1. MDS の LCD 表示器(オプション)を使用した外部文字表示

外部表示器の MDS100 は 4 行×20 キャラクターの英数字(文字コード送信時はカタカナ表示可能)を T100MX の RS485(COMM3)に接続可能です。

AD 変換した温度データを毎秒表示します。

```
Temperature Check
Setting Rm = XX ℃
```

XX は AD チャンネル#1 のデータを演算した数値を表示します。
各 AD チャンネルのフルスケールは 4096 (0～4096)で温度センサの表示温度(0～50)を演算変換しています。

Custom Function #1

```
PRINT #3 "?P0101"
PRINT #3 "Temperature Check"

PRINT #3 "?P0102"
PRINT #3 "Setting Rm = "; ADC(1) * 50/4096; CHR$(223); "C"
```

コメント:
TRILOGI 組込みの 1.0sec パルスで(毎秒時)にカスタム関数#1 を呼び出し(起動)し、カスタム関数#1 にプログラムされた AD チャンネル#1 の入力電圧を AD 変換し、PRINT #3 (COMM ポート 3)に接続された外部表示器に表示データを書式指定して送信します。

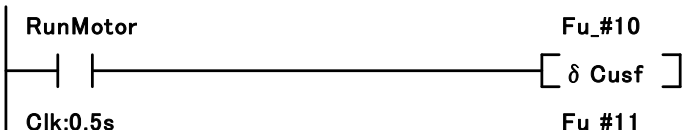
2. ステッピングモータの制御

ステッピングモータの最大スピード・加/減速度およびステップパルスのトータルのカウント数を組込みコマンドで簡単制御できます。

プログラム割当

- 回転パルス　100,000 ステップ
- 最大速度　5000 pps
- 加速速度　100 ステップ/フルスピード
- 表示器　ステップ数/0.5sec 毎
- 移動終了　内部リレー#20 番 ON

```
X = 5286
```



Custom Function #10

```
SETPSPEED 1, 5000, 100        ' Motor1 Max 5,000pps, acc100
STEPMOVE 1, 100000, 20        ' Move 100,000step RLY20 ON
```

Custom Function #11

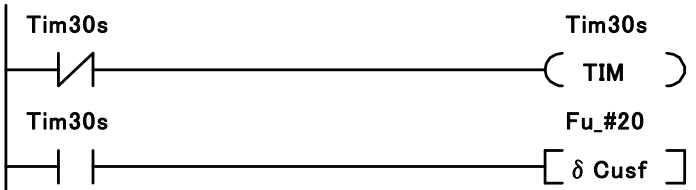
```
PRINT #3 "?P0103"                       ' Display line3
PRINT #3 "X = "; STEPCOUNT(1)           ' Move Steps
```

3. タイムスケージュールに基づくイベント

T100MX に内臓されたリアルタイムクロックによって正確な時間および日付からカレンダースケジュールイベントに基づくプログラムを作成できます。

- 毎日 19:00 時に出力ビット 1 番を ON
- 毎日 07:00 時に出力ビット 1 番を OFF
- 西暦 2000 年の 1 月の 12:00 時に
  出力ビット 5 番を ON
- 西暦 2000 年の 1 月の 18:00 時の
  出力ビット 5 番を OFF

組込み Tim30s パルスでカスタム関数#20 を起動します。RTC の時間および日付関数で理論演算を行い真偽判定の結果、TBASIC 組込みコマンドの SETBIT/LRBIT で出力ビットの ON/OFF を行います。



Custom Function #20

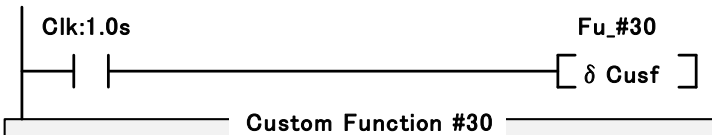
```
IF TIME[1] = 19 AND TIME[2] = 0          ' Hour at 19
   SETBIT OUTPUT[1],0                    ' Minute at 00
ELSE IF TIME[1] = 7 AND TIME[2] = 0
   CLRBIT OUTPUT[1],0
ENDIF

IF DATE[1] = 2000 AND DATE[2] = 1        ' Jan, Year 2000
   IF DATE[3] = 1
      IF TIME[1] = 12 SETBIT OUTPUT[1],4: ENDIF
      ELSE TIME[1] = 18 CLRBIT OUTPUT[1],4: ENDIF
   ENDIF
ENDIF
```

4. 自動空調温度制御

プログラム割当

- A/D チャンネル#5 から設定温度用のポテンシャルメータの設定値を変数(S)に代入します。(温度 16.0～30.0℃/変換 0～4096)
- A/D チャンネル#1 から温度センサーの変換データを変数(T)に代入します。(温度-10.0～50.0℃/変換 0～4096)
- 設定温度(S)が温度センサー(T)より 1.5℃高い場合は出力ビット 1 を ON ます。(加熱)
- 設定温度(S)が温度センサー(T)より 1.5℃低い 場合は出力ビット 2 を ON ます。(冷却)
- 設定温度(S)と温度センサー(T)の差が 1.5℃以内の場合は出力ビット 1･2 を OFF にします。
- 設定温度(S)と温度センサー(T)の値を表示器に表示します。

```
Set Point : 25.0 C
Actual    : 28.0 C
```

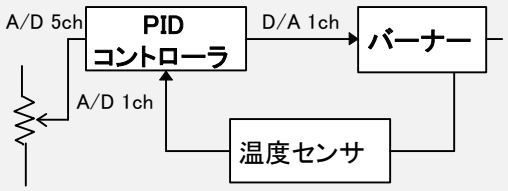


Custom Function #30

```
S = ADC(5) * (300+160)/4096 + 160        ' Convert℃ × 10
T = ADC(1) * (500-100)/4096 - 100        ' Convert℃ × 10

IF S - T > 15 : SETBIT OUTPUT[1],0       ' Out1 ON
ELSE CLRBIT OUTPUT[1],0 : ENDIF          ' Out1 OFF

IF S - T < -15 : SETBIT OUTPUT[1],1      ' Out2 ON
ELSE CLRBIT OUTPUT[1],1 : ENDIF          ' Out2 OFF


PRINT #3 "?P0101" : PRINT #3 "Set Point : " ; S/10 ; " C"
PRINT #3 "?P0102" : PRINT #3 "Actual    : " ; T/10 ; " C"
```

コメント:
全ての温度読取りデータは、× 10 倍の単位で演算し(16.0℃は 160 /-10.0℃は-100)、表示データには小数点を付加します。

5. ヒーターの PDI 制御



PID制御のフードバックダイアログ

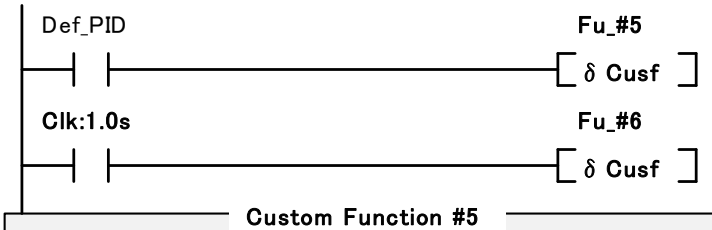
P.I.D.制御の自動演算関数

$$G(s) = K_P + \frac{K_I}{s} + K_D s$$

$$K_P = \text{Proportional Gain} = \frac{1}{\text{Porportional Band}}$$

$$K_I = \text{Integral Gain} = \frac{1}{\text{Integral Time Constant}}$$

- A/D 5ch から設定温度用(50～200℃)のポテンシャルメータの設定値(S)を読込みます。
- A/D 1ch から温度センサの測定値(T)を読込みます。
- PID の自動演算の結果を変数 X に代入します。
- 変数 X の演算結果に基くフィードバック制御として制御用電圧 D/A 1ch に出力します。
- サンプリングと制御演算は毎秒時に実行します。



Custom Function #5

```
P = 500: I = 50: D = 0
PIDDEF 1, P, I, D, 2048 * 100    ' Use PID Engine #1, Max Limut
                                 ' = +/-50% Full Scale
```

Custom Function #6

```
S = ADC(5) * (200-50)/4096 + 50          ' Convert to ℃
T = ADC(1) * (300-0)/4096

X = PIDcompute(1, S - T)/100 + 2048      '
setDAC 1, X                              ' D/A Output #1
```

コメント:

比例係数のゲイン K_P･K_I･K_Dは演算すために 0.01 を表します。
例えば K_P =5 は変数 P=500/K_I=5 は変数 K=50 で表されます。

PIDDEF ステートメントの+2048 は K_Dの D パラメータと同じスケール単位に合わせるために 100 乗じて演算します。
そして PIDcompute( )関数よって返された値を、コントローラ出力に割り当てるには再度 100 で割ります。

PIDcompute( )は下限値から上限値に変化したら正整数値を返します。ここで D/A 出力の 50% (4096/2 = 2048)が負の数値を表せるように演算します。