジだけ1/60000で測定しているレンジがあります。仕様の特種入力レンジにある60 mVレンジ (0 ~ 60 mV) と6 Vレンジ (0 ~ 6 V) がそれぞれにあたります。これは、入力をプラス側の入力レンジに絞ることで最高分解能を標準60 mV、6 Vレンジから1桁あげています。ちなみに、特種入力レンジ 6 V (0 ~ 6 V) の最高分解能は、 $100 \mu V (0.1 mV) = 6V \div 60000$ となります。
なお、横河純正ソフトウェアでは、この特種入力レンジは、別売ソフトウェア (MXLOGGER) でのみ使用可能です。

2. マイナス温度測定時の基準接点補償確度について

基準接点補償確度の仕様欄には，“0 以上測定時”という但し書きがあり ± 1 ， ± 0.5 といった確度値が示されています。では，マイナス温度測定時での基準接点補償確度はどうなるのでしょうか？

マイナス温度測定時は，基準接点補償確度値は大きく（悪く）なってしまいます。（MX機器本体がマイナス温度にあるのではなく，測温対象温度がマイナス温度にある状態を述べています。MX本体の使用温度範囲は0～50 です。）

MXでは，トランジスタを基準接点補償器としているのですが，このトランジスタは，測定器側端子温度に応じて電圧を発生させます。この電圧と実測定中の熱電対からの電圧（熱起電力）とで，基準接点が0 の場合の電圧（熱起電力）を算出し，それを使用熱電対の熱起電力曲線に照らし合わせ温度として出力します。

では，何故，マイナス側では，基準接点補償確度値は大きく（悪く）になってしまうのでしょうか？

それは，熱起電力曲線と言うように，熱電対の温度と電圧の関係は，傾きが一定ではないことによります。（専門書を参照下さい）一般的に熱電対は，プラス側に比べマイナス側の温度では，温度変化に対して熱起電力変化が小さくなります。ところで，MXでは，電圧を熱起電力表に照らして温度表示すると説明しました。したがって，トランジスタの出力電圧誤差を温度換算した時，マイナス温度測定時の誤差は，プラス温度測定時の誤差より大きくなってしまいます。

（ちなみに，0～50 でのトランジスタ出力電圧の温度変動は，ほとんどありません。）

以下に温度別の基準接点補償確度の一例を示します。

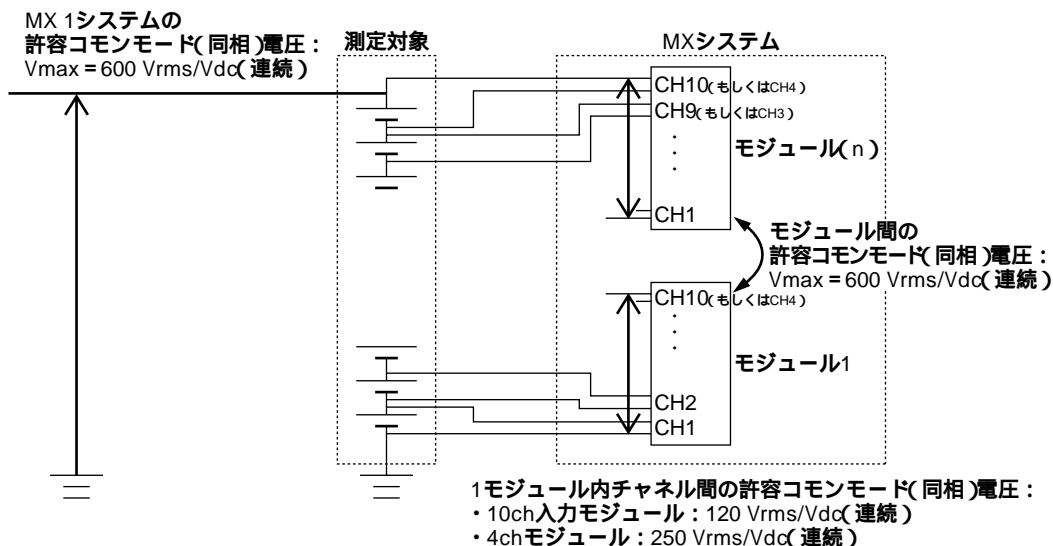
測定対象温度 \ 熱電対タイプ	K	E	J	T	L	U
23°C	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$
0°C	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$
- 50°C	$\pm 0.6^{\circ}\text{C}$	$\pm 0.6^{\circ}\text{C}$	$\pm 0.6^{\circ}\text{C}$	$\pm 0.6^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$	$\pm 0.5^{\circ}\text{C}$
- 100°C	$\pm 0.7^{\circ}\text{C}$	$\pm 0.7^{\circ}\text{C}$	$\pm 0.7^{\circ}\text{C}$	$\pm 0.7^{\circ}\text{C}$	$\pm 0.7^{\circ}\text{C}$	$\pm 0.7^{\circ}\text{C}$
- 150°C	$\pm 0.9^{\circ}\text{C}$	$\pm 0.9^{\circ}\text{C}$	$\pm 0.8^{\circ}\text{C}$	$\pm 0.9^{\circ}\text{C}$	$\pm 0.7^{\circ}\text{C}$	$\pm 1.0^{\circ}\text{C}$
- 200°C	$\pm 1.3^{\circ}\text{C}$	$\pm 1.2^{\circ}\text{C}$	$\pm 1.2^{\circ}\text{C}$	$\pm 1.3^{\circ}\text{C}$	$\pm 0.9^{\circ}\text{C}$	$\pm 1.0^{\circ}\text{C}$

T0201.EPS

3. MXの耐圧（絶縁）性能について

MXは高耐圧が大きな特徴となっています。強化（2重）絶縁を施していますので、カタログにもあるような値が連続でかかっても、安全に使用することができます。

カタログにある図を再度示します。



F0301.EPS

図3-1

図3-1は、図3-2の様にまとめることができます。

V1, V2, V3をそれぞれ以下の位置にかかる許容共通モード（同相）電圧とすると、

- V1：入力 - ケース間
- V2：同一モジュール内のch間
- V3：異なるモジュールのch間

図3-1で示した各許容共通モード（同相）電圧は、

- V1= MX 1システムの許容共通モード（同相）電圧
- V2= 1モジュール内チャンネル間の許容共通モード（同相）電圧
- V3=モジュール間の許容共通モード（同相）電圧

となります。

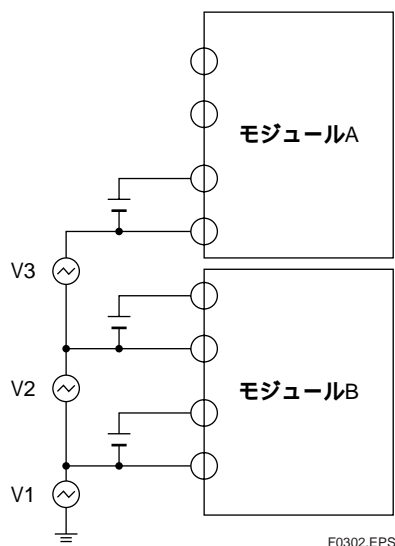


図3-2

図3-2に照らしてMXの各許容電圧とDARWINでの値を一覧にしてみます。

		V1 : 入力 - ケース間 V3 : 異なるモジュールのch間	V2 : ch間
MX110-UNV -H04	許容共通モード(同相)電圧	600 Vrms/Vdc (連続)	250 Vrms/Vdc (連続)
	試験電圧 (1 min)	3700 Vrms (50/60Hz)	2300 Vrms (50/60 Hz)
MX110-UNV -M10	許容共通モード(同相)電圧	600 Vrms/Vdc (連続)	120 Vrms/Vdc (連続)
	試験電圧 (1 min)	3700 Vrms (50/60Hz)	1000 Vrms (50/60 Hz)
DARWIN	許容共通モード(同相)電圧	250 Vac ノイズ電圧	150 Vac ノイズ電圧
	試験電圧 (1 min)	1500 Vrms (50/60Hz)	1000 Vrms (50/60 Hz)

F0303.EPS

図3-3

DARWINの仕様では、許容共通モード(同相)電圧をノイズ電圧で表しています。これは、DARWINが測定電圧レンジが低い“SELV(Safety Extra Low Voltage)”を対象とする測定器として位置付けられていることによります。“SELV”を対象とする測定器では、連続許容電圧(2重または強化絶縁を前提としています)までうたう必要がなく、こういった表記を採用しています。SELV(Safety Extra Low Voltage)とは、IEC1010の規定にしたがう安全電圧のことを言います。

ちなみに、DARWINの許容同相電圧をMXと同じ強化(2重)絶縁を前提とした記述にしますと、250 Vacノイズ電圧は160 Vrms/Vdc(連続)、150 Vacノイズ電圧は120 Vrms/Vdc(連続)となります。

なお、参考までにSELVの定義を示します。(IEC1010 6.3項)

電圧：電圧レベルは30 V r.m.s, 42.4Vピーク又は60 V d.c.以下である。

電流：電圧が、上の値を超えていれば、正弦波形に対しては、0.5 mA r.m.s 非正弦波形又は混合周波数に対しては、0.7mA ピーク

他に、キャパシタンスにも制限があります。

DARWINの許容同相電圧として記載されているノイズ電圧の250 Vac, 150V acは当然上記SELVの電流の制限を受けています。つまり、250 Vや150 Vがかかる場合は、電流値が上記の範囲内である必要があります。

4. MXのノイズ除去性能について

MXのノイズ除去性能は、A/D変換方法（積分型A/D）によるノイズ除去機能と、ファームウェア処理によるフィルタ機能（1次遅れフィルタ）の2段構成となっています。ここでは、A/D変換方法によるノイズ除去性能、および耐ノイズ性の指標として用いられるノーマルモード除去比（NMRR）・コモンモード除去比（CMRR）を中心に、MXのノイズ除去性能を解説します。

・積分型A/Dとそのノイズ除去性能

積分型A/Dは、指定時間幅で測定値を積分していきます。単純な矩形積分の場合、指定時間幅が取りたい周波数の周期と一致していれば、周波数成分は除去できます。積分時間が20 msであれば、それは、20 ms=1/50 Hzのことであり、50 Hzの電源周波数とその整数倍の周波数を除去することが出来ます。

図示すると以下の様になります。

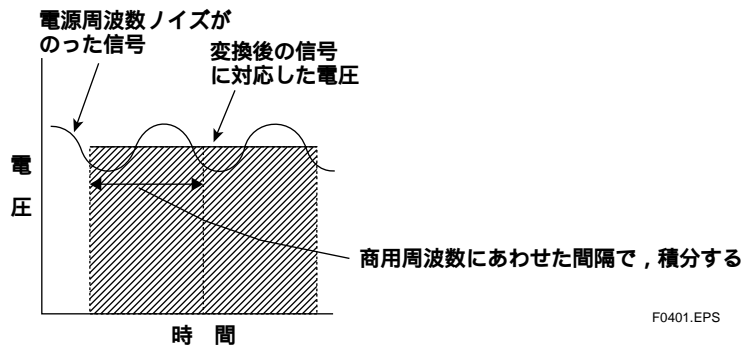


図4-1 積分型 A/Dの場合

同様に、16.67 msは60 Hzとその整数倍、100 msは10 Hzとその整数倍、1.67 msは600 Hzとその整数倍の周波数が除去できます。したがって、1.67 ms積分での測定は、電源周波数には強いとは言えません。

また、36.67 ms、200 msの積分は、MX100では上記のような単純な矩形積分とは計算過程を変えています。詳しい説明は省略しますが、36.67 ms積分の場合は、50 Hz、60 Hzとその整数倍の周波数が除去でき、200 ms積分の場合は、 F_c （カットオフ周波数、後述）= 5 Hzのローパスフィルタとして機能します。

MX100で用意されているA/D変換の積分時間とそのノイズ除去性能を表にまとめると、下表のようになります。なお、適用される積分時間は、入力モジュールの種類と設定する測定周期などによって自動的に決まりますので、詳細は各モジュールの仕様書をご覧ください。

A/D変換の積分時間とノイズ除去周波数

積分時間	対象除去周波数・備考
1.67ms	600Hzとその整数倍
16.67ms	60Hzとその整数倍
20ms	50Hzとその整数倍
36.67ms	50Hz, 60Hzとその整数倍
100ms	10Hzとその整数倍
200ms	$F_c=5\text{Hz}$ ローパスフィルタ

T0401.EPS

実際、ノイズがどのように除去されるか、周波数特性のグラフを見てみます。
 下図は、20 ms積分時の周波数特性グラフです。

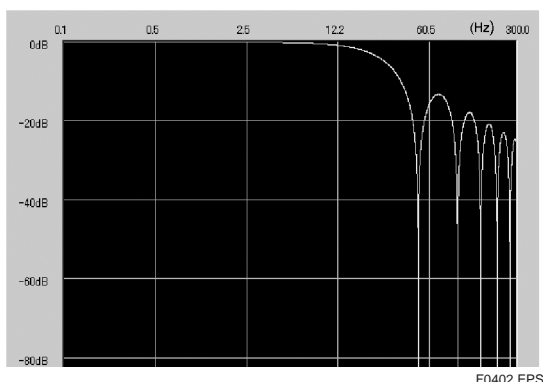


図4-2 20 ms矩形積分 横軸log表示 (理論値)

縦軸は、振幅比 (ゲイン), 横軸は周波数を示しています。振幅比 (ゲイン) とは、出力信号の振幅と入力信号の振幅との比であり、dB値で表示します。-(マイナス)のdB値は入力が減衰して出力されていることを示しています。(0は減衰していない状態) 上図から、数Hz程度の信号までは、振幅がほとんど減衰しないのに、数10Hz以上の周波数の信号の場合は、振幅をかなり減衰させることが分かります。

ただ、前の図からは、本当に50 Hzとその整数倍の信号が除去できているのか今一步はつきりしません。

そこで、横軸のlog表示をやめ、周波数の値でそのまま横軸にプロットしてみます。50 Hzとその整数倍の周波数が著しく減衰 (除去されている) しているのが良く分かります。

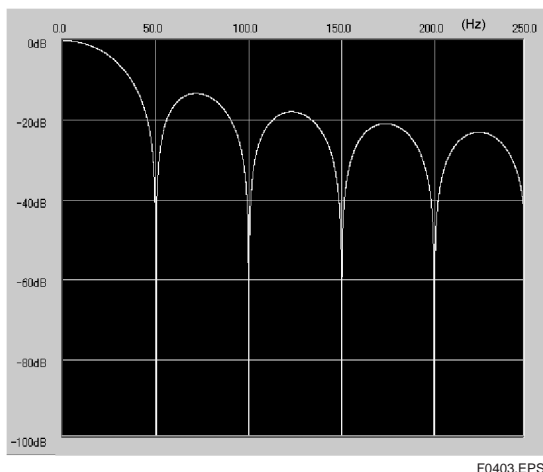


図4-3 20 ms矩形積分 (理論値)

では、- 20 dBや - 40 dBとは、どの程度の減衰にあたるのでしょうか？

dBとは、“ $20 \log | \text{出力信号の振幅} / \text{入力信号の振幅} |$ ” で表されます。この値が+なら増幅、-なら減衰を意味します。

$$- 20\text{dBとは、} - 20\text{dB} = 20 \log | \text{出力信号の振幅} / \text{入力信号の振幅} |$$

$$\log | \text{出力信号の振幅} / \text{入力信号の振幅} | = - 1$$

$$\text{出力信号の振幅} / \text{入力信号の振幅} = 1/10$$

という事になります。同様に、- 40dBは1/100、- 6dBは1/2、- 3dBは1/ 2になります。

次に、1.67 ms積分の周波数特性を見てみます。
前述した通り600 Hzとその整数倍の周波数を除去できます。一方、図4-4からも明らかな様に、50 Hzや60 Hzは除去できません。つまり、この積分時間での測定では、電源周波数に強いとは言えません。

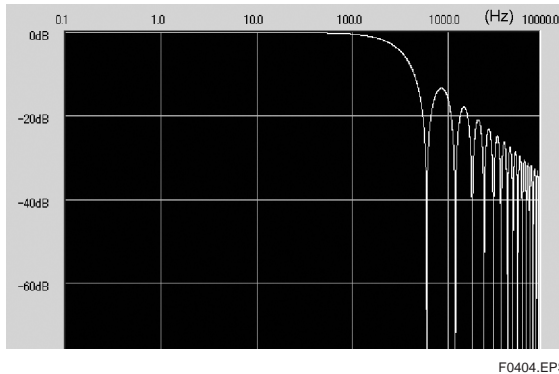


図4-4 1.667 ms矩形積分 横軸log表示 (理論値)

200 ms積分の場合は、下図のように、カットオフ周波数が5 Hzとなるローパスフィルタ (低い周波数帯を通す) になっています。5 Hzとその整数倍の周波数のノイズを除去可能なフィルタとの表現もできますが、5 Hz以降の除去性能が大きいので、カットオフ周波数が5 Hzとなるローパスフィルタと、仕様には記載しています。

カットオフ周波数とは、振幅比が -3dB (振幅が1/ 2になる所) になる周波数を言います。カットオフ周波数より低い周波数帯域を通過帯域、高い周波数帯域を遮断帯域と呼び、フィルタの能力を示す値に使います。

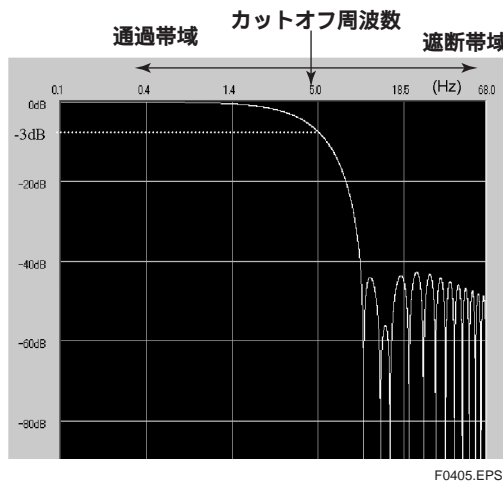


図4-5 200 ms矩形積分 横軸log表示 (理論値)

さて、ここで、MXの仕様を見てみましょう。

・ **ノーマルモード除去比 (NMRR) :**

積分時間16.67 ms以上時40 dB以上 (50/60 Hz ± 0.1%)

これは、“積分時間を16.67 ms以上にして測定実施中に、ノーマルモードに50/60 Hzのノイズがかかった場合、40 dB以上で50/60 Hzのノイズを除去できる。” という意味になります。

ノーマルモードとは、図4-6のごとくHigh - Low間にかかるノイズのことを言います。40 dBとは、前述の通り、 $V_{out}/V_{nm}=1/100$ のノイズ除去能力があることを示します。

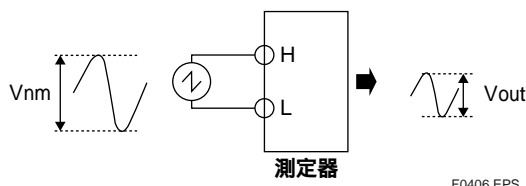


図4-6

図4-2 ~ 4-5で見てきた、周波数特性の図はこのノーマルモードの除去性能を示しています。

・ **コモンモード除去比 (CMRR) :**

積分時間16.67 ms以上時120 dB以上 (50/60 Hz ± 0.1%)

これは、“積分時間を16.67 ms以上にして測定実施中に、コモンモードに50/60 Hzのノイズがかかった場合、120 dB以上で50/60 Hzのノイズを除去できる。” という意味になります。

コモンモードとは、下図のごとくHigh - Low間に同相にかかるノイズを言います。(ノーマルモードとは、ノイズのかかり方が違うことに注意して下さい。)

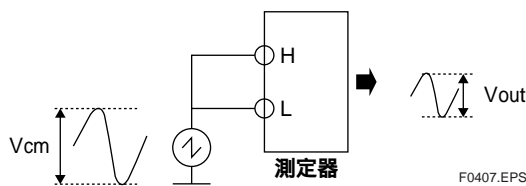


図4-7

コモンモード電圧は、同相にかかるノイズですので、High - Low間の差が常に一樣であれば本来測定に影響の無いノイズと言えます。しかし、実際は、H側とL側の測定回路の完全な対地絶縁は不可能であり、H側とL側とで同相にかかったノイズでも、測定器内では、どうしてもH - L間にアンバランスが生じてしまいます。そのアンバランスがノーマルモードノイズとして測定値に影響を及ぼしてしまうのです。よく電源周波数が信号線によって測定がうまく出来ないということを聞きますが、多くはこの現象によります。

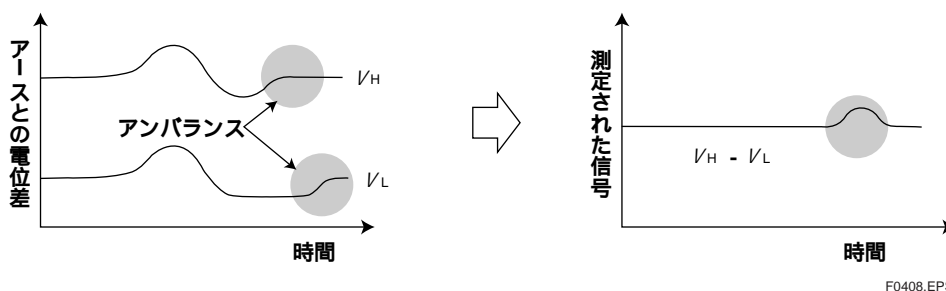


図4-8 コモンモードアンバランスの影響（概念図）

コモンモード除去比（CMRR）は、このようにコモンモードノイズがノーマルモードノイズに変換され、除去しきれずに出力誤差として現れてくる比率であり、先ほどのノーマルモード除去比（NMRR）を含んだ値となります。

では、コモンモード除去比120 dBとは、どの程度の誤差を生じるのでしょうか？
 120 dBとは、 $1/10^6$ を意味しますので、
 100 Vのコモンモードノイズ電圧がのった場合は、 $100 \text{ V} \div 10^6 = 100 \mu\text{V}$ 、熱電対Kで2～3 程度の誤差
 250 Vのコモンモードノイズ電圧がのった場合は、
 $250 \text{ V} \div 10^6 = 250 \mu\text{V}$ 、熱電対Kで6 程度の誤差になります。

・4ch高速ユニバーサルモジュール（MX110-UNV-H04）の優位性

MXでは、電圧や温度などを測定できるユニバーサルモジュールとして、4ch高速ユニバーサルモジュール（MX110-UNV-H04）と10ch中速ユニバーサルモジュール（MX110-UNV-M10）の2つのラインアップを用意しています。

仕様上、両者のノイズ除去性能（ノーマルモード除去比およびコモンモード除去比）は同じですが、4ch高速ユニバーサルモジュール（MX110-UNV-H04）は10ch中速ユニバーサルモジュール（MX110-UNV-M10）に比べ、

同一の測定周期では、A/D変換の積分時間をより長くとれる
 （同一のA/D積分周期では、より高速の測定ができる）

各チャンネル独立にA/D変換器を搭載しているため測定回路がよりシンプルとなり、高い周波数におけるコモンモードノイズからノーマルモードノイズへの変換量が小さい

という点で、より耐ノイズ性に優れていると言えます。

高速測定での耐ノイズ性や高い周波数における耐ノイズ性をお求めの場合は、ぜひ4ch高速ユニバーサルモジュール（MX110-UNV-H04）をご使用ください。

・一次遅れフィルタ

MXには、測定値を最終的に出力するところでファームウェア処理により一次遅れフィルタをかけることができます。丁度、抵抗とコンデンサで形成されるローパスフィルタがMXの出力最終段に設けられていることと同じ動作になります。

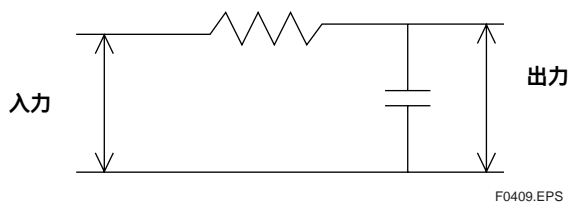


図4-9

一次遅れ系とは、下記のようなステップ入力が入った時、

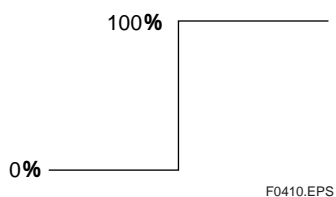


図4-10

下記のような出力特性（exponentialを用いた指数関数特性）を示す系を指します。

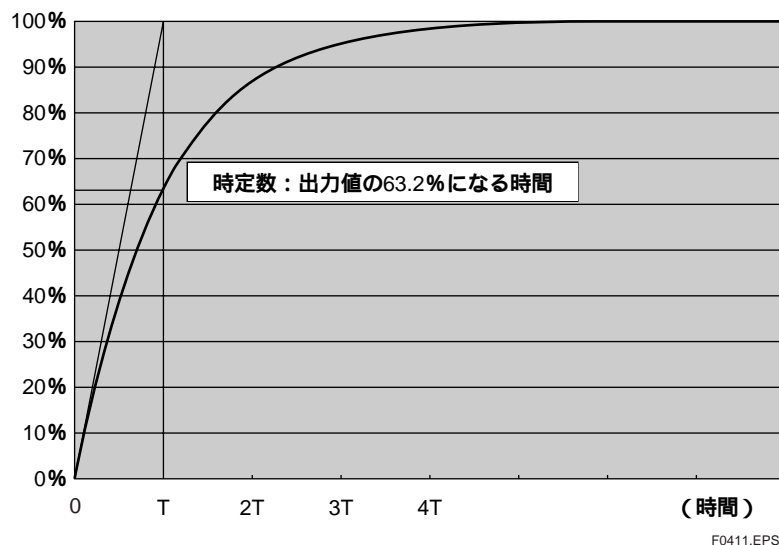


図4-11

仕様にある時定数とは、上記Tの値の事を意味しています。時定数を長くすれば、100%の出力になるまでに長い時間を要し、よりフィルタ機能が強くかかる事を意味します。（図4-12参照）MXでは、測定周期の5～100倍の時定数まで用意してありますので、取りたいノイズに合わせて柔軟にフィルタを設定できます。

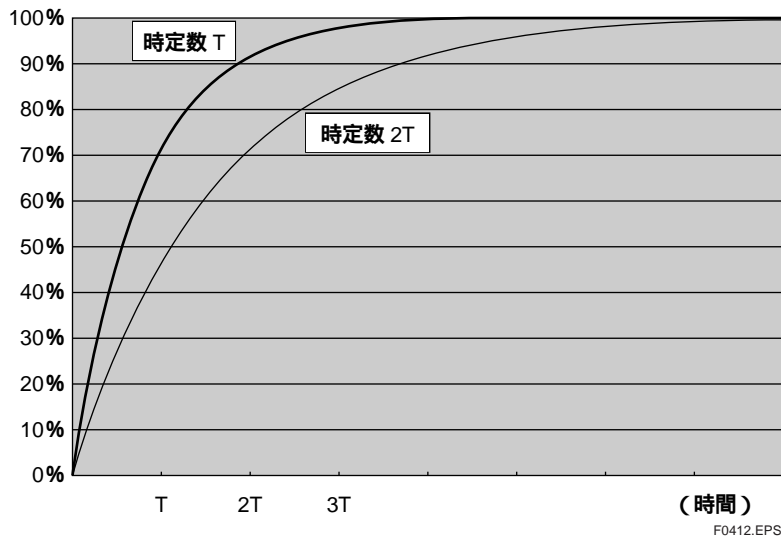


図4-12

さて、一次遅れ系の周波数応答をみます。図から、一目瞭然ですが、特定の周波数を強く除去するという特性は持っていません。周波数が高くなるにつれて徐々に除去性能が強まる特性を持っています。

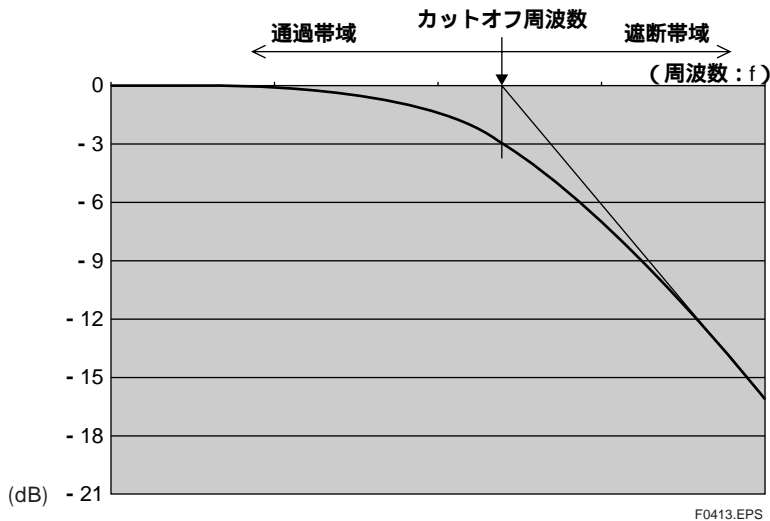


図4-13

参 考 ・ 移 動 平 均

ノイズ除去においては、積分型A/Dと一次遅れフィルタを双方利用することで実用上は十分と考え、MXには移動平均は搭載していません。

では、移動平均と一次遅れフィルタとはどう違うのでしょうか？ 参考に、サンプリング周期10 msで5点の移動平均をとる場合の周波数特性図を図4-14に示します。図からも明らかな様に、移動平均では、積分型A/Dの様に、ある特定の周波数を除去する特性があります。一次遅れフィルタ（図4-13）には、その特性はありません。

しかしながら、電源周波数以外のノイズで、決まった値のノイズ周波数というものは、一般的には見当たりません。一次遅れフィルタが移動平均の機能を完全に包括している訳ではありませんが、“電源周波数ノイズは積分型A/Dで除去し、それ以外のノイズは、一次遅れフィルタで除去する”という方式でノイズ除去という目的に対しては十分に機能します。

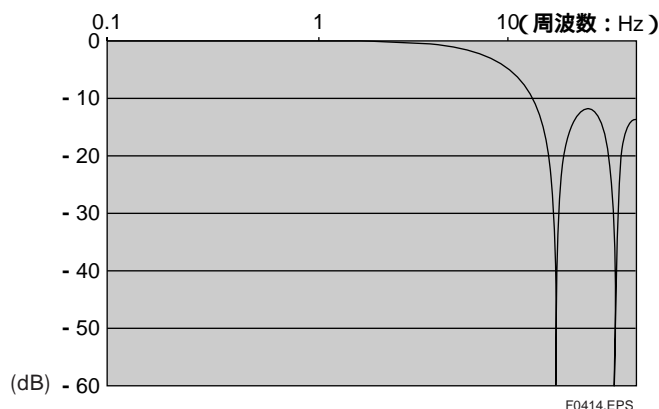


図4-14

最後に、ノイズ波形に対する、フィルタと移動平均の実際の効果（理論値）を示します。

図4-15, 16において, でプロットされた波形 “ Input ” が, 入力されたノイズ波形です。 それに対し, でプロットされた波形 “ Filter Out ” がフィルタをかけたあとの出力, でプロットされた波形 “ 移動平均 ” が移動平均をかけたあとの出力になります。

フィルタと移動平均には実用上差が無いことが, おわかりいただけると思います。

なお, 縦軸に単位はありません。 入力ノイズの大きさとフィルタ後のノイズの大きさを比べる指標にして下さい。

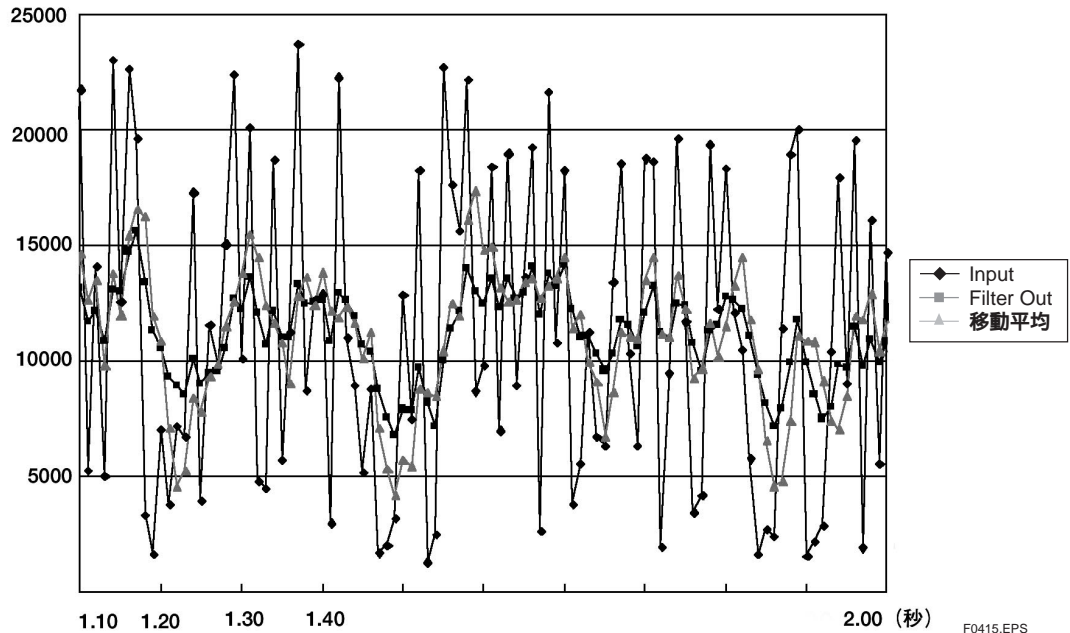


図4-15 フィルタ：測定周期10 ms, 時定数0.05 sのフィルタ
移動平均：5回の移動平均

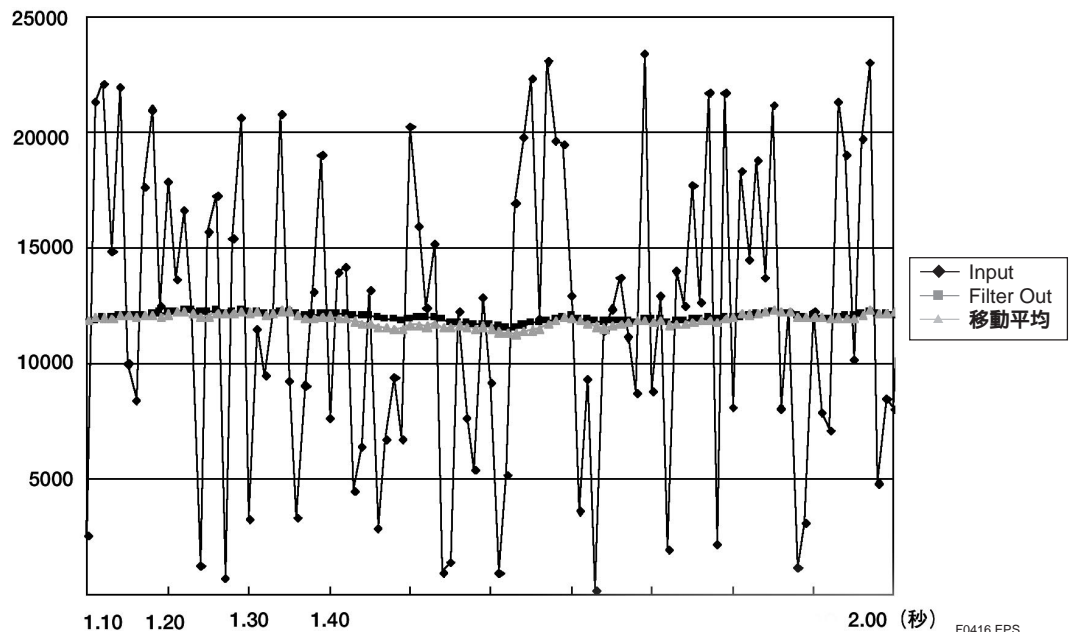


図4-16 フィルタ：測定周期10 ms, 時定数1 sのフィルタ
移動平均：100回の移動平均

5. MXの測定値時刻について

・MXの時刻

MXのデータに付加される時刻情報には、2種類の時刻情報があります。一つは、MXメインモジュールに内蔵されている時計の時刻情報で、もう一つがPCからの時刻情報になります。そのタイムシーケンスについて順を追って説明します。なお、PCソフトには、横河純正ソフトウェアを使用することを前提とします。

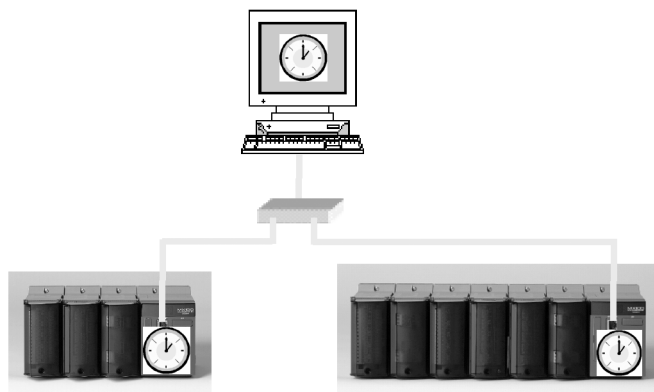


図5-1

・1ユニット内での時間管理



図5-2

MX 1ユニットは、メインモジュール (MX100) の時計により管理されます。この時計は、PCとは関係なく100ms周期で時間を刻んでいます。各入力モジュールは、PCからの測定開始要求後一番近いメインモジュールのクロック (100ms周期の刻み) をきっかけとしてモジュール別にデータ収集をスタートさせます。測定されたデータは、メインモジュールで測定周期ごとにまとめられ、PCの出力要求に従いIPCへ転送されます。

まず、入力モジュール単体で見えていきます。
 入力モジュールは、測定開始要求があっても、すぐに実際の測定が開始出来るわけではありません。まず、A/D誤差やRJC温度をチェックする時間が必要となります。図5-3に出てくる初期cal (calとはcalibration (校正) のことを意味しています。) という部分がそれにあたります。このcal動作は、測定中も連続的に実行されています。初期cal後、測定が開始されるのですが、測定周期ごとに即出力可能なデータが作られる訳ではありません。入力モジュールからメインモジュールへのデータ格納や、演算 (ch間差, スケーリング), アラーム, フィルタといった処理をメインモジュール内で実行する時間が必要になります。実際に出力可能状態になる時刻は、測定した時刻から2測定周期分, 3測定周期分遅れた時刻になります。(初期calの時間とデータ格納に要する時間は、測定周期により異なります。)

具体的な測定タイムシーケンスを以下に示します。

・4ch高速ユニバーサル入力モジュール (MX110-UNV-H04) の場合

各chが独立に測定を行います。(A/Dを4つ搭載しているため) したがって、各chの測定時刻は同期していると言えます。

測定周期が10 msと50 msの場合は、最短クロック100 msごとのデータの集合体をPCに転送します。下に、10 msサンプル状態で出力データが一つできるまでを示します。

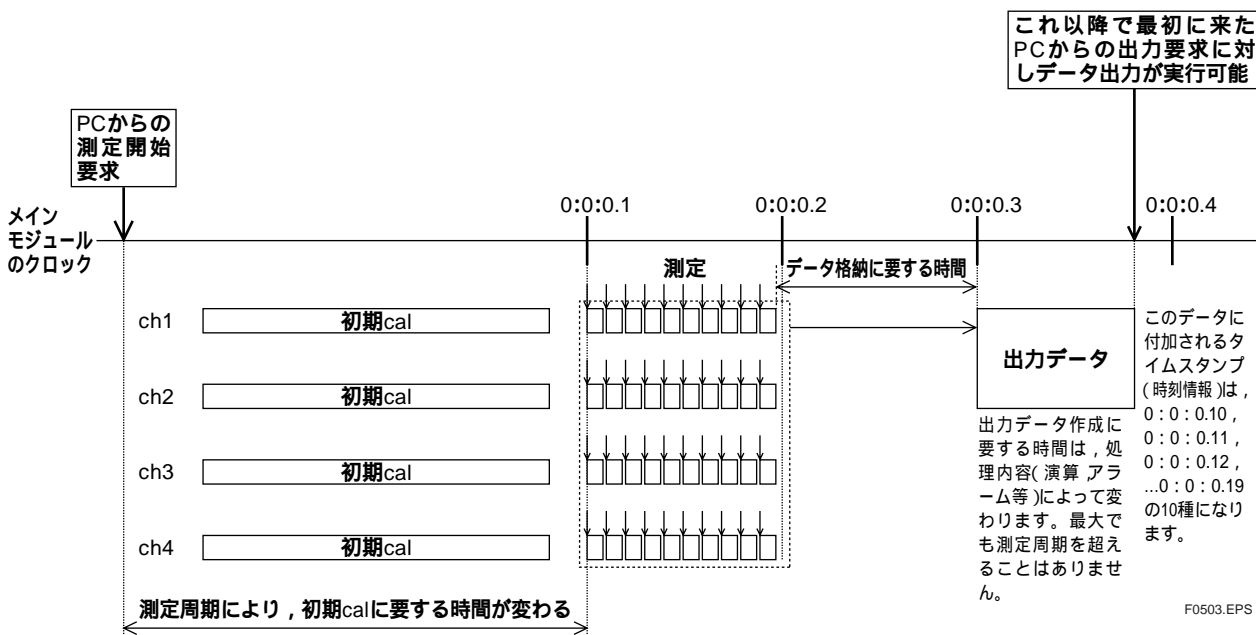
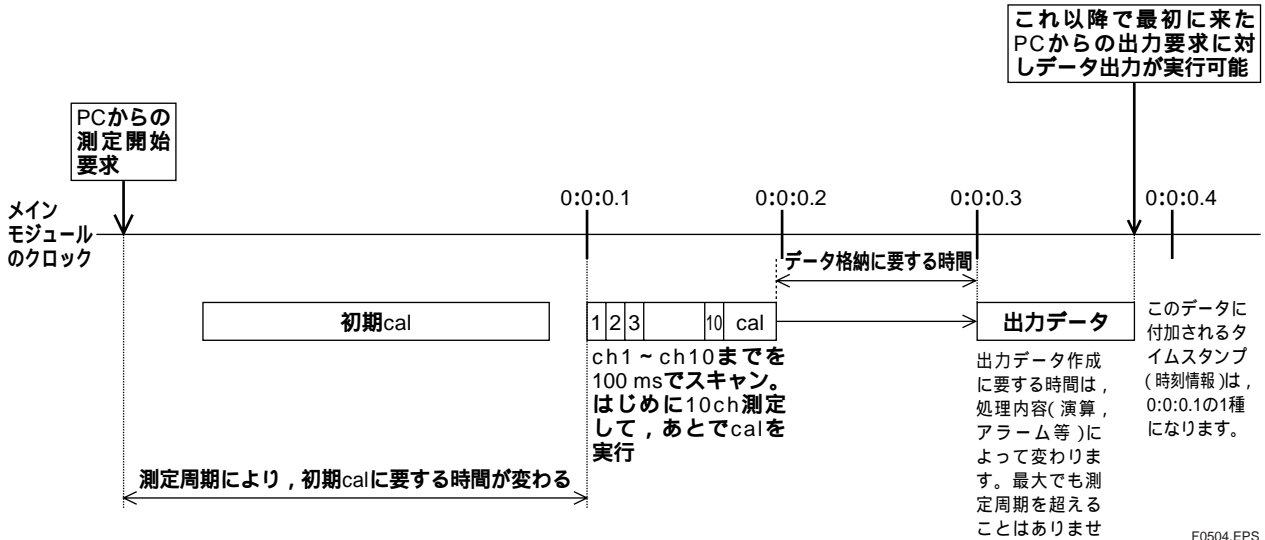


図5-3

10 msの間に、測定とcalを実行します。

・10ch中速ユニバーサル入力モジュール (MX110-UNV-M10)

A/Dは一つしかありませんので、10chを指定測定周期でスキャンしていきます。
 図5-4に、100msサンプル状態で出力データが一つできるまでのタイムシーケンスを示します。ch1からch10までを100msの間でデータをスキャンしているため、ch間のデータ時刻同期性は、“測定周期内で同期”ということになります。



F0504.EPS

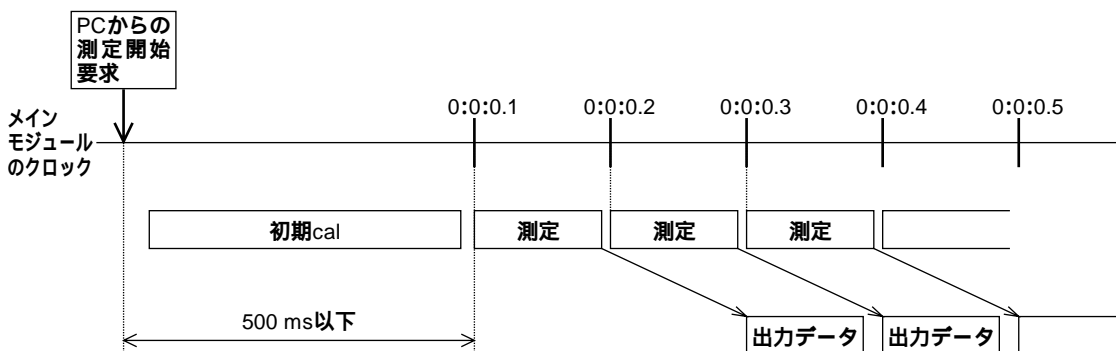
図5-4

上では、1つの出力データが出来るまでを示しましたが、実際の連続測定の例を下に示します。

アナログ入力モジュールの場合、測定周期とモジュールタイプの組み合わせにより、測定から出力データ完成までのタイムシーケンスに違いがでます。組み合わせによって、初期calの有無、出力データが出来るまでの時間に差が有る事に注意して下さい。
 なお、これ以降の図では、測定中のcal動作は書かずに、測定=測定+calとして、測定とだけ表記しています。

4ch高速ユニバーサル入力モジュール (MX110-UNV-H04) で測定周期10ms/50msの場合)

10ch中速ユニバーサル入力モジュール (MX110-UNV-M10) で測定周期100msの場合)



F0505.EPS

図5-5

4ch高速ユニバーサル入力モジュール(MX110-UNV-H04)で測定周期100msの場合)

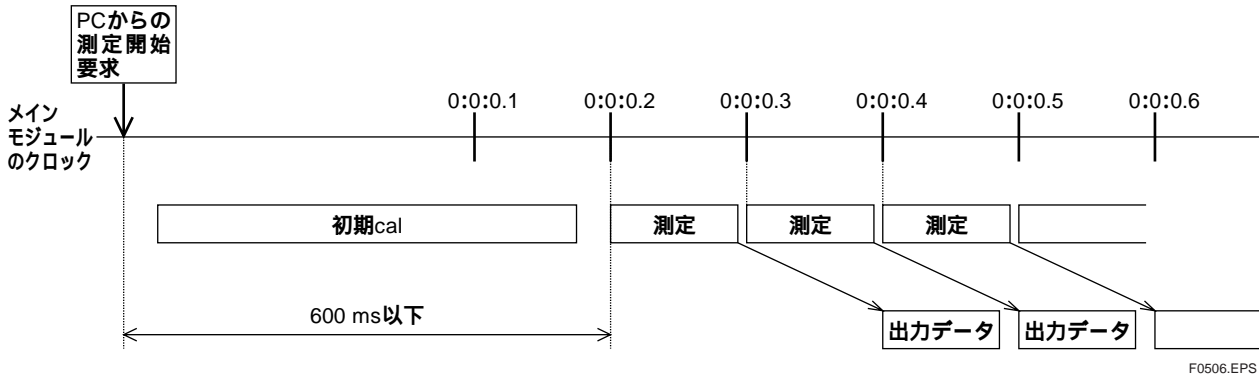


図5-6

4ch高速ユニバーサル入力モジュール(MX110-UNV-H04)で測定周期200ms/500ms/1sの場合)
 10ch中速ユニバーサル入力モジュール(MX110-UNV-M10)で測定周期200ms/500ms/2sの場合)

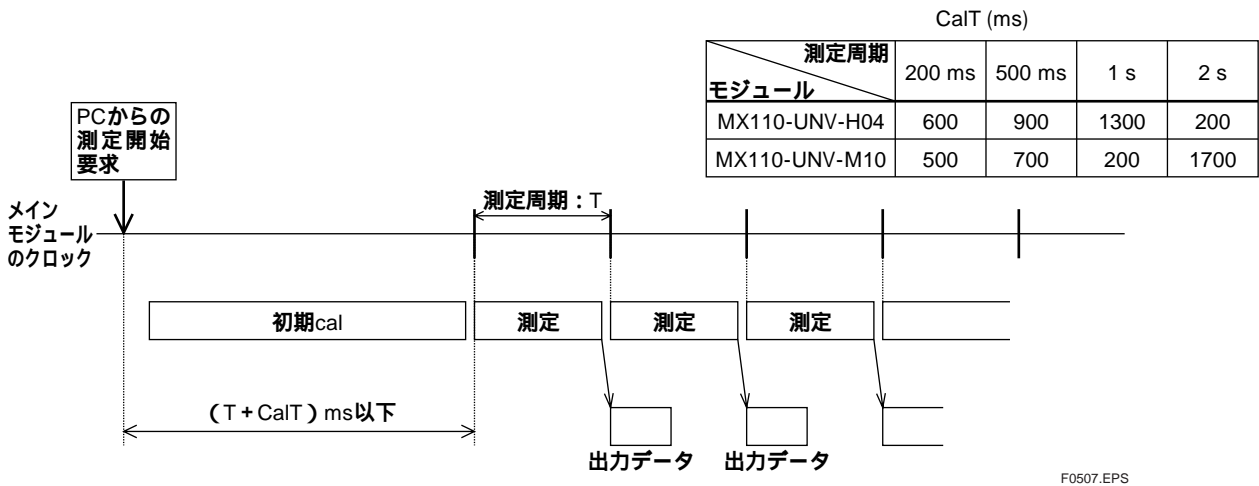


図5-7

4ch高速ユニバーサル入力モジュール(MX110-UNV-H04)で測定周期2s以上の場合)
 10ch中速ユニバーサル入力モジュール(MX110-UNV-M10)で測定周期1s/5s以上の場合)

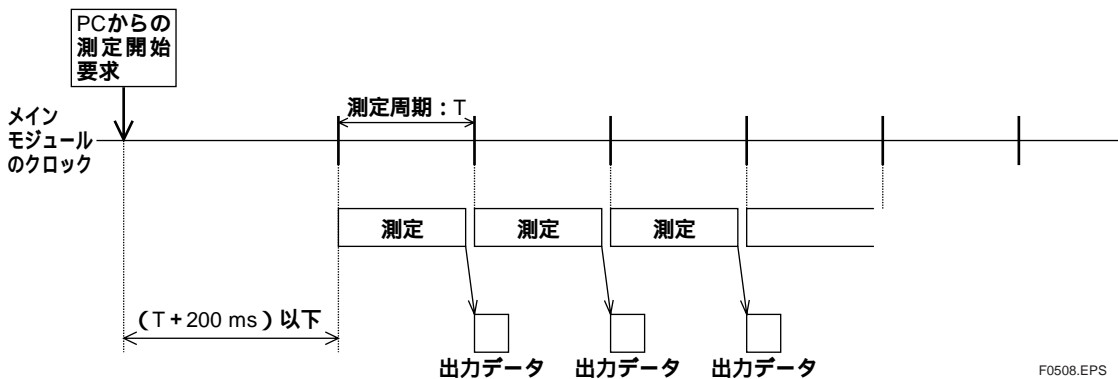


図5-8

測定周期10s/20s/30s/60sの場合)

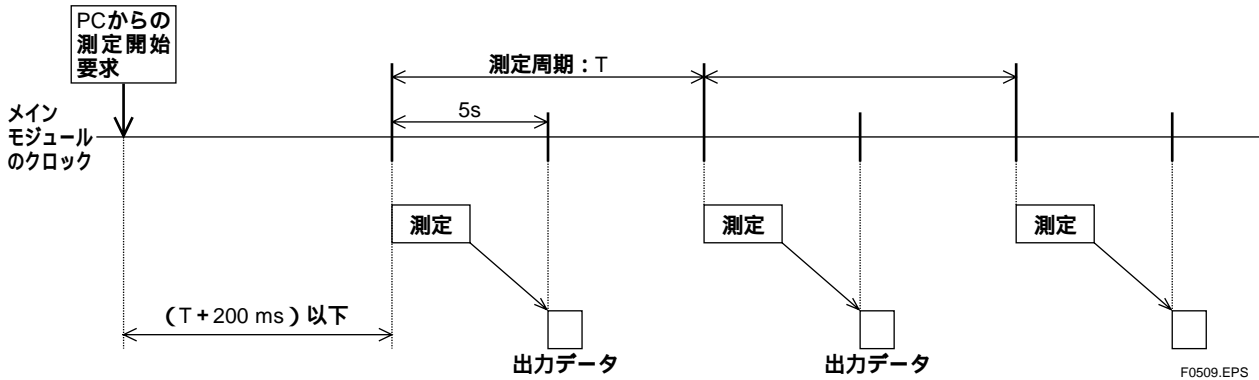


図5-9

・10ch高速デジタル入力モジュール (MX115-D05-H10)

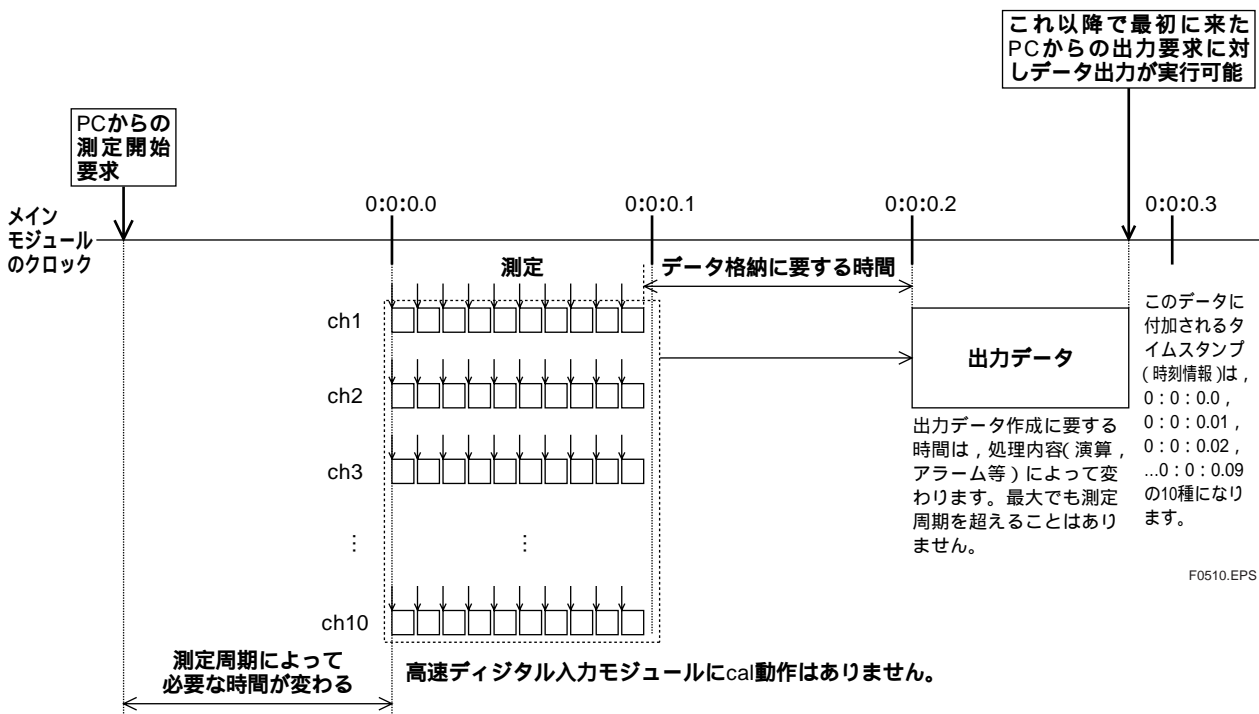
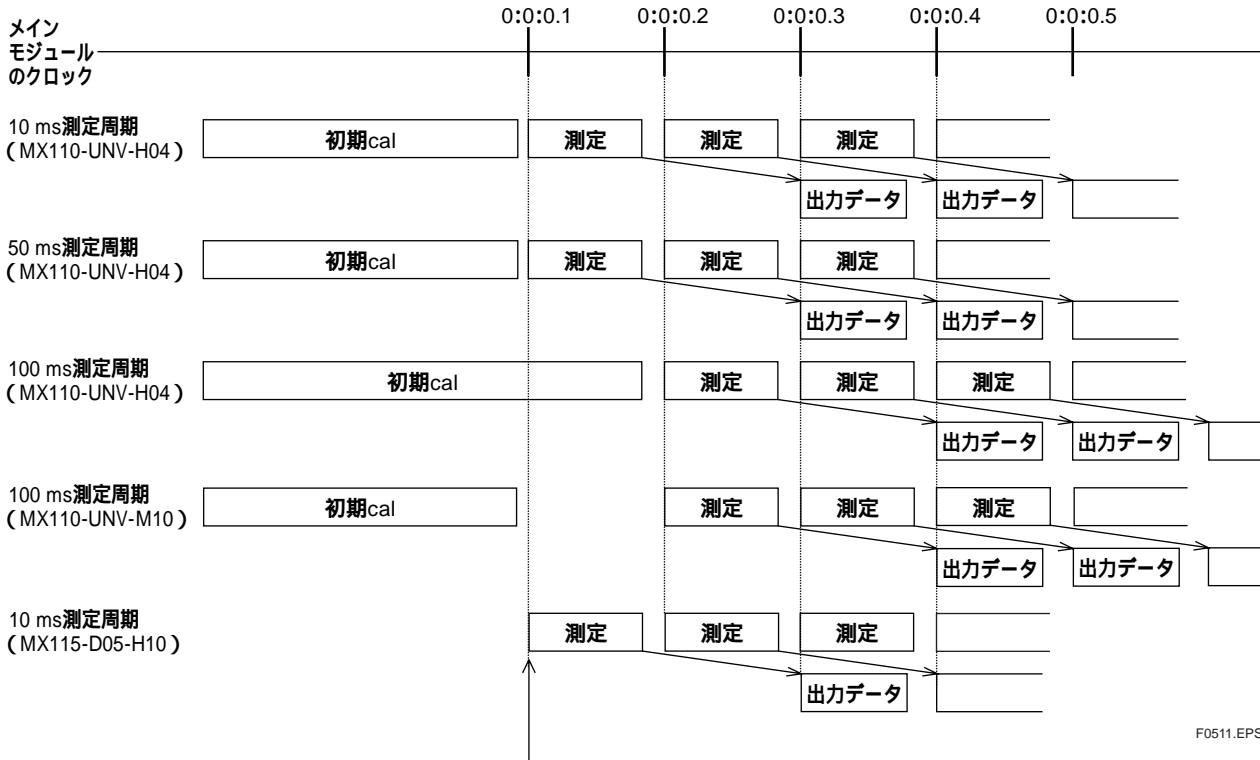


図5-10

これまでは、同一測定周期で単一モジュール内での測定から出力データ作成までの流れを見てきました。

複数モジュール収集についても単一モジュールの場合と同様に考えることが出来ます。1モジュールごとのタイムシーケンスが並列に動作していると考えて下さい。

注) 1ユニット内における各モジュールの測定開始時刻は、メインモジュール内部クロック(100msの刻み)に対して0.03ms~2.00ms遅れます。図5-11では、その遅れは無視しています。



デジタル入力モジュールには、初期calがありませんが、同じ測定周期のアナログ入力モジュールがある場合は、そのアナログ入力モジュールの初期cal動作を待って測定が開始されます。

図5-11

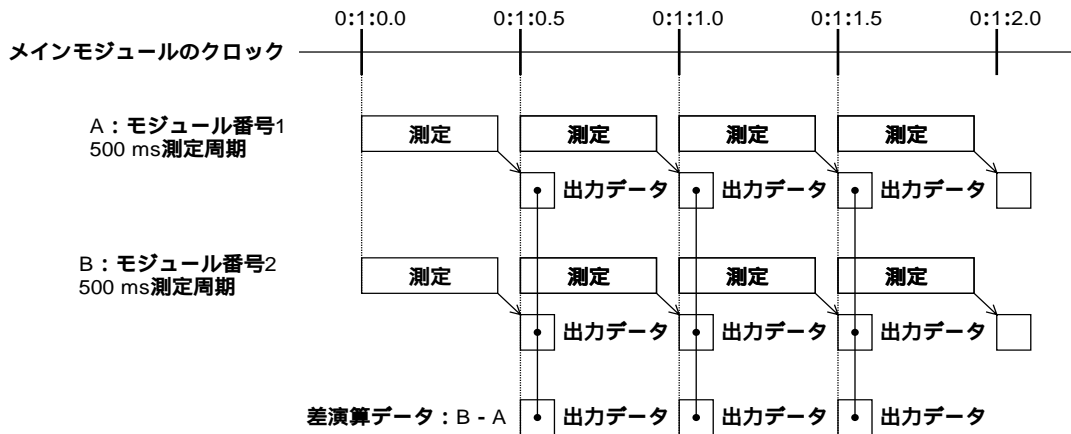
・チャンネル間演算

チャンネル間演算において、対象chの測定時刻を厳密にしたい場合は、同一モジュール内でしかも基準chには小さいch No.を指定することをお勧めします。

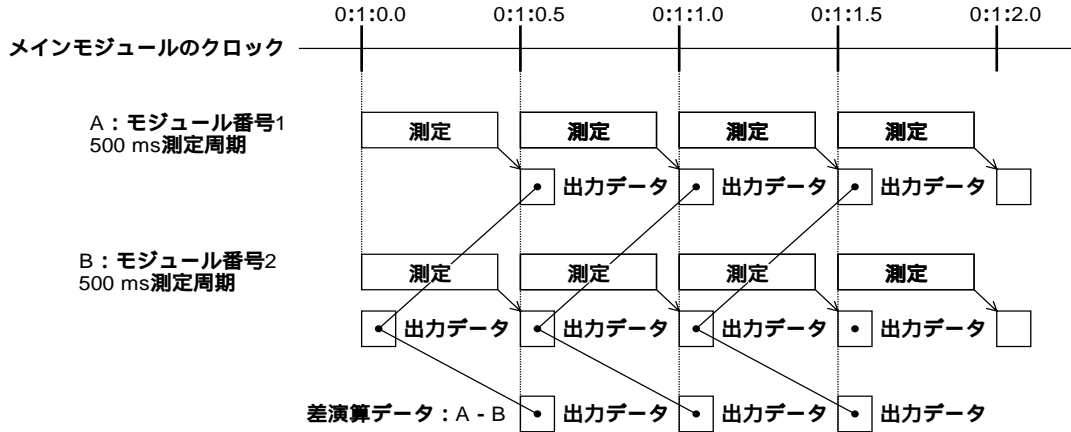
これまで、測定値を出力可能データにするには若干時間がかかることをご説明しました。この変換作業は、測定周期の短いモジュール、モジュール番号の小さい順、ch No.の小さい順で、順次処理されます。したがって、差をとるchの条件によって、差演算データの元となるデータに違いが生じてしまいます。

図5-12に示します。基準となるchの取り方によって、計算の元となるデータ時刻に違いがあることに注意下さい。

基準となるchを A：モジュール番号1とした場合



基準となるchを B：モジュール番号2とした場合



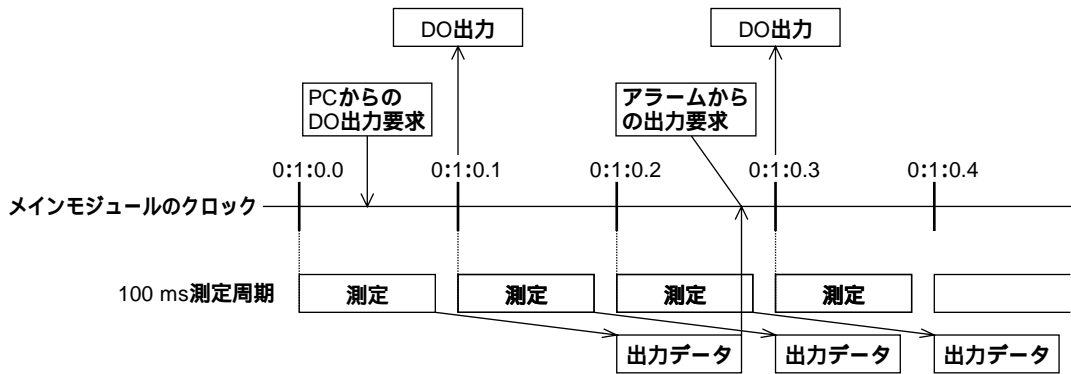
F0512.EPS

図5-12

・10ch中速デジタル出力モジュール (MX125-MKC-M10)

測定クロックと同期して100 msごとに動作します。出力タイミングでそれまでに来たDO出力要求を実行します。

下の図では、ユニバーサル入力モジュールの初期cal動作は終了し定常測定に入っている状態を示しています。なお、デジタル出力モジュール (MX125-MKC-M10) にcal動作はありません。



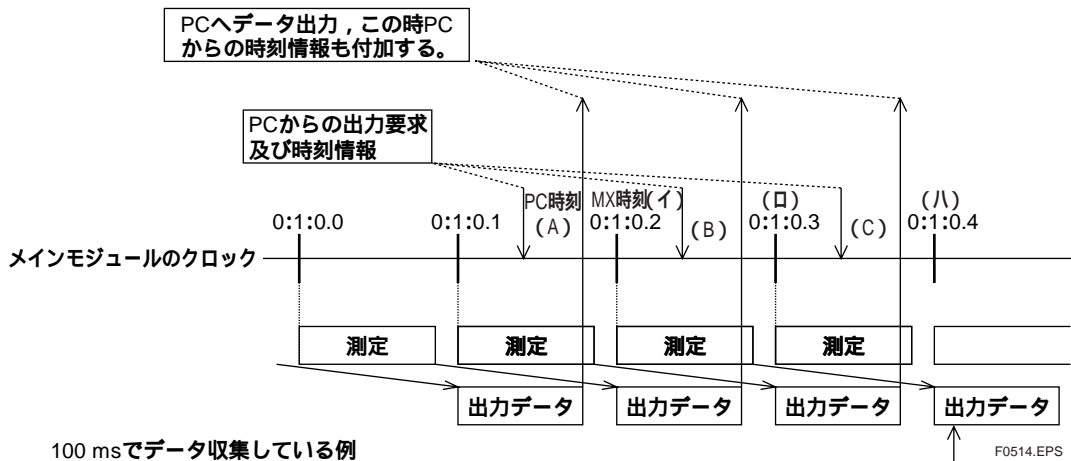
F0513.EPS

図5-13

・PCとの通信 (横河純正ソフトウェアの場合)

PCソフトからのデータ出力要求に従い、MXはその時点で用意できるデータをPCに返します。その時、PCの時刻情報もデータに付加して送信します。

(PC時刻は、データ出力要求に限らず1通信毎に送られてきます。このPC時刻情報を直近の開始する測定データに付加します。)



F0514.EPS

この出力データに付加される時刻情報はPC時刻(A)とMX時刻(イ) (0:1:0.2)になります。

図5-14

ところで、これまで初期calについて述べてきましたが、横河純正ソフトウェアを使用する上では、ユーザが初期cal動作を意識する必要はありません。接続すれば、自動で初期calを実行し、測定状態 (モニタ状態) になります。レンジ変更した場合でも自動で初期calまで行います。

・リアルタイムモニタ画面

リアルタイムモニタ表示画面では、PCの時刻情報ではなく、MXメインモジュールの時刻情報を元にデータ表示をしていきます。

・ビューア画面

一方、保存データは、MXメインモジュールの時刻情報、PCの時刻情報のどちらかを選択して再表示させることが出来ます。なぜ、PCの時刻情報を表示する必要があるのでしょうか？

下記の様に、PC1台で2ユニットのMXデータを収集する場合を考えます。

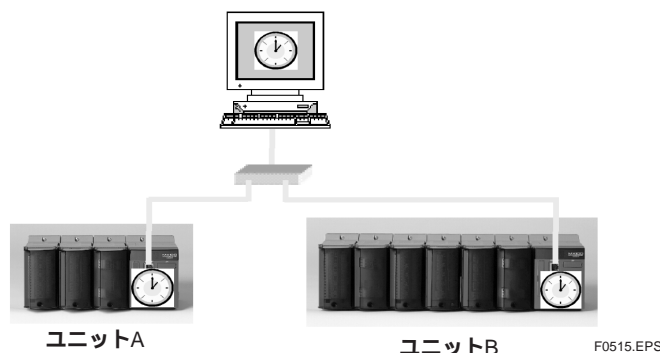


図5-15

ユニットAとユニットBの時計が完全同期していればMXの時刻だけを頼りに、データ収集すれば良いのですが、実際には、同期しません。

どの程度ズれるかは、時計精度によります。MXメインモジュールの時計精度仕様は、±100 ppmです。この仕様ですと、一週間で、 $3600 \times 24 \times 7 \div 1000000 \times 100$ 60秒になります。つまり、1秒周期で1週間連続してデータ収集を行った場合、同期間データ収集したにもかかわらず、ユニットAとユニットBの間で最大120個(60×2)もデータ数が違ってくる可能性が出てきます。ビューア画面ではそのズレを吸収するためにPCの時刻を使いデータを補正し再表示をさせています。

データ時刻補正の概念を説明します。(理解を簡単にするために、下の事例は、データ数や測定時間を極端に描いています。)

PCの時刻で5秒間のデータを収集したとします。その期間でMXから帰ってきたデータが7個存在したとします。まず、その7個を使い測定値間を補間し波形を作成します。次にその補間曲線上のPCの時刻での値を、PCの時刻での測定値とします。

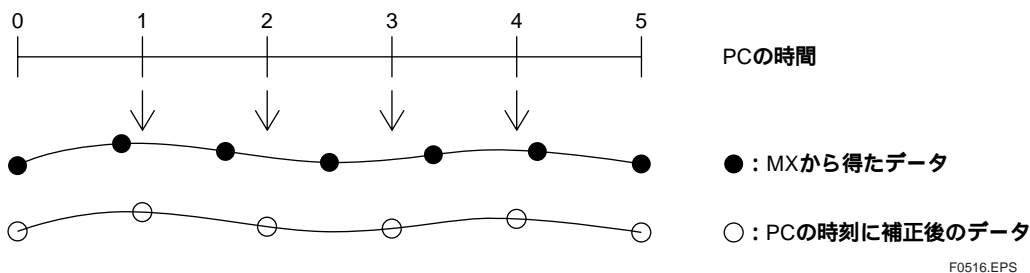


図5-16

この動作をユニットA，ユニットBの双方に施す事により，どちらのユニットのデータもPC時刻で補正後の測定値を持つ事になります。時間軸があえば，あとはそのデータを並べればユニット間をまたいだ波形を描くことが出来ます。

MXのデータに付加されたPCの時刻と，実際サンプルされた時刻との間には，1測定周期（最短で100 ms）内のズレを含んでいます。（例：図5-14，PC時刻（A）とMX時刻（イ）0:1:0.2とのズレ）しかし，上記の補正処理によりユニット間での時刻ズレがどの時点でも1測定周期（最短で100 ms）内であり，時刻差が積算されない表示が可能になります。

6. ひずみ変換ケーブル (DV450-001) の使用について

DAQMASTERのカタログやIMのひずみモジュールの項目には，“センス線がないひずみゲージ式センサをMX112-NDI-M04で使用する際は，DV450-001（DARWINアクセサリひずみ変換ケーブル）を介して結合してください。”という注意書きがされています。ここでは，DV450-001の必要性について説明します。

MX112-NDI-M04は，センス線付き（リモートセンシング機能付き）ひずみゲージ式センサに対応しており，結線の概略は図6-1のようになります。この場合は，直接センサをひずみモジュールに結合するだけで正しく測定できます。

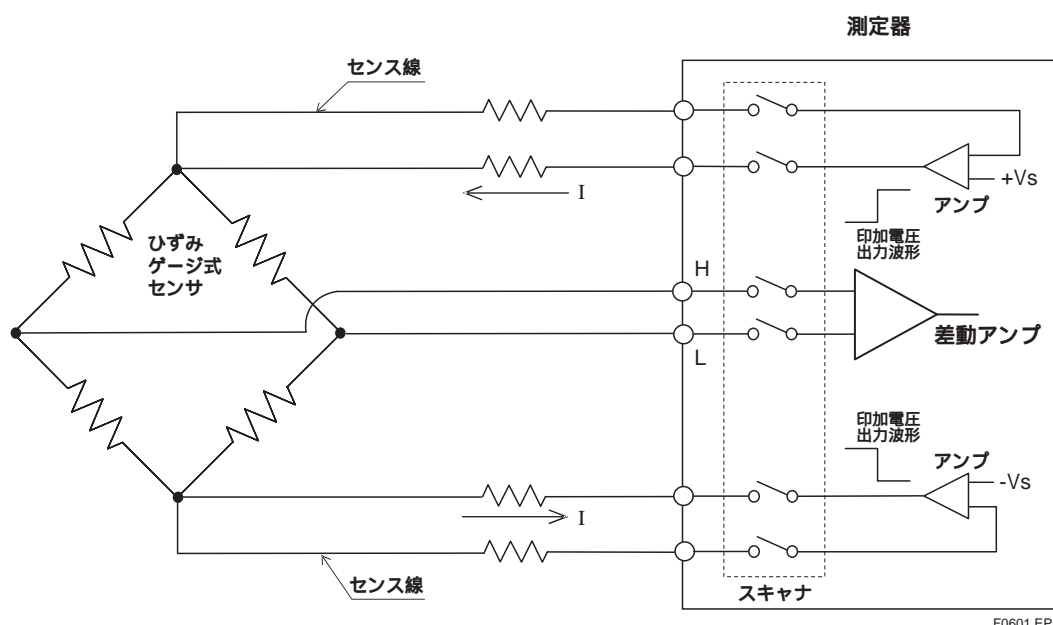


図6-1

ところで，ひずみゲージ式センサには，センス線無し（リモートセンシング機能無し）のセンサも多く存在します。配線抵抗に起因する誤差を考慮する必要の無い測定（配線が短い測定）の場合には，より安価なセンス線無しのセンサが使われることが多いようです。

そのセンス線無しのセンサをMX112-NDI-M04にDV450-001を使用しないで結合しますと図6-2の様になります。

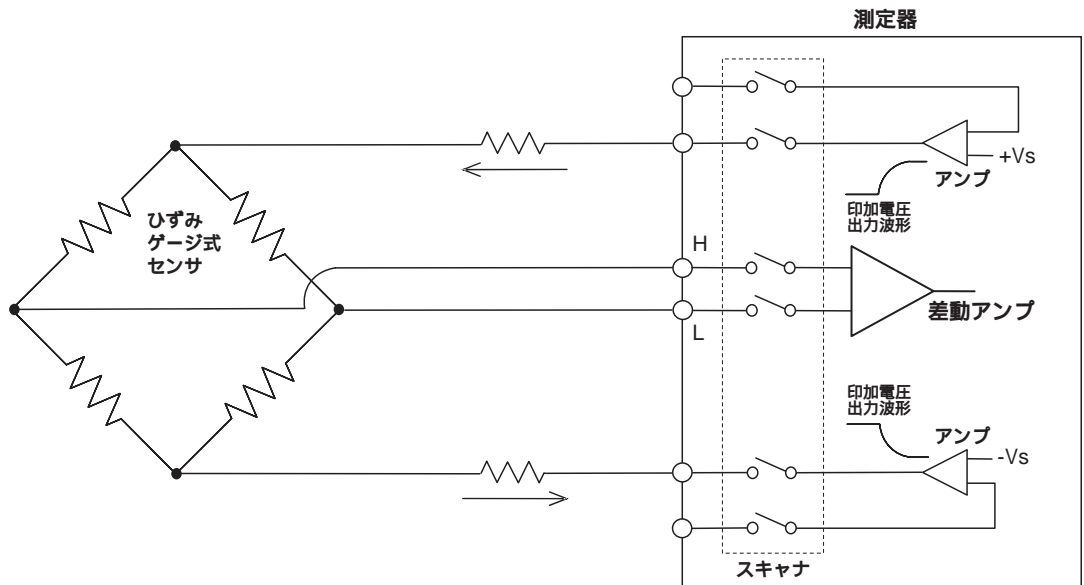


図6-2

図6-2の状態では、MX112では測定ができません。図6-1と図6-2の図中にある印加電圧出力波形を比較してください。図6-2のアンプの応答が遅いことがわかります。図6-2のアンプの応答が遅くなる理由は、MX112がスキャナ（切り替えスイッチ）を使っていることによります。MX112のようなスキャナタイプの測定器は、1チャンネル当りの測定時間に制限があるために、アンプが短い時間で応答することが必要になります。図6-1のアンプは、センス線からブリッジ電圧を検知しているので一回の測定時間内に遅延ない応答が可能なのですが、図6-2のアンプは、ブリッジ電圧を検出できないため、応答遅れが発生してしまいます。結果として、図6-2では適切な印加電圧にならないままに一回の測定が終わってしまい正確な測定が行えません。

この応答遅れを解消するために用いるのがDV450-001です。DV450-001を介してセンス線無し（リモートセンシング機能無し）のひずみゲージ式センサとMX112を結合した状態が図6-3になります。

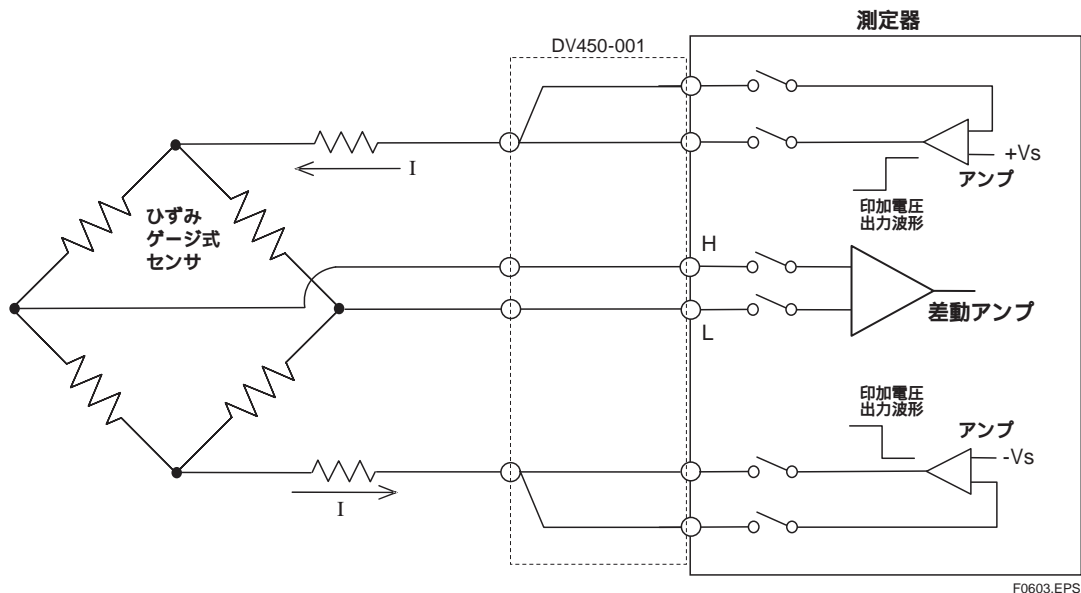


図6-3

DV450-001は、図のようにセンス線とブリッジ電圧端子をケーブル内でショートしています。このケーブルを使用することにより、ブリッジ電圧の応答おくれの問題が解決され、安定した測定が可能になります。ただし、ここで注意すべきは、図6-3ではリモートセンシングは行われていないと言う点です。つまり、配線抵抗による誤差は図6-3での測定では解消されていません。DV450-001より先の配線が長い場合は、注意が必要になります。

最後に、リモートセンシングについて簡単におさらいしておきます。詳しくは、専門書もしくはセンサメーカーの説明書等をごらんください。

単純にブリッジに電圧を印加する測定器を考えます。図6-4のように配線抵抗が無視できない場合（配線が長い場合）は、ブリッジへの印加電圧は、配線抵抗分低下してしまいます。（ブリッジ両端に $\pm V_s$ の電圧を印加したくても配線抵抗分電圧低下してしまいます。）

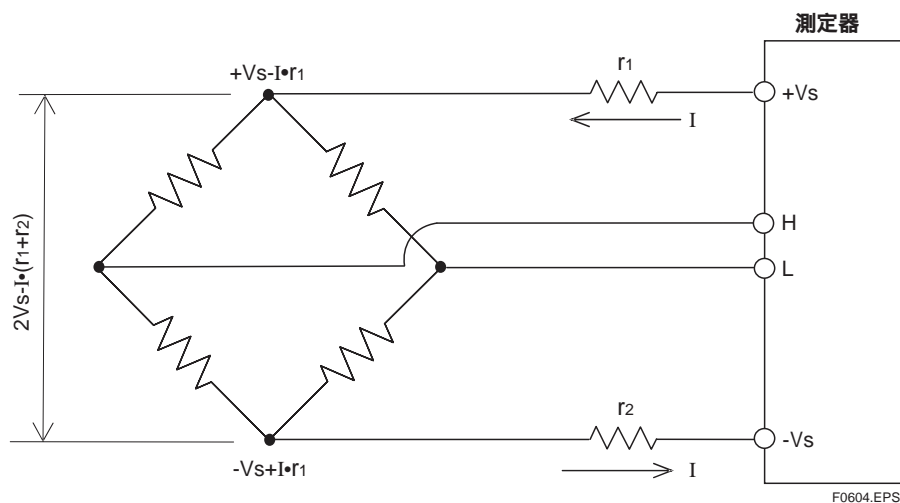


図6-4

この問題を解決するため、リモートセンスが使われます。図6-5のように、センス線を追加し、ブリッジの両端が正確に $\pm V_s$ になるようにアンプに帰還をかけます。(アンプは、配線抵抗の電圧降下分を足して出力します。)これにより配線抵抗に影響されない測定が可能になります。

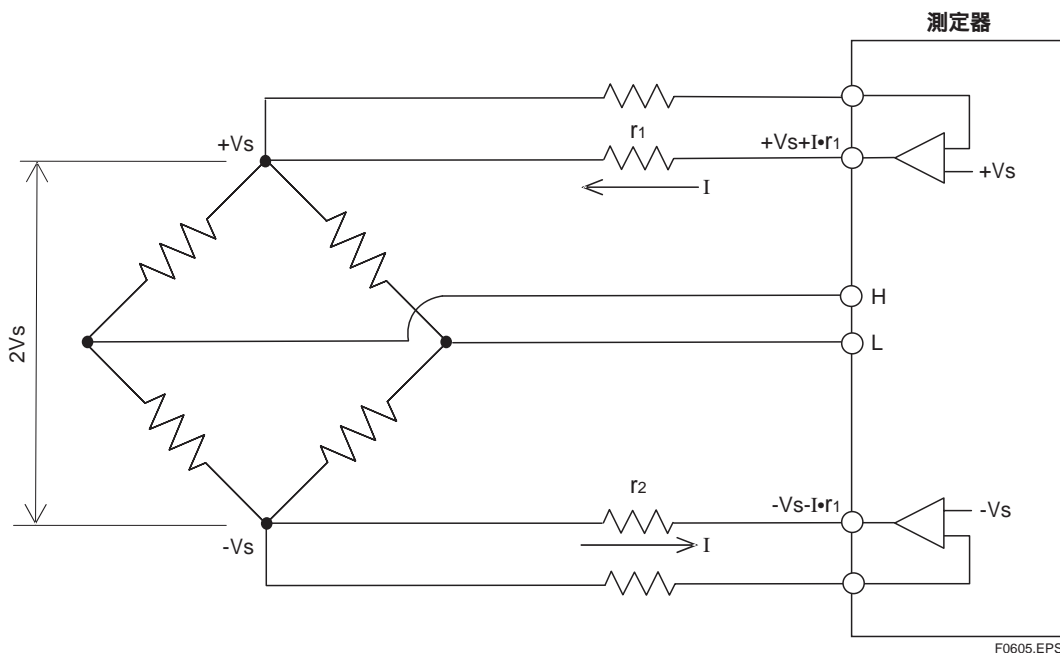


図6-5

7. 横河純正PCソフトウェアによる演算機能について

MX100は、多くの演算をPCソフトウェア側にゆだねています。したがって、横河純正PCソフトウェア（MX100スタンダードソフトウェア / DAQWORX MXLOGGER）には、多くの演算子が用意されています。弊社DARWINシリーズがハードウェアに内蔵していた演算機能のほとんどが実行可能です。また、定数を直接演算式に書き込める等、使い勝手の向上にも気を配った作りとなっています。反面、測定を行うMX100本体と演算を行うPCとの間には通信が介在するため、アプリケーションによってはテクニックを要する場面も出てきます。

本章ではまず、通信の介在を考慮したテクニック（手法）を紹介します。その後の事例紹介でしばしば登場しますので、適宜参照するようにしてください。詳しい解説は本章の最後「7.7 演算構築上の注意点および回避テクニック（解説）」で行います。続いて、弊社DARWINシリーズやDAQSTATIONシリーズの演算機能との違いのポイントをご説明します。すでにDARWINシリーズやDAQSTATIONシリーズの演算機能をご利用になったことのある方にとって、MX100用横河純正PCソフトウェアの演算機能をご理解いただく近道になれば幸いです。

そして、第7.1節～第7.6節で、測定の幅を広げる演算の事例を紹介していきます。お使いのアプリケーションに応じた演算式を手取り早く知るには、まずこれらを参照されることをおすすめします。紹介する事例は、次のとおりです。

第7.1節 事例：測定チャンネルの状態による動作制御

第7.2節 事例：イベントを数える

第7.3節 事例：時間の使い方

第7.4節 事例：統計演算

第7.5節 事例：操作・出力関係（Release 2以降）

第7.6節 事例：その他

これらの事例にて紹介する演算式の多くには、MX100本体とPCとの間に通信が介在することを考慮したテクニックが含まれています。そのテクニックはアプリケーションを通して共通的に使うものですが、さまざまな実アプリケーションに応用していくには、その内容を十分にご理解いただく必要があります。本章の最後「7.7 演算構築上の注意点および回避テクニック（解説）」で詳しく解説しますので、必要に応じて参照してください。

・通信の介在を考慮したテクニック（手法）

MX100用横河純正PCソフトウェアの演算構築にあたっては、測定を行うMX100本体と演算を行うPCとの間に通信が介在することにより、以下の点に注意が必要となります。

測定チャンネルへの参照結果が、[NaN]（不定な値、またはエラー値）になることがある

測定チャンネルへのprech()関数の参照結果が、ch()関数の前回値とは限らない

これらの注意点を考慮すると、アプリケーションによっては次のテクニックが必要になる場合があります。詳しい解説は本章の最後「7.7 演算構築上の注意点および回避テクニック（解説）」をご覧ください。

表7-1 通信の介在を考慮したテクニック (手法)

No.	気をつける場面	手法	記述例
1	測定値に基づく動作条件判別	IsNaN()関数を使用して[NaN]への保護を入れる	IsNaN(ch(<M01>))?[NaN]時の処理}:{本来の処理} "ch(<M01>)" "(IsNaN(ch(<M01>))?[NaN]の代替値):ch(<M01>)"
2	TLOG関数によらない測定値加算	sum()関数を使用して[NaN]を回避	"({加算対象}+ch(<M01>))" "sum({加算対象}, ch(<M01>))"
3	測定チャンネルの前回値比較	演算チャンネルに取り込んでから比較	<C91> = IsNaN(ch(<M01>))?prech(<C91>):ch(<M01>) <C92> = IsNaN(prech(<C91>))?[比較しない]:{比較処理}

凡例 <M01>: 測定チャンネル番号 <C91><C92>: 演算チャンネル番号

T0701.EPS

・他機種との比較

MX100をお買い求めのお客様の中には、弊社DARWINシリーズやDAQSTATIONシリーズの演算機能をご利用になったことがある方もいらっしゃると思います。そこで、DARWIN DA100およびDAQSTATION DXシリーズの演算オプションとMX100スタンダードソフトウェアとの比較を、表7-2に示します。

表7-2 他機種との演算機能の比較

項目	DAQMASTER MX100 スタンダードソフト	DARWIN DA100 演算オプション	DAQSTATION DX シリーズ演算オプション
絶対時間タイマ	なし	" 毎日 時 分を基準に1分~24時間毎 " の形式で相対時間タイマとあわせて6個設定可	" 毎日 時00分を基準に1分~24時間毎 " の形式で相対時間タイマとあわせて3個設定可
相対時間タイマ	秒単位で8個設定可	" 0~31日 時間 分毎 " の形式で絶対時間タイマとあわせて6個設定可	" 時間 分毎 " の形式で絶対時間タイマとあわせて3個設定可
絶対時間タイマ / 相対時間タイマ活用先	演算式にてtimer()関数で取得	「イベント/アクション機能」のイベントとして機能	TLOG演算のリセット間隔のみ
相対時間タイマの初期化	ResetTimer()関数による任意タイマもしくは全タイマのリセット	イベント/アクション機能のアクションとして全相対時間タイマを一括リセット	なし
マッチタイム	なし(かわりにhourly(), daily()などの関数で直接取得可)	" 毎月1~31日 時 分 " か " 毎日 時 分 " の形式で3個設定可	" 毎月 日 時 / 毎週 曜日 / 毎日 時 / 毎時 " の形式で1個設定可
マッチタイム活用先	演算式にてhourly(), daily()などの関数を使用可	「イベント/アクション機能」のイベントとして機能	メモリタイムアップのみ
時系列統計演算	TLOG関数による指定チャンネルのMax, Min, P-P, Sum, Ave時系列演算	TLOG関数による指定チャンネルのMax, Min, P-P, Sum, Ave時系列演算	TLOG関数による指定チャンネルのMax, Min, P-P, Sum, Ave時系列演算
TLOG演算の初期化	ResetTLog()関数による全TLOG演算の一括リセット	RESET()関数による任意TLOG演算のリセット	タイマによるリセットのみ
TLOG積算時の積算単位の指定	なし	全TLOG共通でOFF, sec, min, hourから選択	各演算チャンネルごとにOFF, Sec, Min, Hourから選択
チャンネルのグループ化	なし	7グループに任意に割付可	なし(画面表示上のグループ化のみ)
チャンネル間統計演算	なし(かわりに任意パラメータに使用できるMax, Min, P-P, Sum, Ave関数を用意)	CLOG関数による指定グループ内のMax, Min, P-P, Sum, Ave演算	なし
表示のホールド	なし	HOLD()関数による任意チャンネルの表示保持	なし
測定チャンネルの移動平均	なし	各測定チャンネルごとに2~64サンプルで設定可	各測定チャンネルごとに2~16サンプルで設定可(ただし中速モデルのみ)
演算チャンネルの移動平均(長時間移動平均)	なし	なし	各演算チャンネルごとに、間隔1秒~1時間、サンプル数1~64、で設定可

T0702.EPS

7.1 事例：測定チャンネルの状態による動作制御

MX100用横河純正PCソフトウェアでは、測定値やアラーム状態によって記録動作やマーク追加の制御を行うことができます。ここでは、そのいくつかの事例を紹介します。ただし、通信エラーなどによって、動作のタイミングが遅れたり、検出されずに動作が行われなかったりする場合がありますので、そのことを十分ご理解いただいたうえで活用してください。とくに、MX100には強力な通信リカバリ機能があるため、通信障害が短時間のうちに復帰すれば測定データは取りこぼし無く記録されますが、その間の演算は通信障害前の測定データを繰り返し用いることになり、通信リカバリした測定データに対する演算処理は行われないことに、ご注意ください。

なお、いくつかの事例で登場するStartRec()関数およびStopRec()関数を機能させるためには、PCソフトウェア上の“収集条件”画面にて「記録開始条件」「記録終了条件」を「演算」に設定したうえで実際に「記録開始」し、記録を“待機中”の状態にしておく必要があります。また、実際に記録されるデータ範囲は、StartRec()やStopRec()などのイベント関数が評価（実行）されたタイミングと完全に同一ではなく、若干（1秒以内）前後します。

・外部接点入力（レベル動作）による記録の制御

例えば、CH00001を記録開始/終了のための接点入力信号として用い（測定レンジをDIにする）、「CH00001の接点がONの間は記録、OFFの間は待機」というような事例です。

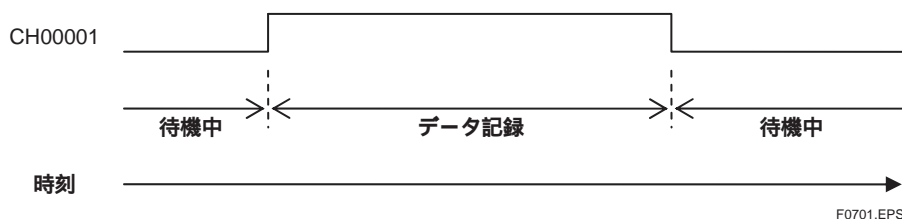


図7-1 外部接点入力（レベル動作）による記録の制御

実現のための考え方はさまざまにありますが、例えば「CH00001の値が0.5より大きかったら記録開始、そうでなければ記録停止」とすると、次式ようになります。

$$CH99001 = \text{IsNaN}(\text{ch}(1)) ? 0 : \text{ch}(1) > 0.5 ? \text{StartRec}() : \text{StopRec}() \quad (7.1)$$

または

$$CH99001 = (\text{IsNaN}(\text{ch}(1)) ? 0 : \text{ch}(1)) > 0.5 ? \text{StartRec}() : \text{StopRec}() \quad (7.2)$$

この(7.1)、(7.2)式とも、最初でIsNaN()関数を使っていますが、これが、通信を考慮したテクニックになります。(7.1)式では、CH00001の値が[NaN]になる可能性を考慮し、もし[NaN]だった場合には何もしないようにしています。(7.2)式では、CH00001の値が[NaN]だった場合には0とみなして処理を行っています。これらを本章冒頭「通信の介在を考慮したテクニック（手法）」の表7-1と見比べると、ご紹介したテクニック（手法）をそのまま適用していることがわかれると思います。

補足ですが、この(7.1)、(7.2)式とも、接点がONの間中StartRec()関数が、OFFの間中StopRec()関数が、それぞれ評価され続けます。本来であればONからOFF、もしくはOFFからONへの変化を検出するような式にするべきですが、実はStartRec()関数やStopRec()関数は、すでにその状態にあるときに評価されても無視されるので、この式で問題ありません。

・外部接点入力によるマークの挿入

例えば，CH00001をマーク挿入のための接点入力信号として用い（測定レンジをDIにする），「CH00001の接点がONになったとき“a”というマークを打つ」というような事例です。

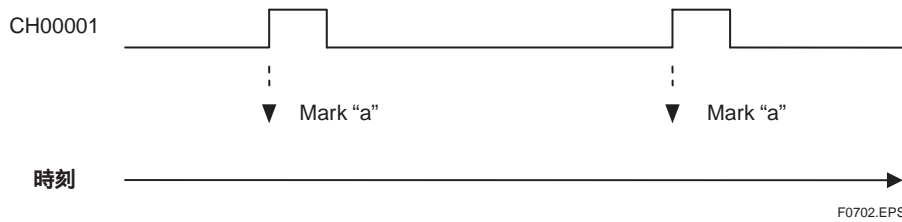


図7-2 外部接点入力によるマークの挿入

実現のための考え方はさまざまにありますが，少なくとも「CH00001の値が0.5を超えたら“a”というマークを打つ」という考えから“CH99001 = ch(1)>0.5?Mark(“a”):0”という演算式を導いてはいけません。考え方は間違っていないが，その演算式では「CH00001の値が0.5より大きかったら“a”というマークを打つ」になり，接点がONの間ずっと“a”というマークが打たれ続けてしまいます。「超えたら」という判断をするためには，前回と今回とでの変化を検出することが必要になります。

例えば「CH00001の値が前回の値より大きかったら“a”というマークを打つ」と考えると，以下のように実現できます。

$$CH99001 = \text{IsNaN}(ch(1)) ? \text{prech}(99001) : ch(1) \tag{7.3}$$

$$CH99002 = \text{IsNaN}(\text{prech}(99001)) ? 0 : \text{prech}(99001) < ch(99001) ? \text{Mark}("a") : 0 \tag{7.4}$$

少々難しくみえるかもしれませんが，これは，通信を考慮したテクニックが2つ含まれているためです。1つは，prech(1)とch(1)との比較では必ずしも変化を検出できないため，いったん演算チャンネルに取り込んでからprech(99001)とch(99001)との比較にしていること，もう1つは，先ほども挙げた[NaN]への対処です。テクニックは2つ分ですが，よく見ると，本章冒頭「通信の介在を考慮したテクニック（手法）」の表7-1の手法をそのまま適用していることがわかるとおもいます。

なお，(7.3)，(7.4)式を工夫すると，1本の演算式にまとめることもできます。CH00001の値を演算の最後で自身のチャンネルに放り込む形にすると，自身とCH00001との比較をすればよくなり，次式のように実現できます。

$$CH99001 = \text{IsNaN}(ch(1)) ? \text{prech}(99001) : (\text{IsNaN}(\text{prech}(99001)) ? 0 : \text{prech}(99001) < ch(1)) ? \text{Mark}("a") : 0, ch(1) \tag{7.5}$$

・外部接点入力（エッジ動作）による記録の制御

例えば，CH00001 / CH00002をそれぞれ，記録開始 / 記録停止のための接点入力信号として用い（測定レンジをDIにする），「CH00001の接点がONになったとき記録開始，CH00002で同様に記録停止」というような事例です。

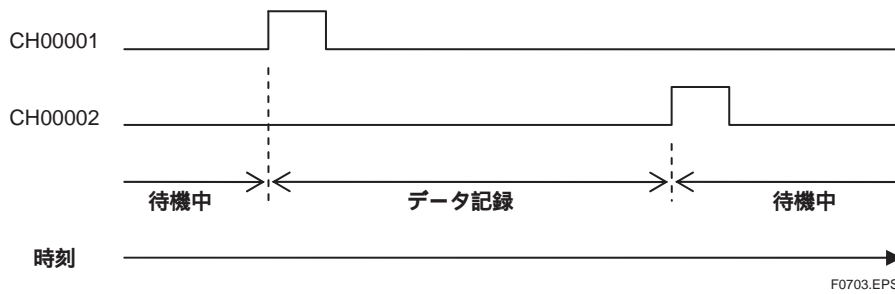


図7-3 トリガ入力（エッジ動作）による記録の制御

先にご紹介した「外部接点入力によるマークの挿入」と同じ方法で実現できます。(7.5)式を応用すると，次式のようになります。

$$CH99001 = \text{IsNaN}(ch(1)) ? \text{prech}(99001) : (\text{IsNaN}(\text{prech}(99001)) ? 0 : \text{prech}(99001) < ch(1) ? \text{StartRec}() : 0, ch(1)) \tag{7.6}$$

$$CH99002 = \text{IsNaN}(ch(2)) ? \text{prech}(99002) : (\text{IsNaN}(\text{prech}(99002)) ? 0 : \text{prech}(99002) < ch(2) ? \text{StopRec}() : 0, ch(2)) \tag{7.7}$$

・測定値レベルによる記録の制御

例えば，「CH00001の値が10を超えたら記録を開始する」というような事例です。

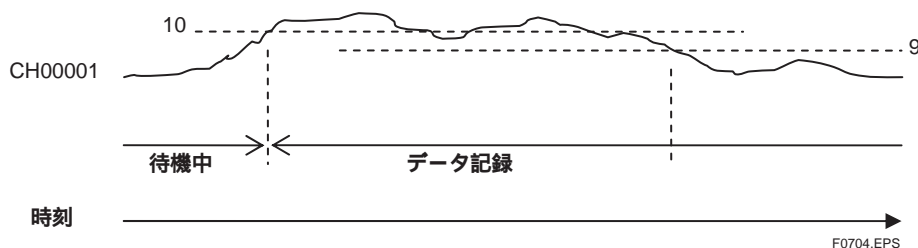


図7-4 測定値レベルによる記録の制御

先にご紹介した「外部接点入力（レベル動作）による記録の制御」と同じ方法で実現できます。(7.1)式を応用すると，次式のようになります。

$$CH99001 = \text{IsNaN}(ch(1)) ? 0 : ch(1) > 10 ? \text{StartRec}() : 0 \tag{7.8}$$

もし記録停止も行いたい場合は，例えば(7.8)式にStopRec()関数および条件判断を追加し，次のように実現できます。

$$CH99001 = \text{IsNaN}(ch(1)) ? 0 : (ch(1) > 10 ? \text{StartRec}() : ch(1) < 9 ? \text{StopRec}() : 0) \tag{7.9}$$

この例では，CH00001の測定値が9を下回ったときに記録を停止します。

・測定値レベル超過によるマークの挿入

例えば、「CH00001の値が10を超えたら“a”というマークを挿入する」というような事例です。

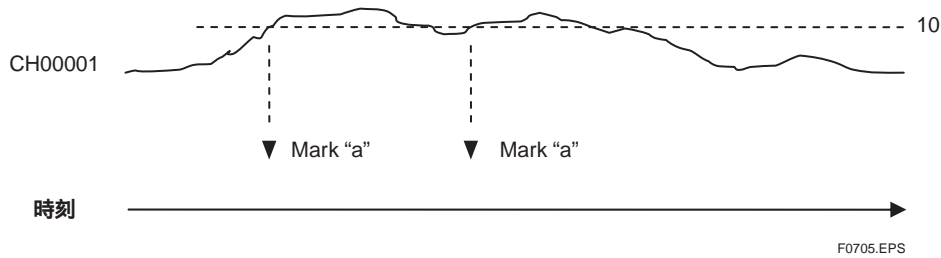


図7-5 測定値レベル超過によるマークの挿入

事例の文章は、先にご紹介した「測定値レベルによる記録の制御」に似ていますが、(7.8)式ではなく、おなじく先にご紹介した「外部接点入力によるマークの挿入」を応用して実現します（理由は、先の説明を参照してください）。(7.5)式を応用し、条件判別を“前回参照値が10未満かつ今回参照値が10以上”とすればよいので、次式のようにになります。

$$CH99001 = \text{IsNaN}(\text{ch}(1)) ? \text{prech}(99001) \\ : (\text{IsNaN}(\text{prech}(99001)) ? 0 : (\text{prech}(99001) < 10) \&\& (\text{ch}(1) \geq 10) ? \text{Mark}(\text{"a"}): 0, \text{ch}(1)) \quad (7.10)$$

・測定チャンネルのアラームIn / Outによるマークの挿入

例えば、「CH00001のアラームレベル1の状態をウォッチし、アラームのIn / Outにあわせてコメントを入力させる」というような事例です。

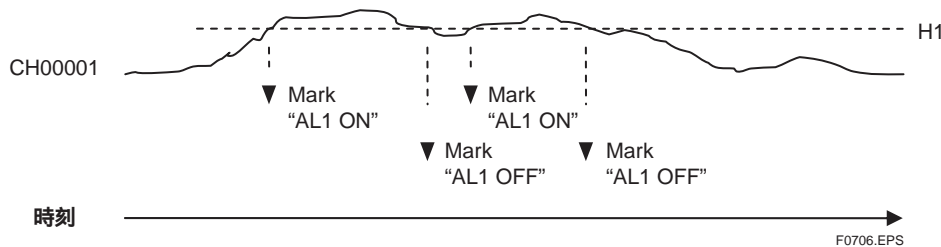


図7-6 測定チャンネルのアラームIn/Outによるマークの挿入

先にご紹介した「外部接点入力によるマークの挿入」を応用して実現できます。順序演算子“,”を用いてアラームIn時の処理とOut時の処理を1本の演算式中にまとめられますので、次式のようにになります。

$$CH99001 = \text{IsNaN}(\text{prech}(99001)) ? 0 \\ : \text{prech}(99001) < \text{alarm}(1,1) ? \text{Mark}(\text{"AL1 ON"}) \\ : \text{prech}(99001) > \text{alarm}(1,1) ? \text{Mark}(\text{"AL1 OFF"}): 0 \\ , \text{alarm}(1,1) \quad (7.11)$$

基本的に(7.5)式の応用ですが、alarm()関数は[NaN]を返すことが無いので、IsNaN()関数による処理が一部省略されています。

7.2 事例：イベントを数える

前節「7.1 事例：測定チャンネルの状態による動作制御」にて、測定値やアラーム状態を検出する事例を紹介しました。この検出テクニックを応用してイベントを数えると、更にアプリケーションの幅が広がります。ここでは、そのいくつかの事例を紹介します。なお、やはり通信エラーなどの場合には、数えるタイミングが遅れたり、数え漏れしたりという可能性もあるので、そのことを十分理解したうえで使用してください。

・外部接点入力（エッジ）の回数を数える / パルス数を数える

例えば、CH00001をパルス信号として用い（測定レンジをDIにする）、「CH00001に入力される立ち上がりパルスの回数をカウントアップする」というような事例です。先に第7.1節でご紹介した「外部接点入力によるマークの挿入」を応用することで実現できます。

(7.3)，(7.4)式をベースにすると、CH99002にて「CH99001が0から1へ変化したらCH99002の値をカウントアップ、それ以外ではCH99002の値を保持」とすればよいので、次の2式が導かれます。

$$CH99001 = \text{IsNaN}(\text{ch}(1)) ? \text{prech}(99001) : \text{ch}(1) \tag{7.12}$$

$$CH99002 = \text{IsNaN}(\text{prech}(99001)) ? \text{ch}(99002) : \text{prech}(99001) < \text{ch}(99001) ? \text{ch}(99002) + 1 : \text{ch}(99002) \tag{7.13}$$

注意点として、前述のとおり通信エラーなどでカウントロスが発生することのほかに、DIの測定周期と通信周期とが設定上で一致していたとしても、時刻が同期していないためサンプルロスが発生する可能性があり、パルス幅はHigh / Lowとも演算周期の3倍は必要

PCソフトウェアの再スタートで積算値がゼロリセットされる

ことにも注意が必要です。

なお、この事例では(7.5)式のように1本の演算式にまとめることはできません。

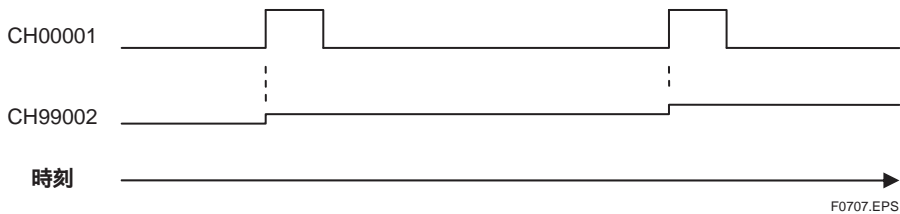


図7-7 外部接点入力（エッジ）の回数を数える / パルスを数える

・測定チャンネルのアラーム回数を数える

例えば、「CH00001のアラームレベル1の状態をウォッチし、アラームInした回数をカウントアップする」というような事例です。先に第7.1節でご紹介した「外部接点入力（エッジ）の回数を数える / パルスを数える」と同じ考え方で、次の2式が導かれます。

$$CH99001 = \text{alarm}(1,1) \tag{7.14}$$

$$CH99002 = \text{IsNaN}(\text{prech}(99001)) ? \text{ch}(99002) : \text{prech}(99001) < \text{ch}(99001) ? \text{ch}(99002) + 1 : \text{ch}(99002) \tag{7.15}$$

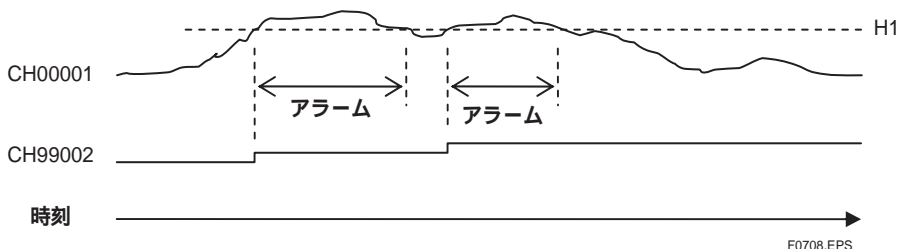


図7-8 測定チャンネルのアラーム回数を数える

7.3 事例：時間の使い方

MX100用横河純正PCソフトウェアでは、DARWINやDAQSTATIONなどの従来機種がハードウェアに内蔵していた演算機能に比べ、豊富な時間関係の処理を備えています。ここでは、それらを活用した事例のいくつかを紹介します。

なお、いくつかの事例で登場するStartRec()関数およびStopRec()関数を機能させるためには、PCソフトウェア上の“収集条件”画面にて「記録開始条件」「記録終了条件」を「演算」に設定したうえで実際に記録を開始し、記録を“待機中”の状態にしておくことが必要です。また、実際に記録されるデータ範囲は、StartRec()やStopRec()などのイベント関数が評価されたタイミングと完全に同一ではなく、若干(1秒以内)前後します。

・不規則な時間間隔での記録

例えば、「最初のデータ取りを10:00～10:10までの10分、次は10:15～10:30までの15分、最後に10:35～10:55までの20分、という測定を毎日自動で行いたい」とします。こういった場合は、時間関数とイベント関数で実行可能です。演算式は、次のようになります。

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

CH99002 = daily(10,10)?StopRec():0 (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)

このままでも構いませんが、第7.1節の「外部接点入力(レベル動作)による記録の制御」で説明したようにStartRec()関数、StopRec()関数はレベル動作でも問題ないことを考慮して、さらに「10:00～10:10の間、10:15～10:30の間、10:35～10:55の間、のいずれかであれば記録開始、さもなければ記録停止」のように工夫すると、次のように演算式の本数を抑えることもできます。

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

・指定時刻でファイル分割する

1ファイルの時刻範囲を交代制勤務の各作業時間にあわせたいなど、連続記録を指定時刻で分割したい場合があります。DAQWORX MXLOGGERをお使いの場合は、収集条件設定によって「毎日 時 分 秒でファイル分割」という指定をすることができますが、1日の中で複数の時刻を指定することはできませんし、MX100スタンダードソフトウェアでは、そのような指定自体ができません。このような場合は、演算式で解決することが可能です。

たとえば「毎日7:00、15:00、23:00にファイル分割したい」とすると、つぎのような演算式で実現できます。

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

・月～金の昼間と土曜日の午前中のみ記録する

例えば、「平日8:00～17:00，土曜日8:00～12:00だけ記録したい」とします。残念ながら祝祭日を判断させることはできませんので，実現できるのは「月～金曜日かつ8:00～17:00，もしくは土曜日かつ8:00～12:00ならば」というところまでになります。

```
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 事例：統計演算

従来機種がハードウェアに内蔵していた統計演算には，機種にもよりますが，時系列の統計演算と，チャンネル間の統計演算があります。MX100用横河純正PCソフトウェアでは，時系列の統計演算を実行する方法としてTLOG関数を用意していますが，その使い方は従来機種のハードウェア演算とはやや異なっています（本章冒頭「他機種との比較」参照）。本節ではその差異に注意しながら，いくつかの事例を紹介します。

・TLOG関数による積算

「CH00001の流量 [L/min.] を積算し，毎正時でリセットする」という，シンプルかつ実用的な事例を考えます。従来機種のハードウェア演算と比べて注意すべき要点は，下記のとおりです。

絶対時間タイマがないので，演算機能の時間関数を活用する

積算単位の処理機能が無いので，演算周期との関係を考えて定数倍する

リセットは全TLOG演算一括なので，リセット間隔の異なる複数の時系列統計処理が必要な場合は，TLOG関数で実現できない

演算周期が100msec.でTLOG演算がこの1点のみという場合，この事例は次のように実現できます。(7.26)式では，毎正時にTLOG演算をリセット（正確には毎正時の直後にリセット）しています。(7.27)式では，CH00001の単位をtlogsum()関数にて単純積算した後，積算単位を“分あたり”から“100msec.あたり”に変換しています。CH99002が積算値です。

```
CalcInt = 0.1 (7.25)
```

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

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

以上で積算演算を実現したことになりますが，実用上では次のような「積算ローカット」を行い，流量センサ（変換器）出力のゼロ近傍のノイズ影響を省くのが一般的です。この例ではローカットポイントを4L/min.としています。CH99003が積算値です。

```
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)
```

なお，TLOG演算は対象チャンネルの周期で実行されるため，対象チャンネルの周期が異なると，統計処理の精度に差異が生じます。今回の事例では，例えばCH00001のサンプリング周期が10msecだった場合，(7.27)式では10msec周期で実行されたTLOG演算を100msec周期で参照することになりますが，(7.32)式はTLOG演算自体が100msec周期で実行されます。ご注意ください。

・TLOG関数によらない積算

先の「TLOG関数による積算」の事例を，TLOG関数を使わずに実現します。ResetTLog()関数に頼ることができないため，時間関数の状態により，前回値に加算するか0に加算するかを切り替えるようにします。次のCH99002が積算値です。

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)

(7.36)式にて算術関数sum()を使っているのは，本章冒頭の表7-1でご紹介した，通信の介在を考慮したテクニックです。とはいえ，(7.35)式ではCH99001が[NaN]になり得ないので，実はsum()を使わずに単項演算子“+”で加算しても大丈夫です。ですが，もし(7.35)式を“CH99001 = ch(1)>=LowCut?ch(1):0”とすると，同じに見えてもCH99001が[NaN]になる可能性が出てくるので，sum()関数で加算しないと問題が生じます。なお，時間関数の参照よりも積算演算が先になるように記述しているのは，時間関数の動作よりリセットタイミングを1周期ぶん遅らせて，毎正時の直後にリセットさせるためです。

・TLOG関数によらない時系列平均演算

TLOG関数が使える場合は，先の「TLOG関数による積算」の事例と同様に実現できます。しかしTLOG関数が使えない場合は，「平均をとる」という演算を独自に構築せねばなりません。具体的には，時系列の合計と個数とをそれぞれ算出して割り算をすることになりますが，測定チャンネルへの参照値が[NaN]になった場合にはsum()関数が加算を見送るため，統計データの個数が演算回数とは異なってくることに，注意が必要です。「CH00001の平均温度を算出し，毎正時でリセットする」という事例は，以下のように実現できます。なお(7.39)式では，“CH00001が[NaN]なら1,さもなければ0”というIsNaN()関数の論理を否定演算子“!”によって反転させ(つまり，[NaN]なら0,さもなければ1)，それを足しこむことで時系列の個数を計算しています。ちょっとしたテクニックです。

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 事例：操作・出力関係（Release 2以降）

MX100スタイルナンバーS2ではアナログ出力機能が加わり，MX100用横河純正PCソフトウェアRelease 2以降では，これに応じた機能が追加されています。また，新たにマニュアル操作に連動するManualDO()関数およびManualAO()関数が追加され，使い勝手の向上が図られています。ここでは，それらを活用した事例のいくつかを紹介します。

・PC操作によって数値を保持する

何らかの操作によって，測定値や演算値を保持させたい場合があります。一番シンプルなのは「CH00011の接点入力ONの間は，CH00001の値を保持させる」という事例であり，これはスタイルナンバーS1のMX100とRelease 1のMX100用横河純正PCソフトウェアでも実現できます。

$$CH99001 = !ch(11)?ch(1):ch(99001) \tag{7.42}$$

少々乱暴な演算式かもしれませんが，これは「CH00011の値が[NaN]（不定）もしくは0（OFF）ならばCH00001の値を取り入れ，1（ON）ならば値を保持しなさい」という考えになっており，わざとIsNaN()関数を使わない形で実現したものです。ちょっとしたテクニックです。

さて，ここで本当にご紹介したいのは(7.42)式ではなく，この事例を「PCソフトウェア上のManual DO 1がONの間は」とするテクニックです。その実現にはRelease 2から追加されたManualDO()関数を使用するのですが，Manual DOボタンはシステム内にマニュアルDOが設定されていないと現れません。したがって，MX100スタンダードソフトウェアでは実際にDOモジュールを装着したハードウェアのみでの実現になります。一方，DAQWORX MXLOGGERの場合は実在しないダミーのユニットでも登録することができるため，実際にDOモジュールが無くても，登録ユニット数に余裕があればこのテクニックを活用することができます。

$$CH99001 = ManualDO(1)?ch(99001):ch(1) \tag{7.43}$$

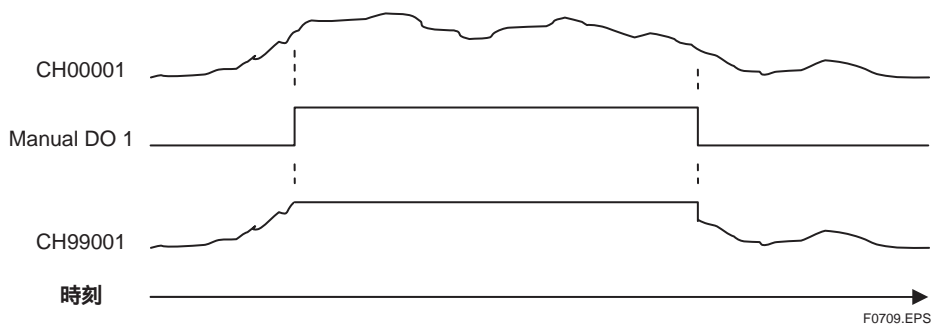


図7-9 PC操作によって数値を保持する

・アナログ出力の緊急停止

演算結果をAOモジュールに出力しているアプリケーションでは，手動介入で緊急停止（出力をある指定値にする）できるようにしておきたい場合があります。この事例でも先の「PC操作によって数値を保持する」同様，接点入力による手動介入の場合と，Manual DO操作による手動介入の場合，どちらも実現できます。例えばCH99001にてCH00021のアナログ出力値が計算されているとして，それをManual DO 1にて緊急停止させる（例えば0%にする）には，次のようにします。

CH00021：動作 = 伝送出力，参照チャネル = CH99902

CH99002：スパン = 0 ~ 100

CH99001 = { 通常出力用の演算式 } (7.44)

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

また、(7.45)式をもう少し工夫して、次の(7.46)式のように出力値の固定先をプリセット値にしたり、

CH00021 : 動作 = 伝送出力, 参照チャンネル = CH99902, エラー時出力動作 = プリセット値
CH99002 = ManualDO(1)?NaN:ch(99001) (7.46)

次の(7.47)式のように出力値をマニュアル操作できるような手動介入も可能です。

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

なお、Manual DOボタンやManual AOパネルを使用するための条件については、先の「PC操作によって数値を保持する」を参照してください。

・アナログ値のパターン出力

DAQWORX MXLOGGER Release 2では、AOモジュール向けのパターン出力機能が提供され、定値出力やランプ出力（直線増加、直線減少）の組合せをアナログ出力させることが可能です。複雑なパターン出力はぜひDAQWORX MXLOGGERのご使用をおすすめしますが、簡単なパターン出力ならば、MX100スタンダードソフトウェアの演算機能で実現することもできます。

例えば「CH00021のアナログ出力で、演算開始後しばらくは0%を出力し、5分経過ごとに順次25%、50%、75%、100%と出力を上げていき、25分経過以降は0%出力に戻す」という事例は、次のように実現できます。

CH00021 : 動作 = 伝送出力, 参照チャンネル = CH99905

CH99003 : スパン = 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)

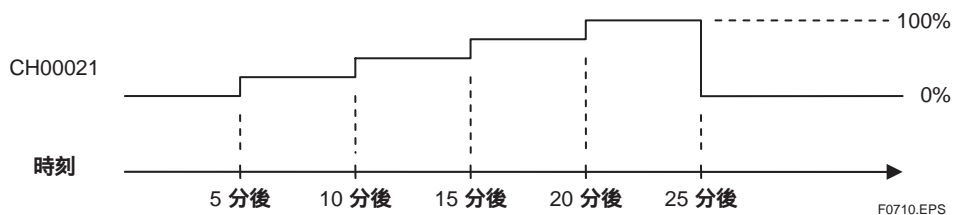


図7-10 アナログ値のパターン出力

出力パターンの計算に(7.51)～(7.53)の3本の演算式を使用していますが、これは演算式の文字数制限（最長127文字）の都合で1本の式に納まらなかったためです。パターンが簡単であれば（今回の事例では、例えば20分後にパターン終了であれば）、1本で済みます。

7.6 事例：その他

・8bit DI入力のデコード（数値としての解釈）

装置の状態を表す数値が、複数本の接点信号（一般的にはオープンコレクタ）を用いて2進数表現で出力されていることがあります。例えば、装置の異常状態を表すエラー番号、試験装置のテストパターンにおけるパターン番号やセグメント番号などがこれにあたりますが、この接点信号をMXに取り込みんで演算機能を活用することで、元の数値として記録に残すことができます。

例えば数値が8bitの2進数で表現されており、CH00001～CH00008に入ってくる信号が順に、L/L/L/L/L/L/H/Lのときは“2”，L/H/L/L/H/H/L/Hのときは“77”，というようなデコード（演算）をさせるための基本式は、

$$\text{CH99001} = \text{poly}(2, \text{ch}(1), \text{ch}(2), \text{ch}(3), \text{ch}(4), \text{ch}(5), \text{ch}(6), \text{ch}(7), \text{ch}(8)) \quad (7.54)$$

です。このままでも十分に使える場面は多いですが、さらに工夫して、測定値が[NaN]だったときには演算結果を保持したり（(7.55)、(7.56)式のCH99002）、数値変化タイミングにおけるビットエラーを回避すること（(7.55)～(7.57)式のCH99003）も可能です。

$$\text{CH99001} = \text{ch}(1) + \text{ch}(2) + \text{ch}(3) + \text{ch}(4) + \text{ch}(5) + \text{ch}(6) + \text{ch}(7) + \text{ch}(8) \quad (7.55)$$

$$\text{CH99002} = \text{IsNaN}(\text{ch}(99001)) ? \text{ch}(99002) : \text{poly}(2, \text{ch}(1), \text{ch}(2), \text{ch}(3), \text{ch}(4), \text{ch}(5), \text{ch}(6), \text{ch}(7), \text{ch}(8)) \quad (7.56)$$

$$\text{CH99003} = \text{IsNaN}(\text{prech}(99002)) ? \text{ch}(99003) : \text{ch}(99002) == \text{prech}(99002) ? \text{ch}(99002) : \text{ch}(99003) \quad (7.57)$$

・移動平均（長時間移動平均）

従来機種がハードウェアに内蔵していた移動平均機能には、入力信号に乗っているノイズを抑えるための移動平均と、演算結果のふらつきを抑えるための長時間移動平均（DAQSTATIONシリーズなど）があります。前者については、一般的にはハードウェア側にデジタルフィルタをつけることで目的を解決でき、MX100でもそうになっています。しかし後者についてはMX100では手段を用意していませんので（本章冒頭「他機種との比較」の表7-2参照）、必要であれば演算式を駆使して実現することになります。

(7.58)式～(7.63)式は「CH99001の演算結果を1分間隔で10分間（10データ）分移動平均をとる」の事例です。おわかりのとおり、演算式の本数を多く必要とします。

タイマ1：動作 = エッジ，周期 [sec] = 60，On時間 [sec] = 0

$$\text{CH99001} = \{ \text{ターゲットとなる演算} \} \quad (7.58)$$

$$\text{CH99011} = \text{timer}(1) ? \text{prech}(99012) : \text{prech}(99011) \quad (7.59)$$

$$\text{CH99012} = \text{timer}(1) ? \text{prech}(99013) : \text{prech}(99012) \quad (7.60)$$

：

$$\text{CH99019} = \text{timer}(1) ? \text{prech}(99020) : \text{prech}(99019) \quad (7.61)$$

$$\text{CH99020} = \text{timer}(1) ? \text{ch}(99001) : \text{prech}(99020) \quad (7.62)$$

$$\text{CH99021} = \text{ave}(\text{ch}(99011), \text{ch}(99012), \dots, \text{ch}(99019), \text{ch}(99020)) \quad (7.63)$$

・チャンネル間統計演算

MX100用横河純正PCソフトウェアでは、チャンネル間の統計演算を行わせるための関数を用意していません（本章冒頭「他機種との比較」の表7-2参照）。ですが、かわりに豊富な算術関数が用意されているので、同等のことを簡単に実現することができます。次の(7.64)式ではCH00001～CH00007の平均が、(7.65)式ではCH00001，CH00011，CH00021の最大差が、それぞれ計算されます。

$$\text{CH99001} = \text{ave}(\text{ch}(1), \text{ch}(2), \text{ch}(3), \text{ch}(4), \text{ch}(5), \text{ch}(6), \text{ch}(7)) \quad (7.64)$$

$$\text{CH99002} = \text{pp}(\text{ch}(1), \text{ch}(11), \text{ch}(21)) \quad (7.65)$$

・任意チャンネルのORアラーム

1点のDOを用い、複数のアラーム条件のいずれかの成立にあわせて警報出力したいという場合があります。MXLOGGERでは、デジタル出力の条件としてアラームの参照先チャンネルを範囲指定できますが、飛び飛びのチャンネルを指定できないなど、自由に選ぶことはできません。また、MX100スタンダードソフトウェアでは範囲指定をすること自体ができません。

このような場合、演算によってORアラームを実現することができます。例えば、「CH00001のアラームレベル1とCH00011のアラームレベル2のいずれかの成立でCH00031の接点出力をONにする」という事例は、次のように実現できます。

CH00031：動作 = アラーム，参照先 = CH99001 レベル1，保持 = 非保持

CH99001：アラーム1タイプ = High，アラーム1設定値 = 0.5，アラーム1ヒステリシス = 0

$$\text{CH99001} = \text{alarm}(1,1) \parallel \text{alarm}(11,2) \quad (7.66)$$

ただし、PCソフトウェア上で演算した結果であるため、ハードウェアでの警報に比べて動作が遅れること、また、通信異常のときには正常に動作しないことに、十分ご注意ください。また、この方法で出力保持をしようとすると、新着アラームを検出できないなどの不都合が生じるため、出力保持をしたい場合はもっと高度なテクニックが必要となります（ここでは省略します）。

7.7 演算構築上の注意点および回避テクニック（解説）

本章冒頭「通信の介在を考慮したテクニック（手法）」では、測定を行うMX100本体と演算を行うPCとの間に通信が介在することによる注意点を簡単にご説明し、回避テクニック（手法）だけをご紹介しました。本節では、その注意点とテクニックを深くご理解いただくため、具体例を挙げて詳しく解説します。

演算構築上で気をつけるべき注意点の概略は、次表のとおりです。

表7-3 演算構築上で気をつけるべき注意点

No.	注意点	要因	気をつける場面	手法
1	測定チャンネルへの参照結果が[NaN]になることがある	通信断やモニタ開始直後でのデータ不着	測定値に基づく動作条件判別、TLOG関数によらない測定値加算、など	IsNaN()関数の使用、sum()関数の活用
2	測定チャンネルへのprech()関数の参照結果がch()関数の前回値とは限らない	通信に起因する測定データの到着遅れ	測定チャンネルの前回値比較（接点等の変化検出）など	いったん演算チャンネルに取り込む

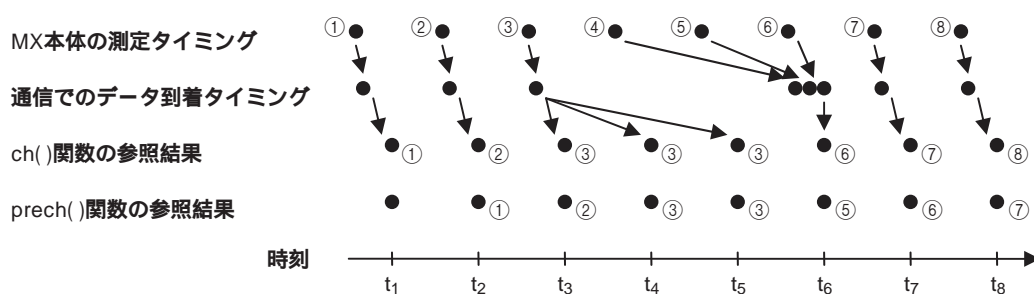
T0703.EPS

これらの注意点をよくご理解いただくため、まず「測定値はどのように演算式に取り込まれるか」および「測定チャンネルへの参照結果が[NaN]になる状況」について簡単にご説明し、続いて具体的な例を挙げてテクニックを解説します。

・測定値はどのように演算式に取り込まれるか

測定値を演算式に取り込む手段として、ch()関数およびprech()関数が用意されています。しかし、演算式がPCにて定周期で実行されるのに対し、測定値も定周期で取得できるとは限りません。これは、MX本体とPCとの間に通信が介在し、MX本体の測定とPCソフトウェア上の演算実行とが同期していないためです。したがって、MX本体で測定されるデータの時系列とch()関数にて演算式に取り込める時系列とは、必ずしも一致しないことになり、注意が必要です。

測定データの到着遅れによってch()関数およびprech()関数の参照がどのように影響されるかを、図7-11および表7-4に示します。



F0711.EPS

図7-11 測定データの到着遅れによるch()関数/prech()関数への影響

表7-4 測定データの到着遅れによるch()関数/prech()関数への影響（解説）

時刻	到着済みデータ	ch()関数の参照	prech()関数の参照
t ₂	①②	時刻t ₂ に最も近い②を参照	時刻t ₁ に最も近い①を参照
t ₃	①②③	時刻t ₃ に最も近い③を参照	時刻t ₂ に最も近い②を参照
t ₄	①②③	時刻t ₄ に最も近い③を参照（④は未着）	時刻t ₃ に最も近い③を参照
t ₅	①②③	時刻t ₅ に最も近い③を参照（④⑤は未着）	時刻t ₄ に最も近い③を参照（④は未着）
t ₆	①②③④⑤⑥	時刻t ₆ に最も近い⑥を参照	時刻t ₅ に最も近い⑤を参照
t ₇	①②③④⑤⑥⑦	時刻t ₇ に最も近い⑦を参照	時刻t ₆ に最も近い⑥を参照

T0704.EPS

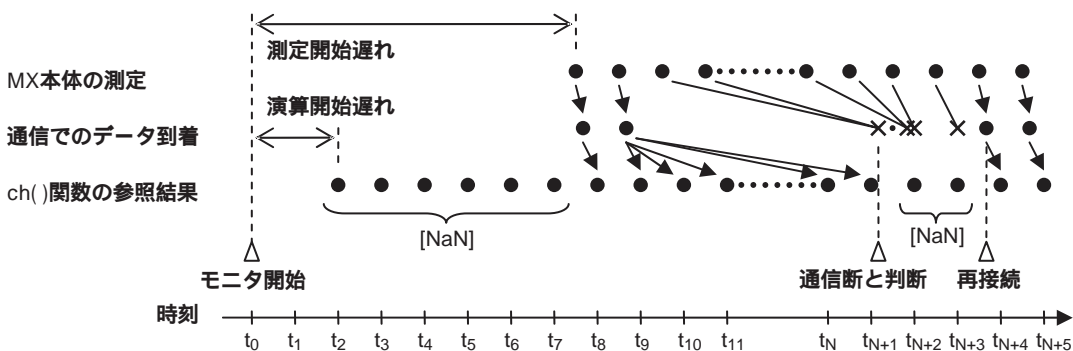
表7-4に示したように，ch()関数は演算実行時刻に最も近いデータを参照し，prech()関数は前回の演算実行時刻にもっとも近いデータを参照します。ところが，通信の影響によってその時点その時点での到着済みデータの状況が変わるため，ch()関数が参照する時系列データとprech()関数が参照する時系列データとは，必ずしも一致しなくなります。すべての場合に問題となるわけではありませんが，測定値の変化を確実に検出したいような場面では，対策が必要となります。このような場面での対策の基本は，

測定値をまず演算チャンネルに取り込んでから，その演算チャンネルの変化を検出することです。後に「例3）CH0001の接点がONになったとき“a”というマークを打つ」で具体例を挙げて詳しく解説します。

・測定チャンネルへの参照結果が[NaN]になる状況

ch()関数およびprech()関数の参照先チャンネルに参照すべき測定データが無いとき，測定値として[NaN]（不定な値またはエラー値）を使用します。PCソフトウェアが通信断状態を検出している場合はもちろんですが，実はモニタリング開始直後でも，演算開始に比べて測定開始が遅れるために，測定データが無い状態で演算が行われ，値として[NaN]が使われることに注意が必要です。

モニタリング開始直後および通信断によってch()関数およびprech()関数の参照がどのように影響されるかを，図7-12および表7-5に示します。



F0712.EPS

図7-12 モニタリング開始直後および通信断による[NaN]の参照

表7-5 モニタリング開始直後および通信断による[NaN]の参照（解説）

時刻	演算による測定データの参照状況
$t_0 \sim t_1$	モニタリング開始直後で演算未開始
$t_2 \sim t_8$	演算は開始されたが、まだ測定データが無いため[NaN]を使用
$t_9 \sim t_{10}$	測定も開始され、到着する測定データを参照
$t_{11} \sim t_{N+1}$	測定データが到着せず、最も近い t_{10} のときの測定データを参照
$t_{N+2} \sim t_{N+3}$	通信断と判断され、測定データが無いため[NaN]を使用
$t_{N+4} \sim t_{N+5}$	通信が再接続され、到着する測定データを参照

T0705.EPS

表7-5に示したように、次のような場合には測定チャンネルへの参照結果が[NaN]となります。

通信断状態を検出しているとき

モニタリング開始直後でまだ測定データが存在しないとき

測定値が[NaN]となることで、大抵の演算結果も[NaN]になります。すべての場合に問題となるわけではありませんが、用途によっては演算結果が[NaN]となって困る場面もあり、対策が必要となります。次より2つの具体例を挙げて詳しく解説していきます。

・例1) CH00001の値が10を超えたら記録を開始する

「測定チャンネルへの参照値が[NaN]となることがある」への対策として、先の第7.1節にてご紹介した「測定値レベルによる記録の制御」の事例を用い、問題ある演算式と対策後の演算式を表7-6に示します。

表7-6 「測定チャンネルへの参照値が[NaN]となることがある」への対策例その1

CH00001の値が10を超えたら記録を開始する		
対策前の演算式（問題あり）	CH99001 = ch(1)>10?StartRec():0	(7.67)
対策後の演算式1	CH99001 = (IsNaN(ch(1))?0:ch(1))>10?StartRec():0	(7.68)
対策後の演算式2	CH99001 = IsNaN(ch(1))?0:ch(1)>10?StartRec():0	(7.8)

T0706.EPS

「CH00001の値が10を超えたら記録開始」という文章をそのまま演算式にすると、条件演算子“?”を使って(7.67)式のような記述になります。しかしこの式では、実用上問題があります。それは、ch(1)の値が[NaN]となる場合があるからです（先の「測定チャンネルへの参照結果が[NaN]になる状況」参照）。ch(1)の値が[NaN]となると、“ch(1)>10”という比較演算の結果も[NaN]となります。論理演算子や条件演算子は、0という値のみを“偽”として扱い、[NaN]を含め0以外の値をすべて“真”とみなすため、対策前の記述では、ch(1)の値が[NaN]となったときにも記録が開始されてしまいます。

この問題への対策として、次の2つの方法を考えます。

1. ch(1)の値が[NaN]だったときには、ch(1)のある代替値におきかえてから10と比較させる
2. ch(1)の値が[NaN]だったときには、比較演算自体を行わない

表7-6に挙げた対策は、これらの考えに基づくものです。1の方法では、問題のある(7.67)式において“ch(1)”の部分“IsNaN(ch(1))?0:ch(1)”におきかえることで、(7.68)式が導かれます。2の方法では、問題のある(7.67)式全体を“IsNaN(ch(1))?0:{元の演算式}”とおおうことで、(7.8)式が導かれます。

対策前後の式それぞれについて、CH00001への参照値が[NaN]となったときにCH99001がどのように計算されていくか、演算式評価の流れを表7-7に示します。対策前の演算式ではStartRec()関数が評価されてしまい記録が開始されてしまいますが、対策後の演算式ではStartRec()の評価に至らないことがわかります。

表7-7 「測定チャンネルへの参照値が[NaN]となることがある」への対策例その1（演算式評価の流れ）

対策前の演算式	対策後の演算式1	対策後の演算式2
CH99001 = <u>ch(1)>10?StartRec():0</u> = <u>[NaN]>10?StartRec():0</u> = <u>[NaN]>10?StartRec():0</u> = <u>[NaN]?StartRec():0</u> = <u>[NaN]?StartRec():0</u> = <u>StartRec()</u> = <u>StartRec() = I</u>	CH99001 = (IsNaN(<u>ch(1)</u>)?0:ch(1))>10?StartRec():0 = (IsNaN(<u>[NaN]</u>)?0:ch(1))>10?StartRec():0 = (IsNaN(<u>[NaN]</u>)?0:ch(1))>10?StartRec():0 = (<u>I</u> ?0:ch(1))>10?StartRec():0 = (<u>1?0:ch(1)</u>)>10?StartRec():0 = <u>0</u> >10?StartRec():0 = <u>0?StartRec():0 = 0</u>	CH99001 = IsNaN(<u>ch(1)</u>)?0:ch(1)>10?StartRec():0 = IsNaN(<u>[NaN]</u>)?0:ch(1)>10?StartRec():0 = IsNaN(<u>[NaN]</u>)?0:ch(1)>10?StartRec():0 = <u>I</u> ?0:ch(1)>10?StartRec():0 = <u>1?0:ch(1)>10?StartRec():0</u> = <u>0</u>

表中の表現は、二重下線の部分が評価され、*太斜字の部分*におきかわったことを表す。

T0707.EPS

・例2) CH00001の値を積算する

もう一つ、「測定チャンネルへの参照値が[NaN]となることがある」への対策例を表7-8に示します。

表7-8 「測定チャンネルへの参照値が[NaN]となることがある」への対策例その2

CH00001の値を積算する		
対策前の演算式（問題あり）	CH99001 = ch(99001)+ch(1)	(7.69)
対策後の演算式	CH99001 = sum(ch(99001),ch(1))	(7.70)

T0708.EPS

この例は一般的にはtlogsum()関数を使用しますが、アプリケーションによってはTLOG関数が適当ではない場合もあります（第7.4節参照）。tlogsum()関数を用いない場合、「CH00001の値をCH99001に加算していく」と言う形で実現することになりますが、この文章をそのまま演算式にすると、算術演算子“+”を使って(7.69)式のような記述になります。しかしこの式では、実用上問題があります。それは、ch(1)の値が[NaN]となる場合があるからです（先の「測定チャンネルへの参照結果が[NaN]になる状況」参照）。ch(1)の値が[NaN]となると、“ch(99001)+ch(1)”という算術演算の結果も[NaN]となります（表示上は“INVALID”）。ch(99001)の値がひとたび[NaN]になると、それ以降“CH99001 = ch(99001)+ch(1)”という算術演算の結果はずっと[NaN]となってしまいます。

この問題への対策として、先の「例1) CH00001の値が10を超えたら記録を開始する」のようにIsNaN()関数を使用する手法もありますが、sum()関数が[NaN]を無視することを活用すれば、表7-8のように簡単に解決できます。

対策前後の式それぞれについて、CH00001への参照値が順に2.7, [NaN], 3.2となったときにCH99001がどのように計算されていくか、演算式評価の流れを表7-9に示します。対策前の演算式ではCH00001の値が[NaN]から復帰してもCH99001の値が[NaN]のままですが、対策後の演算式では[NaN]にならずに加算が続行されていることがわかります。

表7-9 「測定チャンネルへの参照値が[NaN]となることがある」への対策例その2 (演算式評価の流れ)

時刻	対策前の演算式	対策後の演算式
t _{N-1}	CH99001 = <u>ch(99001)+ch(1)</u> = 0 + 2.7 = <u>0+2.7</u> = 2.7	CH99001 = sum(<u>ch(99001),ch(1)</u>) = sum(0,2.7) = <u>sum(0,2.7)</u> = 2.7
t _N	CH99001 = <u>ch(99001)+ch(1)</u> = 2.7 + [NaN] = <u>2.7+[NaN]</u> = [NaN]	CH99001 = sum(<u>ch(99001),ch(1)</u>) = sum(2.7,[NaN]) = <u>sum(2.7,[NaN])</u> = 2.7
t _{N+1}	CH99001 = <u>ch(99001)+ch(1)</u> = [NaN] + 3.2 = <u>[NaN]+3.2</u> = [NaN]	CH99001 = sum(<u>ch(99001),ch(1)</u>) = sum(2.7,3.2) = <u>sum(2.7,3.2)</u> = 5.9

表中の表現は、二重下線の部分が評価され、*太斜字の部分*におきかわったことを表す。

T0709.EPS

なお補足ですが、演算チャンネルの初期値は0となっています。それゆえ、“CH99001 = ch(99001)+.....”という記述や“CH99001 = sum(ch(99001),.....)”という記述で、モニタリング開始時以降もしくは記録開始時以降の積算になるのです。

・例3) CH00001の接点がONになったとき“a”というマークを打つ

「測定チャンネルへのprech()関数の参照結果がch()関数の前回値とは限らない」ことへの対策として、先の第7.1節にてご紹介した「外部接点入力によるマークの挿入」の事例を用い、問題ある演算式と対策後の演算式を表7-10に示します。

表7-10 「測定チャンネルへのprech()関数の参照結果がch()関数の前回値とは限らない」への対策例

CH00001の接点がONになったとき“a”というマークを打つ		
対策前の演算式 (問題あり)	CH99001 = IsNaN(prech(1)+ch(1))?0:prech(1)<ch(1)?Mark("a"):0	(7.71)
対策後の演算式1	CH99001 = IsNaN(ch(1))?prech(99001):ch(1)	(7.3)
	CH99002 = IsNaN(prech(99001))?0:prech(99001)<ch(99001)?Mark("a"):0	(7.4)
対策後の演算式2	CH99001 = IsNaN(ch(1))?prech(99001) :(IsNaN(prech(99001))?0:prech(99001)<ch(1)?Mark("a"):0,ch(1))	(7.5)

T0710.EPS

先のご説明のとおり、「CH00001の値が前回の値より大きかったら“a”というマークを打つ」と考えることにします。この文章をそのまま演算式にし、[NaN]への対策を行うと、(7.71)式のように記述できます。しかしこの式では、実用上問題があります。それは、CH00001の値が0から1へ変化したにもかかわらず、MX本体の測定とPCソフトウェア上の演算実行とが同期していないため、prech(1)=0かつch(1)=1という状態が来るとは限らないからです(先の「測定値はどのように演算式に取り込まれるか」参照)。もしある演算実行時にprech(1)=0、ch(1)=0であり、次の演算実行時にprech(1)=1、ch(1)=1となってしまうと、CH00001の値が0から1へ変化したことを検出できません。

この問題への対策の基本はすでにご説明のとおり

測定値をまず演算チャンネルに取り込んでから、その演算チャンネルの変化を検出することですが、単に取り込むのではなく、検出すべき遷移の分類が表7-11のようになることを考慮すると、

測定開始直後の[NaN]だけは[NaN]として扱う

ひとたび実数値になったら、以降でてくる[NaN]は反映させずに前回値を保持するような取り込み方の工夫が必要になります。

8. MX100とDARWINの機能比較

	MX100	優位性	DA100	備考
システム構成	<ul style="list-style-type: none"> ・PC ・ベースプレート ・コントロールユニット ・I/Oモジュール 	>	<ul style="list-style-type: none"> ・PC ・基本/拡張本体 ・サブユニット ・I/Oモジュール ・通信モジュール 	MX100の方が、少chから多chまで、シンプルに構成可能
モジュール種類	10 msAI, 100 msAI, DI, 24V用DI, ひずみ, 4線式RTD抵抗, DO, アナログ出力, PWM出力		10ch/20ch/30chの Universal(mV/TC), パワーモニタ ひずみ DI, mA入力 パルス アナログ出力 通信モジュール	双方豊富
測定周期	<ul style="list-style-type: none"> ・最速 10 ms ・マルチインターバル (3種類の測定周期を混在可能) 	>>	<ul style="list-style-type: none"> ・最速 500 ms もっとも遅いモジュールの周期にあわせる。 	高速測定, マルチインターバルによりMX100の方が高性能と言える
耐圧	入力 ケース間 3700 Vrms(50/60Hz), 1分	>>	入力 ケース間 1500 Vrms(50/60Hz), 1分	MX100の方が高耐圧
入力点数	最大60ch/1ユニット, システムで1200ch (MXLOGGER使用時)		60ch/1サブユニット, 最大300ch, それ以上はDAQLOGGERで接続	双方多ch測定可能
アラーム	本体とPCソフトで分担。本体: 2レベル/ch 上下限・差上下限, PCソフト: 演算+コマンドDO, 変化率警報	<	すべて本体で行う。4レベル/ch 上下限・差上下限 変化率警報	本体でのアラーム処理はDARWINの方が優位
演算機能	本体とPCソフトで分担。本体: 差演算, スケーリング, ソフトフィルタ。PCソフト: 演算機能標準装備。Darwin/M1相当の演算は, PCソフトで可能。	>	すべて本体で行う。差, スケーリング, 移動平均	MX100の方が, 柔軟性に優れる
レポート機能	なし。Excelマクロなど活用	<	/M3オプションで用意	レポート作成はDA100が優位
通信機能	Ethernetのみ。自動検出, IPアドレス自動設定など初期設定の簡素化を図る。		Ethernet, GP-IB, RS232C, RS422/485	通信種はDA100の方が多いが, 使い勝手はMX100が優る
CFカード	通信断時のデータバックアップ機能等	>	なし	MX100の方が信頼性において優位

F0801.EPS

Blank Page