MITSUBISHI ELECTRIC

三菱電機 通用 可程式控制器

MELSEC iQ-R series

MELSEC iQ-R 結構化文本(ST)
程式指南

安全注意事項

(使用之前請務必閱讀)

使用本產品之前，應仔細閱讀本手冊及本手冊中所介紹的關聯手冊，同時在充分注意安全的前提下正確操作。

本手冊中記載的注意事項僅與本產品相關。關于可程式控制器系統方面的安全注意事項，請參照所使用CPU模組的用戶手冊。

在“安全注意事項”中，安全注意事項分為“⚠警告”和“⚠注意”兩個等級。

⚠警告	表示錯誤操作可能造成災難性後果，引起死亡或重傷事故。
⚠注意	表示錯誤操作可能造成危險的後果，引起人員中等傷害或輕傷，還可能使設備損壞。

此外，根據情況不同，即使“⚠注意”這一級別的事項也有可能引發嚴重後果。

兩級注意事項記載的都是重要內容，請務必遵照執行。

請妥善保管本手冊以備需要時閱讀，並將本手冊交給最終用戶。

[設計注意事項]

⚠警告

- 應在可程式控制器系統的外部設置互鎖電路，以便在通過計算機對運行中的CPU模組進行資料更改、程式更改、狀態控制時，能夠確保整個系統始終安全運行。此外，通過計算機對CPU模組進行在線操作時，應預先確定由于電纜連接不良等導致發生通信異常時的系統處理方法。

關于產品的應用

（1）使用三菱可程式控制器時，請符合以下條件：
即使可程式控制器出現問題或故障時，也不會導致重大事故。並且在設備外部以系統性規劃，當發生問題或故障時的備份或失效安全防護功能。

（2）三菱可程式控制器是以一般工業等用途為對象，設計和製造的泛用產品。
因此，三菱可程式控制器不適用於以下設備、系統的特殊用途上。如果用於以下特殊用途時，對於三菱可程式控制器的品質、性能、安全等所有相關責任（包括，但不限定於債務未履行責任、瑕疵擔保責任、品質保證責任、違法行為責任、製造物責任），三菱電機將不負責。

- 各電力公司的核能發電廠以及其他發電廠等，對公眾有較大影響的用途。
- 各鐵路公司及公家機關等，對於三菱電機有特別的品質保證體制之架構要求的用途。
- 航空宇宙、醫療、鐵路、焚燒、燃料裝置、乘載移動設備、載人運輸裝置、娛樂設備、安全設備等，預測對性命、人身、財產有較大影響的用途。

但是，即使是上述對象，只要有具體的限定用途，沒有特殊的品質（超出一般規格的品質等）要求之條件下，經過三菱電機的判斷依然可以使用三菱可程式控制器，詳細情形請洽詢當地三菱電機代表窗口。

前言

在此感謝貴方購買了三菱電機可程式控制器MELSEC iQ-R系列的產品。
本手冊用于幫助理解如何使用GX Work3進行結構化文本程式等內容。
使用之前應熟讀本手冊及關聯手冊，在充分理解MELSEC iQ-R系列可程式控制器的功能、性能的基礎上正確地使用本產品。
此外，將本手冊中介紹的程式示例用于實際系統時，要對物件系統進行充分驗證以保證在控制方面沒有問題。

目錄

安全注意事項 1
關于產品的應用 2
前言 2
關聯手冊 6
術語 6

第1部分　ST程式

第1章　什麼是ST　8

1.1　國際標准IEC 61131-3 8
1.2　ST語言的特點 8
1.3　與其他語言區分使用、混合使用 10

第2章　編寫的基本規則　11

2.1　可使用的字元 11
字元編碼 11
構成單位（標記） 11
2.2　可使用陳述式和函數 12
2.3　語句和表達式 13
語句 13
表達式 14

第3章　使用ST使程式更易于理解　15

3.1　運算表達式 15
代入(:=) 15
四則運算(+、-、*、/) 15
高級運算（指數函數、三角函數） 17
邏輯運算(AND、OR、XOR、NOT) 18
比較(<,>,<=,>=)、一致/不一致(=,<>) 18
3.2　條件分支 19
基于BOOL值的條件分支（IF） 19
基于整數值的條件分支（CASE） 21
3.3　循環處理 22
通過BOOL條件循環（WHILE、REPEAT） 22
重覆執行直到变数達到設定值（FOR） 24

第4章　多種資料類型的處理　25

4.1　BOOL值 25
4.2　整數和實數 25
值的範圍 25
自動進行的類型轉換 26
算術表達式的運算結果的資料類型 27
整數和實數的除法運算 27
4.3　字元串 28
4.4　時間 29
時間型（TIME）变数 29

時鐘資料（日期資料、時間資料）. 30
4.5 **數組和結構體** 31
數組 31
結構體 32
組合了結構體和數組的資料類型 32

第5章 使用ST表述梯形圖 33

5.1 **表述觸點、線圈** 33
常開觸點和線圈 33
常閉觸點（NOT）. 33
串聯連接、並聯連接（AND、OR）. 34
執行順序比較複雜的觸點、線圈 35
5.2 **陳述式表述** 36
梯形圖和ST均可以使用的陳述式 36
賦值可使用的陳述式 37
可以使用運算符表述的陳述式 37
可以使用控制語法、FUN/FB表述的陳述式 38
5.3 **聲明、注解的表述** 38

第6章 程式創建步驟 39

6.1 **步驟概要** 39
6.2 **打開ST編輯器** 39
6.3 **編輯ST程式** 39
輸入文本 40
輸入控制語法 40
輸入注釋 41
使用標簽 42
創建函數、FB 44
輸入函數 46
輸入FB 48
6.4 **轉換程式、調試** 50
轉換程式 50
確認錯誤/警告. 50
6.5 **確認機器的實際動作** 51
在可程式控制器中執行程式 51
確認執行中的程式 52
6.6 **使梯形圖的一部分成為ST（內嵌ST）** 53

第2部分 程式示例

第7章 程式示例的概要 56

7.1 **程式示例一覽** 56
7.2 **通過GX Works3創建程式示例時** 57

第8章 計算器處理（四則運算和條件分支） 58

8.1 **初始化程式：Initialization** 59
8.2 **四則運算（FUN）：Calculation** 60
8.3 **化整處理（FUN）：Rounding** 61
8.4 **尾數處理（FUN）:FractionProcessing** 62

8.5 計算器程式：Calculator 63
8.6 含稅計算程式：IncludingTax 65

第9章 定位處理（指數函數、三角函數和結構體） 66
9.1 根據X、Y坐標計算旋轉角度(FUN)：GetAngle 67
9.2 計算X、Y坐標的2點間距離（FUN）：GetDistance 68
9.3 根據半徑和角度計算X、Y坐標（FUN）：GetXY 69
9.4 電機的陳述式脈衝數的計算（FB）：PulseNumberCalculation 70
9.5 X、Y坐標的位置控制程式：PositionControl 71

第10章 次品分類（數組和循環處理） 73
10.1 產品檢查（FB）：ProductCheck 74
10.2 產品資料的分類（FB）：Assortment 75
10.3 產品資料管理程式：DataManagement 76

第11章 運轉時間計測（時間和字元串） 78
11.1 運轉時間管理程式：OperatingTime 79
11.2 閃爍定時器（FB）：FlickerTimer 80
11.3 指示燈的點亮/熄滅程式：LampOnOff 81
11.4 从秒至時、分、秒的轉換程式：SecondsToTimeArray 82
11.5 从時間型至字元串的轉換程式：TimeToString 83

附錄 86
附1 ST語言規格 86
語句 86
運算符 87
注釋 87
軟元件 88
標簽 89
常數 90
函數和FB 92
附2 ST中無法使用的陳述式 94
賦值可使用的陳述式 94
運算符可使用的陳述式 94
控制語句、函數等可使用的陳述式 96
不需要的陳述式 96

索引 97

修訂記錄 99
商標 100

關聯手冊

手冊名稱[手冊編號]	說明	提供形式
MELSEC iQ-R 結構化文本(ST)程式指南 [SH-081466CHT](本手冊)	對在GX Works3中的結構化文本 （ST）程式進行說明。以下通過樣本程式對基本操作方法及功能進行說明。	e-Manual PDF
GX Works2入門指南(結構化工程篇) [SH-081010CHT] Structured Text (ST) Programming Guide Book [SH-080368E] 對在GX Developer中的結構化文本 （ST）程式進行說明。 裝訂本	對在GX Works2中的結構化文本 （ST）程式進行說明。	PDF
PDF		

要點

e-Manual是可以使用專用工具進行瀏覽的三菱電機FA電子書籍手冊。

e-Manual具有以下特點。

- 可以从多本手冊同時搜尋需要的資訊 （跨手冊搜尋）
- 可以通過手冊內的鏈接瀏覽其他手冊
- 可以通過產品插圖的各部分瀏覽想要了解的硬體規格
- 可以將需要頻繁瀏覽的資訊登錄到收藏夾

術語

除特別注明的情況外，本手冊中使用以下術語進行說明。

術語	內容
FB	功能塊的簡稱。
FB	可作為程式部件使用的函數。可通過內部变数保持值。要進行實例化後使用。
FUN	函數的簡稱。
GX Developer	MELSEC可程式控制器軟體包的產品名。 產品型號SW8D5C-GPPW的產品名總稱。
GX Works2	MELSEC可程式控制器軟體包的產品名。 產品型號SWnDND-GXW2及SWnDNC-GXW2的產品名總稱。（n代表版本。）
GX Works3	MELSEC可程式控制器軟體包的產品名。 產品型號SWnDND-GXW3的產品名總稱。（n代表版本。）
LD	梯形圖 （Ladder Diagram）的簡稱。
ST	結構化文本的簡稱。
工程工具	用于進行可程式控制器的設置、程式、調試乃至維護的工具。
執行程式	指經過轉換的程式。可在CPU模組中執行的程式。
程式部件	按功能分別定義的程式單位。通過程式的部件化，在分層程式的下位處理中，可按處理內容或功能分成若幹單位，並按單位程式。
軟元件	系統對名稱、型號、使用方法進行定義的变数。
函數	可作為程式部件使用的函數。對同一輸入輸出同一結果。
標簽	用戶聲明的变数。
實例	對定義的內部变数分配軟元件，並使其為可進行處理、執行的狀態的FB的實體。針對同一FB定義，可生成多個實例。

第1部分 ST程式

本部分對MELSEC iQ-R系列中的ST程式進行說明。

1 什麼是ST

2 編寫的基本規則

3 使用ST使程式更易于理解

4 多種資料類型的處理

5 使用ST表述梯形圖

6 程式創建步驟

1 什麼是ST

1.1 國際標准IEC 61131-3

IEC 61131是由國際標准團體之一IEC(International Electrotechnical Commission: 國際電工委員會）制定的可程式控制器（PLC：Programmable Logic Controller)系統相關的國際標准。
IEC 61131-3標准規定了5種程式語言 （IL/LD/ST/FBD/SFC），並提出了程式編制的指導方針。

1.2 ST語言的特點

ST語言是一種文本格式自由的表述方法，可以像C語言等計算機用程式語言一樣進行程式。因為便于對運算處理、資料處理進行表述，因此程式的可視性會更高。

算式的運算處理

算術運算和比較運算等可以像一般的表達式那樣進行表述。

■多個運算可寫在同1行中

可以使用運算符 （+、-等）簡潔地進行表述，因此程式比梯形圖更易于理解。

程式範例

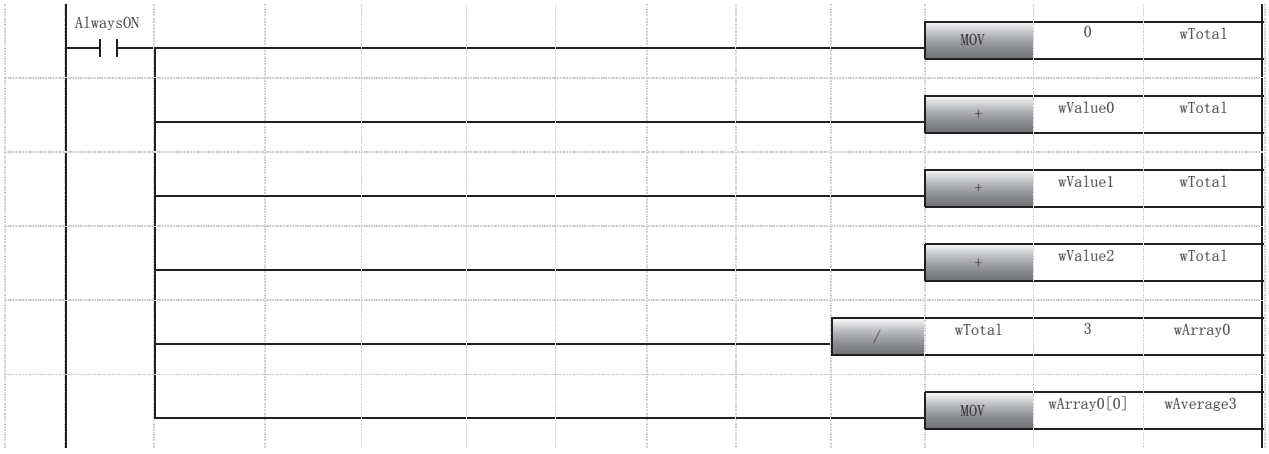
在wAverage3中代入wValue0～wValue2的平均值。

wAverage3 = (wValue0 + wValue1 + wValue2) ÷ 3

ST

```
wAverage3 := (wValue0 + wValue1 + wValue2) / 3;
```

LD



要點

關于使用ST語言的表述方法，請參照以下內容。

☞15頁 運算表達式

■無需考慮資料類型的表達式表述

需要考慮小數點以下部分時，其表達式的表述方法相同。實數型資料的複雜運算也可以簡潔地進行表述。

程式範例

換算類比資料。

換算值 = (轉換後.上限 - 轉換後.下限) ÷ (轉換前.上限 - 轉換前.下限) × (測定值 - 轉換前.下限) + 轉換後.下限

ST

```
eScalingValue := ( stAfter.eUpperLimit - stAfter.eLowerLimit ) / (stBefore.eUpperLimit - stBefore.eLowerLimit) * (eMeasurements - stBefore.eLowerLimit) + stAfter.eLowerLimit;
```

要點

關于使用ST語言的表述方法，請參照以下內容。

☞25頁 整數和實數

複雜的資訊處理

可以通過選擇語句或循環語句等語法編寫控制程式。與梯形圖相比，能夠更簡潔明了地表述根據不同條件對執行內容進行複雜分支處理或循環處理等。

程式範例

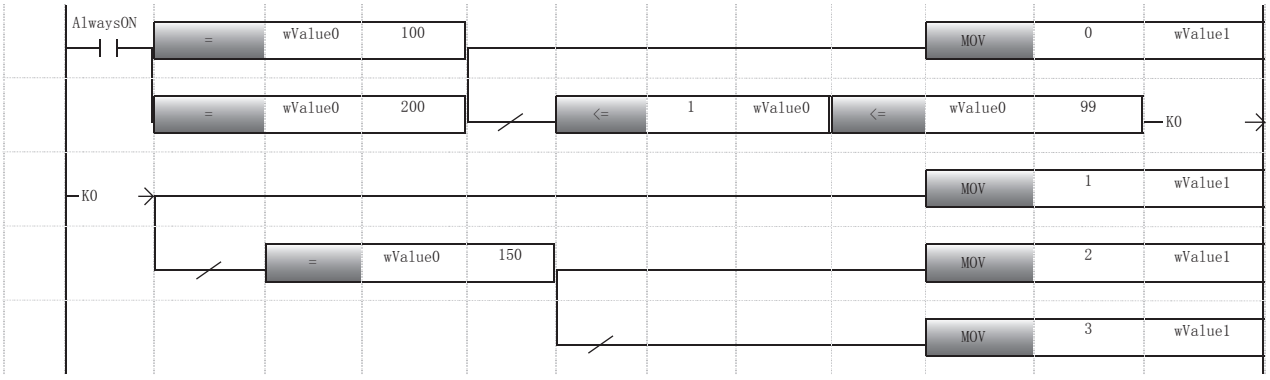
根據wValue0的值在wValue1中設置0～3。

- 100或200時： 0
- 1～99時： 1
- 150時： 2
- 上述以外時： 3

ST

```
CASE wValue0 OF
    100, 200: wValue1 := 0;
    1..99: wValue1 := 1;
    150: wValue1 := 2;
    ELSE wValue1 := 3;
END_CASE;
```

LD



要點

關于使用ST語言的表述方法，請參照以下內容。

☞19頁 條件分支，22頁 循環處理

1.3 與其他語言區分使用、混合使用

IEC 61131-3中所記載的5種程式語言各自的特點如下所示。

程式語言		特點
IL	陳述式列表	適用于有內存限制或需要高速處理的情況。可編寫緊湊的程式。
LD	梯形圖	適用于單純的繼電器順控處理。
ST	結構化文本	適用于算式的運算處理及複雜的資訊處理。是熟悉計算機程式語言的技術人員容易掌握的一種語言。
FBD	FB圖	適用于連續的類比信號處理。
SFC	順序功能圖	適用于基于狀態轉移的順序控制。

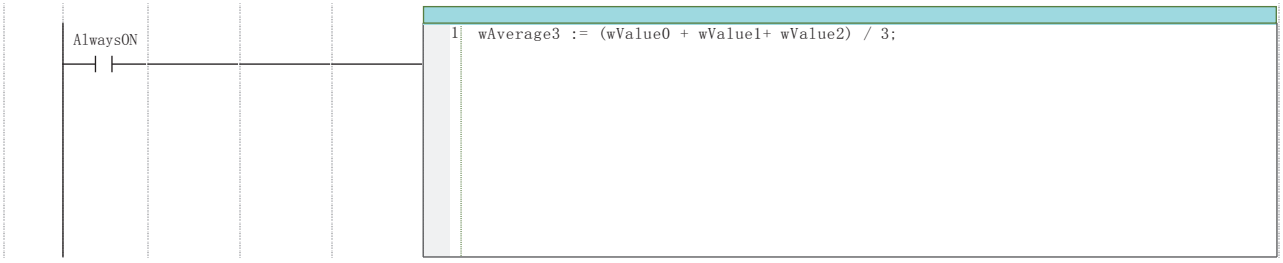
根據不同處理組合使用不同語言來編寫程式，可發揮各種程式語言的優勢。

在GX Works3中，以下功能中可將其他語言和ST組合使用。

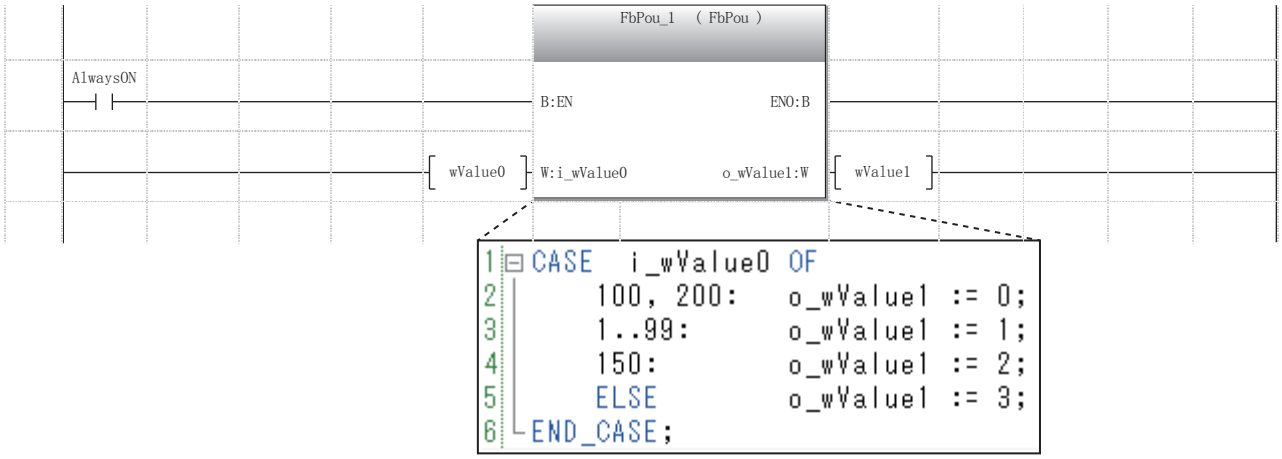
功能	可組合使用的語言	內容
內嵌ST	LD	繼電器順控處理中的一部分處理使用ST表述。
FUN/FB	LD FBD	將使用ST表述的子程式定義為程式部件，在LD等的其他程式中調用。

將單純的繼電器時序處理或需要對動作進行直觀確認的處理使用梯形圖進行表述，將部分複雜處理定義成部件使用ST進行表述，可使程式更易于理解。

使用了內嵌ST的LD



使用了FUN/FB的LD



2 編寫的基本規則

本章對使用基本ST語言編寫程式的方法進行說明。

2.1 可使用的字元

ST語言是文本格式的程式語言。本節對使用的字元及符號進行說明。

字元編碼

GX Works3支持Unicode(UTF-16)的基本多語言面。
日語、英語、中文等多種語言的基本字元及大部分符號除了可以用于注釋外，還可以用于標簽名或資料名中。

限制事項

關于標簽名或資料名等中不能使用的字元串 （保留字），請參照以下手冊。
GX Works3操作手冊

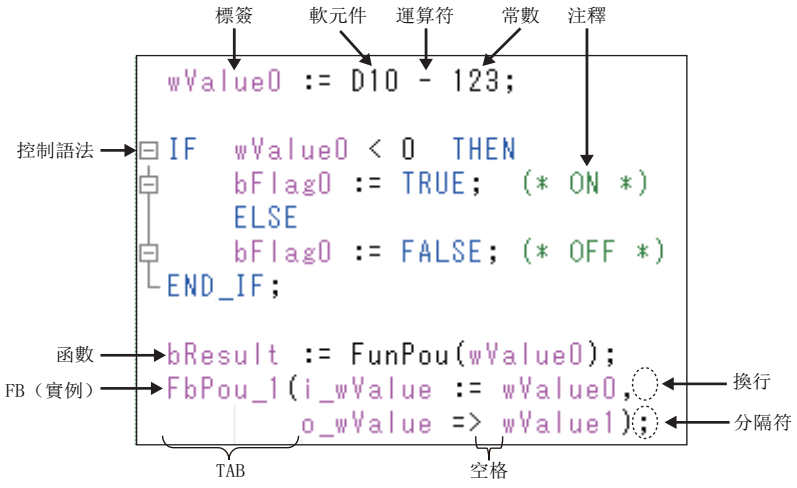
構成單位 （標記）

構成程式的單詞或符號的最小單位稱為標記 （Token）。
ST語言使用以下記號的組合表述程式。

類型		示例	參照
運算符		+、-、<、>、=、&、NOT	87頁 運算符
控制語法的關鍵字 （定義的標准識別符）		IF、CASE、WHILE、RETURN	86頁 語句
識別符	变数 （標簽、軟元件等）	X0、Y10、M100、D10、ZR0、 bSwitch_A （任意的名稱）	88頁 軟元件 89頁 標簽
	程式部件 （函數、FB）	BMOV、FunPou(任意的名稱)、 COUNTER_FB_M_1、FbPou_1(任意的名稱)	92頁 函數和FB
常數		123、'字元串'、TRUE、FALSE	90頁 常數
分隔符		;、(、)	—

各記號之間可以自由插入空白、換行及注釋。

類型	示例	參照
空白	空格(全角/半角)、TAB	—
換行	換行代碼	—
注釋	(**)、/**/、//	87頁 注釋



2.2 可使用陳述式和函數

在ST語言中，將梯形圖所使用的陳述式作為函數處理。
可以像C語言等中的函數調用一樣使用。

GX Works3中可使用以下函數、FB。

類型	MELSEC-Q/L系列中相當的陳述式、函數	
	GX Works2	GX Developer
CPU模組用陳述式 模組專用陳述式	公共陳述式	MELSEC函數
通用函數 通用FB	應用函數	IEC函數
模組FB	MELSOFT Library	—
MELSOFT Library （樣本庫）		—
任意創建的函數、FB	—	—

限制事項

ST語言不支持一部分並非必要的陳述式 （觸點陳述式）。
☞ 94頁 ST中無法使用的陳述式

要點

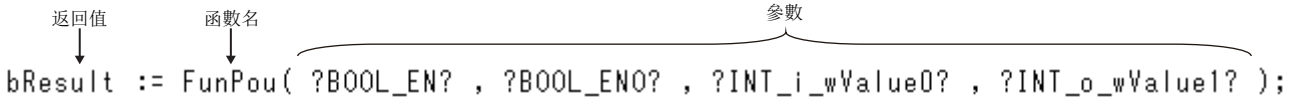
關于陳述式的詳細內容，請參照以下手冊。
📖MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)
📖 MELSEC iQ-R的各模組的FB參照

什麼是函數和FB

函數和FB是對程式中調用的子程式進行了定義的程式部件。

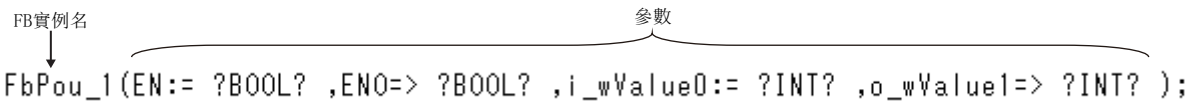
■函數

對同一輸入，輸出同一結果的程式部件。用于將單純的獨立處理定義為部件。
在ST程式中使用時，函數名和參數按如下所述表述。



■FB

可將內部保有的值用于運算的程式部件。根據各實例 （實體）所保有的值，可對同一輸入輸出不同結果。用于將比函數更為複雜的處理或需要利用保有的值反覆執行的處理等定義部件。
在ST程式中使用時，實例名和參數按如下所述表述。



要點

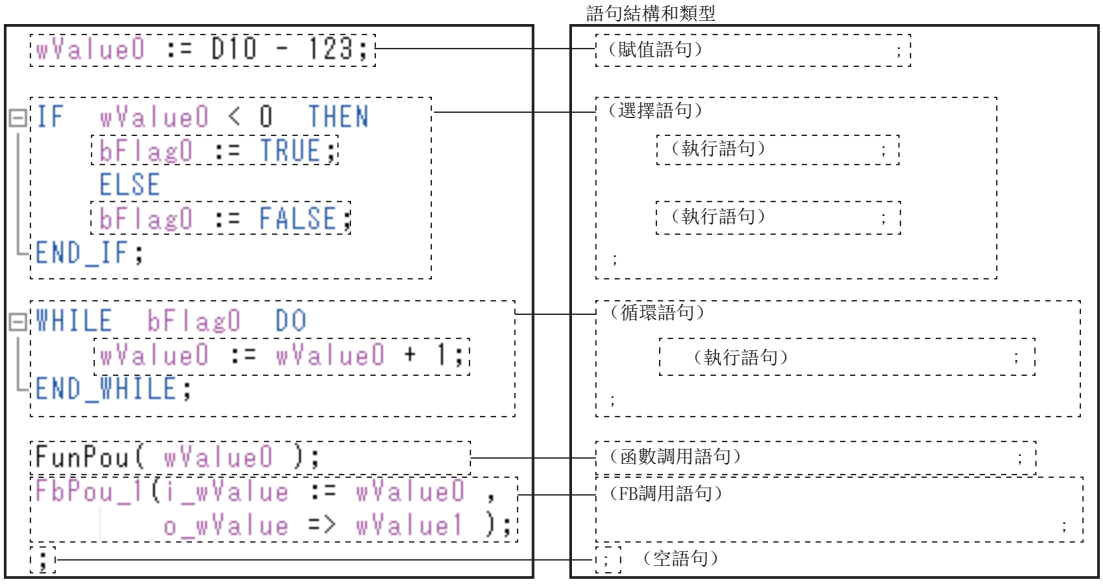
關于使用ST語言的表述方法，請參照以下內容。
☞ 92頁 函數和FB
關于函數和FB的詳細內容，請參照以下手冊。
📖MELSEC iQ-R 程式手冊(程式設計篇)

2.3 語句和表達式

本節對ST語言中編寫程式時使用的“語句”和“表達式”進行說明。

語句

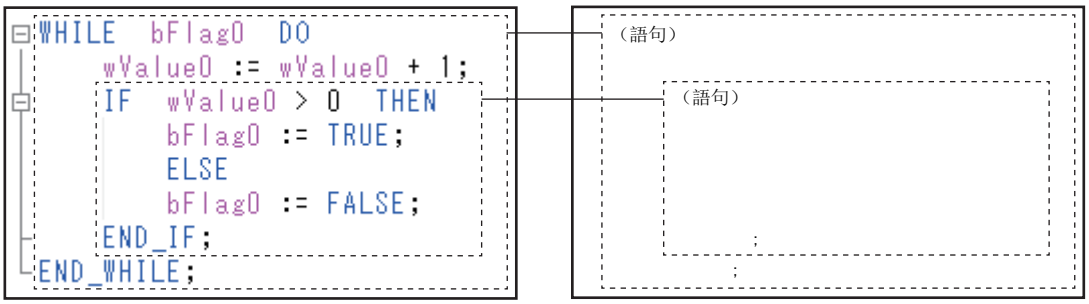
ST語言中，將執行處理的集合稱為“語句”。
程式以“語句”為單位編寫。
“語句”的結尾處，必須帶有“;”（分號）。



以下所示為語句的類型。

類型		內容	示例
賦值語句		將右邊的評價結果代入左邊的變數。	15頁 代入(:=)
控制語法	選擇語句（IF、CASE）	根據條件，選擇執行語句。	19頁 條件分支
	循環語句（FOR、WHILE、REPEAT）	根據結束條件，多次執行執行語句。	22頁 循環處理
	循環語句的中斷（EXIT）	中斷循環語句。	EXIT;
子程式控制語句	調用語句	調用函數、FB等。	92頁 函數和FB
	RETURN語句	中途結束程式。	RETURN;
空語句		不執行任何處理。	;

控制語法可分層。（可以使用其他的語句作為選擇語句或循環語句的執行語句。）

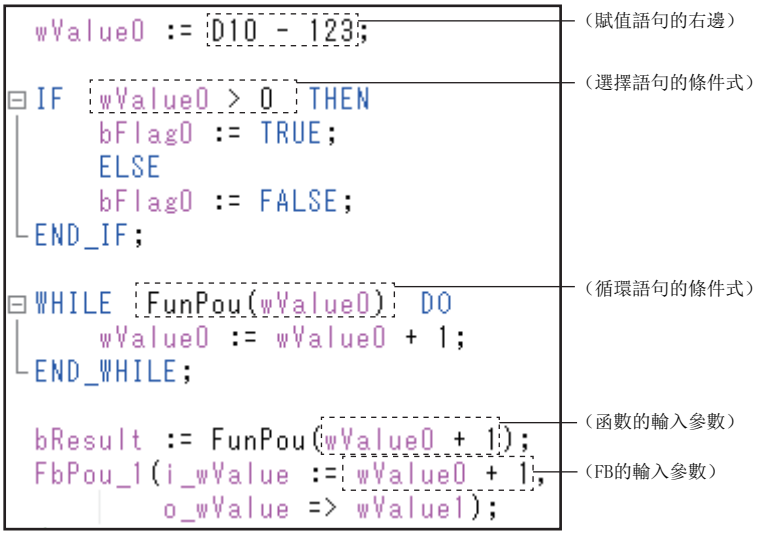


表達式

對處理語句時所需的值進行的表述稱為“表達式”。
表達式由变数、運算符等構成。表達式的值在程式執行中獲得。
表達式用于文中以下所示的位置。

- 賦值語句的右邊
- 函數及FB的執行條件 （EN）或輸入參數
- 選擇語句及循環語句中指定的條件

表達式的使用位置



表達式的資料類型在編譯 （轉換）時自動判斷。表達式的值在程式執行中獲得。
算術、比較等運算式可與一般的表達式一樣用常數、变数和運算符的組合來表述。
ST語言中，变数和常數也作為“表達式”處理。（基本表達式）
以下所示為表達式的類型。

類型		表達式 （運算結果）的資料類型	示例
運算表達式	算術表達式	整數、實數等 （依據運算物件）	wValue0 + wValue1
	邏輯表達式	BOOL值(TRUE/FALSE)	bFlag0 OR bFlag1
	比較表達式	BOOL值(TRUE/FALSE)	wValue0 > 0
基本表達式	变数、常數	定義的資料類型	X0, wValue0, 123, TRUE
	函數調用表達式	返回值的資料類型	FunPou(wValue0, wValue1)

要點

沒有返回值的函數不能作為表達式處理。

3 使用ST使程式更易于理解

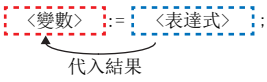
本章以示例對使用ST語言表述可使處理更加簡單易讀進行說明。

3.1 運算表達式

ST語言中，含小數點或指數的複雜運算也能像一般的算術表達式那樣簡潔地進行表述。

代入(:=)

使用賦值語句可以將表達式的結果存入变数。
賦值語句通過:=表述。將右邊的表達式的結果代入左邊的变数。



要點

一般的表達式使用=，但ST語言使用帶有:(冒號)的運算符。

四則運算(+、-、*、/)

四則運算使用與一般的算術符號相同的運算符(+、-、*、/)表述。
對于使用梯形圖陳述式無法一次性表述完整的運算，可以通過單行表達式簡潔地表述出來。

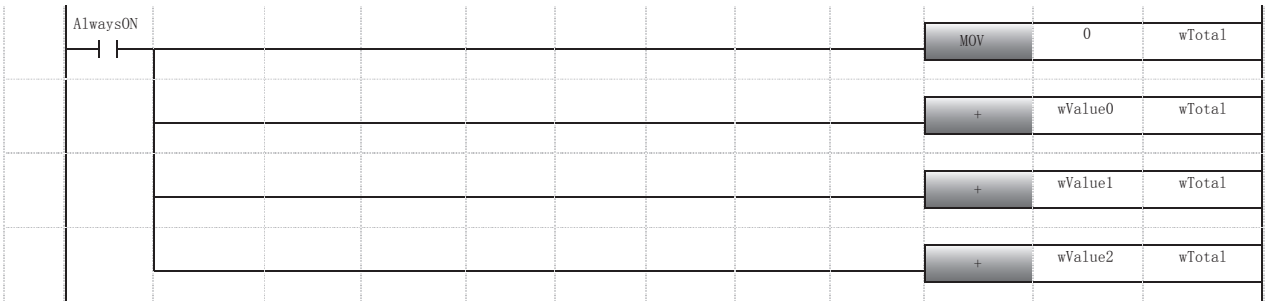
程式範例

在wTotal中代入wValue0～wValue2的和。
wTotal = wValue0 + wValue1 + wValue2

ST

```
wTotal := wValue0 + wValue1 + wValue2;
```

LD



要點

用1個語句彙總表述多個運算表達式時，將从優先級最高的運算符開始處理。
• 四則運算符的優先級 （从高到低）：乘除運算(*、/)，加減運算(+、-)
有多個優先級相同的運算符時，从最左邊的運算符開始運算。

小括號()內的運算表達式將優先進行運算。
在複雜的四則運算中，使用小括號()可使優先級清楚明了。

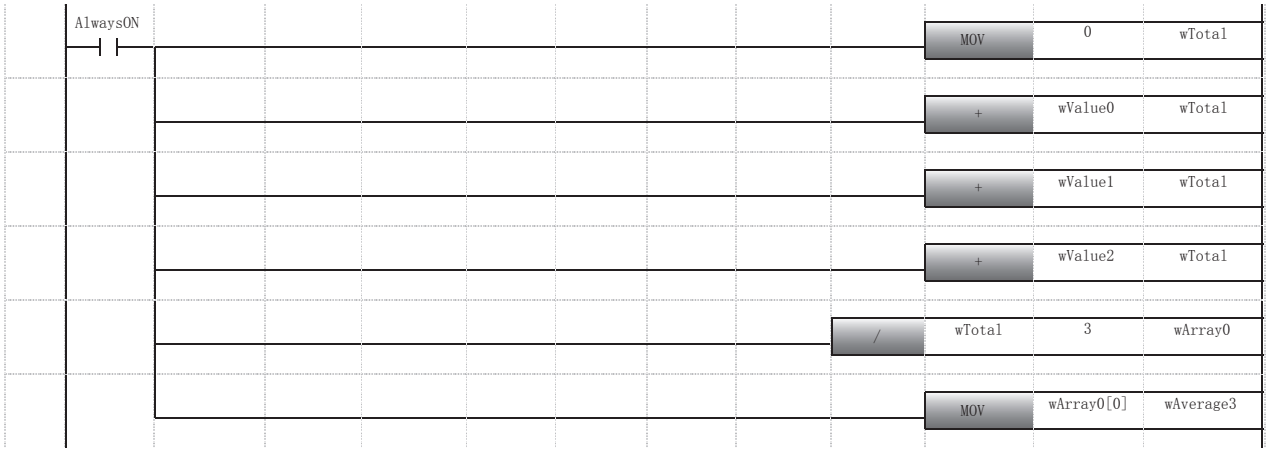
程式範例

在wAverage3中代入wValue0～wValue2的平均值。
wAverage3 = (wValue0 + wValue1 + wValue2) ÷ 3

ST

```
wAverage3 := ( wValue0 + wValue1 + wValue2) / 3;
```

LD



高級運算（指數函數、三角函數）

指數運算或三角函數運算使用通用函數。

類型		函數名	示例	
			一般算式表述	ST
絕對值		ABS	B = |A|	eValueB := ABS(eValueA);
平方根		SQRT	B = $\sqrt{A}$	eValueB := SQRT(eValueA);
自然對數		LN	B = log e^A	eValueB := LN(eValueA);
常用對數		LOG	B = log 10^A	eValueB := LOG(eValueA);
指數		EXP	B = e^A	eValueB := EXP(eValueA);
三角函數	正弦、反正弦	SIN、ASIN	B = SIN A	eValueB := SIN(eValueA);
	餘弦、反餘弦	COS、ACOS	B = COS A	eValueB := COS(eValueA);
	正切、反正切	TAN、ATAN	B = TAN A	eValueB := TAN(eValueA);

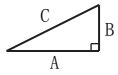
冪乘可使用“**”進行表述。

類型	運算符	示例	
		一般算式表述	ST
冪乘	**	B = C^A	eValueB := eValueC ** eValueA;

程式範例

求直角三角形的斜邊長。

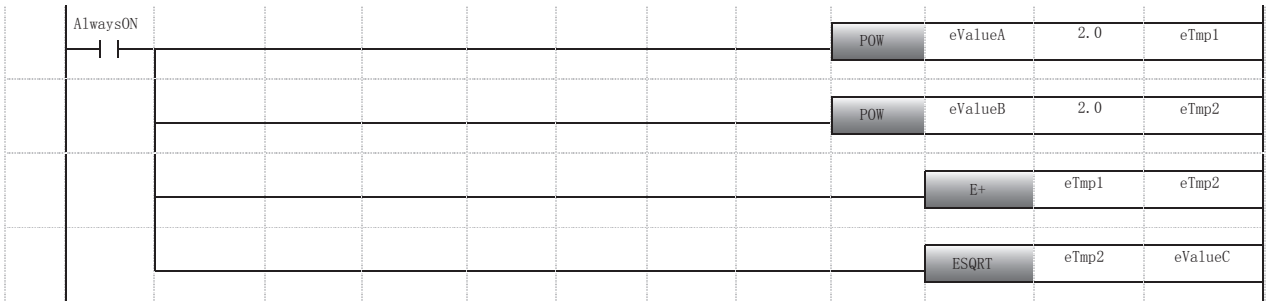
$C = \sqrt{(A^2 + B^2)}$



ST

```
eValueC := SQRT((eValueA ** 2.0)+( eValueB ** 2.0));
```

LD



邏輯運算(AND、OR、XOR、NOT)

邏輯運算不使用符號 （∧、 ∨、 ∀等），而是使用容易輸入容易理解的運算符來表述。

程式範例

在bResult中代入bFlag0和bFlag1的邏輯與(AND)。

ST

```
bResult := bFlag0 AND bFlag1;
```

LD

要點

邏輯與也可使用運算符“&”表述。

用1個語句彙總表述多個運算表達式時，將从優先級最高的運算符開始處理。

• 邏輯運算符的優先級 （从高到低）：邏輯非 （NOT）、邏輯與 （AND、&）、邏輯異或 （XOR）、邏輯或 （OR）

有多個優先級相同的運算符時，从最左邊的運算符開始運算。

比較(<, >, <=, >=)、一致/不一致(=, <>)

比較運算使用與一般算術符號相同的等號、不等號的運算符進行表述。

程式範例

在bResult中代入wValue0和wValue1的比較結果 （一致時： TRUE、不一致時： FALSE）。

ST

```
bResult := wValue0 = wValue1;
```

LD

要點

“=”號在ST語言中作為對右邊和左邊的值比較是否一致的運算符使用。

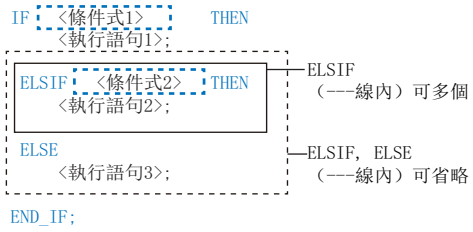
賦值語句應使用帶分號的“:=”表述。

3.2 條件分支

ST語言中可像C語言等高級程式語言一樣使用根據條件進行的分支處理。
使用選擇語句 （IF、CASE），可清楚明確地表述不同場合下的執行動作。

基于BOOL值的條件分支 （IF）

基于某個條件的真 （TRUE）或假 （FALSE）的分支處理可使用IF語句表述。
IF語句執行以下處理。



1. IF的判斷
條件式1為真 （TRUE）時，執行執行語句1。

2. ELSIF的判斷
此前的條件式為假 （FALSE）時，對條件進行判斷。
條件式2為真 （TRUE）時，執行執行語句2。

3. ELSE的判斷
“IF”及“ELSIF”的所有條件式均為假 （FALSE）時，執行“ELSE”後的執行語句3。

要點

在IF語句的條件式中，可作以下指定。
- 結果為BOOL值的運算表達式
- BOOL型的变数
- 返回值為BOOL型的函數調用式

可設置多項基于ELSIF的條件分支 （ELSIF<條件式>THEN<執行語句>;）。
根據實際需要使用基于ELSIF、ELSE的條件分支。（可省略）

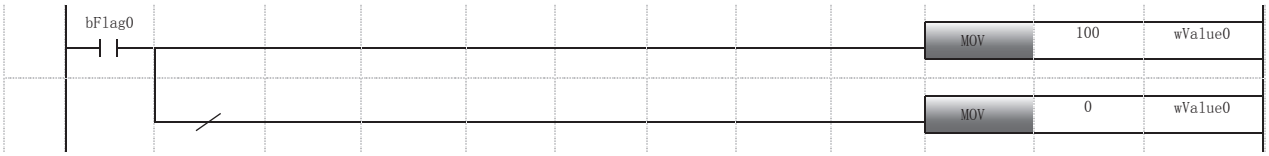
程式範例

根據bFlag0的條件，在wValue0中設置不同的值。
- ON時 （bFlag0為TRUE）： wValue0 = 100
- OFF時 （bFlag0為FALSE）： wValue0 = 0

ST

```
IF bFlag0 THEN
    wValue0 := 100;
    ELSE
    wValue0 := 0;
END_IF;
```

LD



通過將邏輯運算（AND、OR等）和比較（<、>、=等）的組合表達式用于選擇語句的條件式，可表述按複雜的條件進行分支的程式。

程式範例

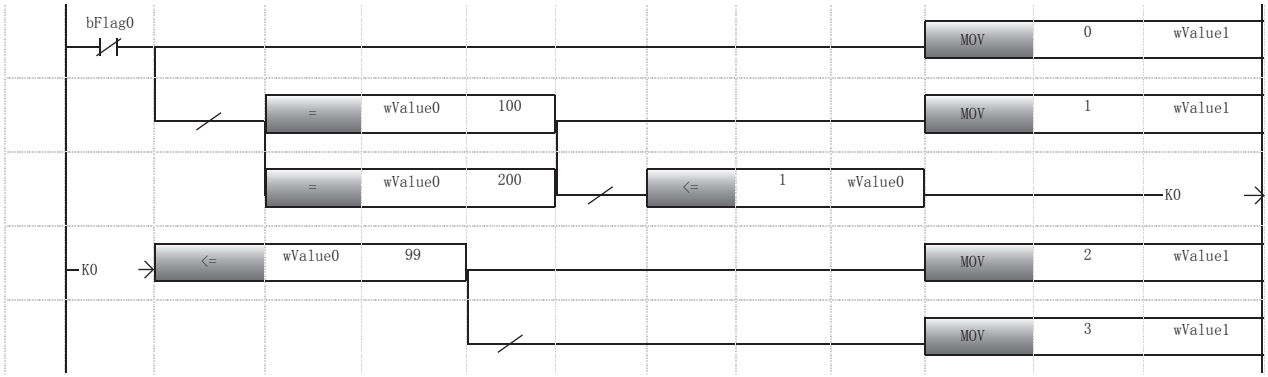
根據bFlag0和wValue0的值在wValue1中設置0～3。

- bFlag0為FALSE時： 0
- bFlag0為TRUE，wValue0為100或200時： 1
- bFlag0為TRUE，wValue0為1～99時： 2
- 上述以外時： 3

ST

```
IF NOT bFlag0 THEN
    wValue1 := 0;
    ELSIF (wValue0 = 100) OR (wValue0 = 200) THEN
    wValue1 := 1;
    ELSIF (1 <= wValue0) AND (wValue0<= 99) THEN
    wValue1 := 2;
    ELSE
    wValue1 := 3;
END_IF;
```

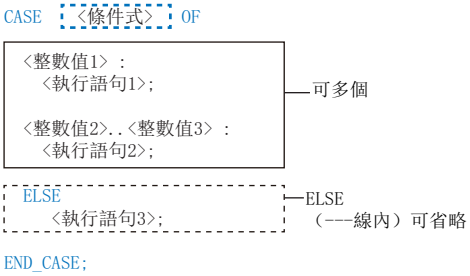
LD



基于整數值的條件分支 （CASE）

基于整數值的分支處理可以使用CASE語句簡潔明了地進行表述。

CASE語句執行以下處理。



根據整數值的條件，選擇執行語句。

1. 整數值1的判斷

條件式的結果與整數值1相一致時，執行執行語句1。

2. 對指定了範圍的整數值的判斷

通過範圍指定要判斷的整數值時，使用“..”表述。

條件式結果的值在整數值2～整數值3的範圍內時，執行執行語句2.

3. ELSE的判斷

與所有的整數值或範圍都不一致時，執行“ELSE”後的執行語句3。

要點

在CASE語句的條件式中，可作以下指定。

- 結果為整數 （INT）值的運算表達式
- 整數 （INT）型的变数
- 返回值為整數 （INT）型的函數調用式

可設置多項基于整數值的條件分支 （〈整數值〉:〈執行語句〉;）。

根據實際需要使用基于ELSE的條件分支 （ELSE〈執行語句〉;）。（可省略）

程式範例

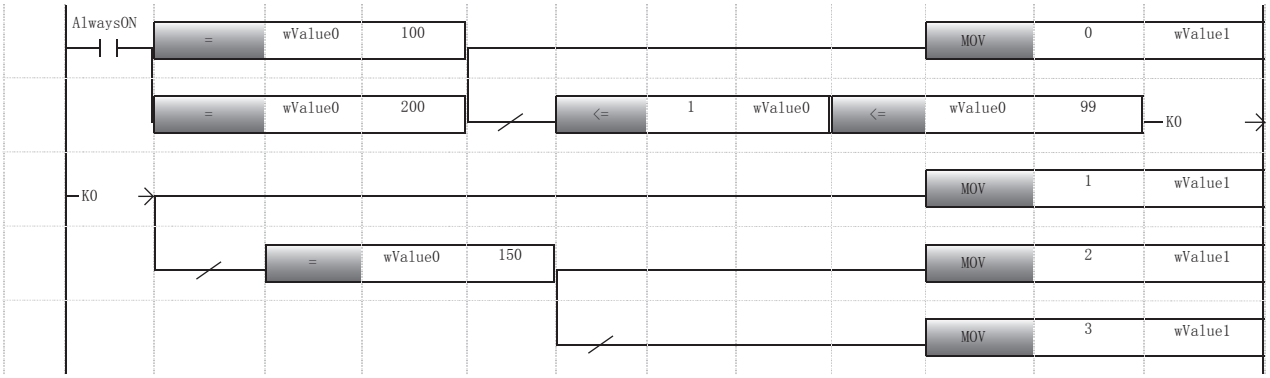
根據wValue0的值在wValue1中設置0～3。

- 100或200時： 0
- 1～99時： 1
- 150時： 2
- 上述以外時： 3

ST

```
CASE wValue0 OF
    100, 200: wValue1 := 0;
    1..99: wValue1 := 1;
    150: wValue1 := 2;
    ELSE wValue1 := 3;
END_CASE;
```

LD



3.3 循環處理

使用循環語句 （WHILE、REPEAT、 FOR）時，可以指定結束條件，重覆多次執行處理。

通過BOOL條件循環 （WHILE、REPEAT）

基于某個條件的真 （TRUE）或假 （FALSE）的循環處理，可使用WHILE語句或REPEAT語句進行表述。

WHILE語句執行以下處理。

```
WHILE  <條件式>
    DO
      <執行語句>;
END_WHILE;
```

循環直至變為假（FALSE）

根據BOOL值的結束條件，多次執行執行語句。
反覆執行直到條件式的結果變為假 （FALSE）為止。

1. 條件的判斷

條件式為假 （FALSE）時，結束處理。

2. 處理的執行

條件式為真 （TRUE）時，執行執行語句並重覆處理。

REPEAT語句執行以下處理。

```
REPEAT
  <執行語句>;
  UNTIL  <條件式>
END_REPEAT;
```

循環直至變為真（TRUE）

根據BOOL值的結束條件，多次執行執行語句。
反覆執行直到條件式的結果變為真 （TRUE）為止。

1. 處理的執行

執行執行語句。

2. 條件的判斷

條件式為真 （TRUE）時，結束處理。
條件式為假 （FALSE）時，重覆處理。

要點

在WHILE語句、REPEAT語句的條件式中，可作以下指定。

- 結果為BOOL值的運算表達式
- BOOL型的变数
- 返回值為BOOL型的函數調用式

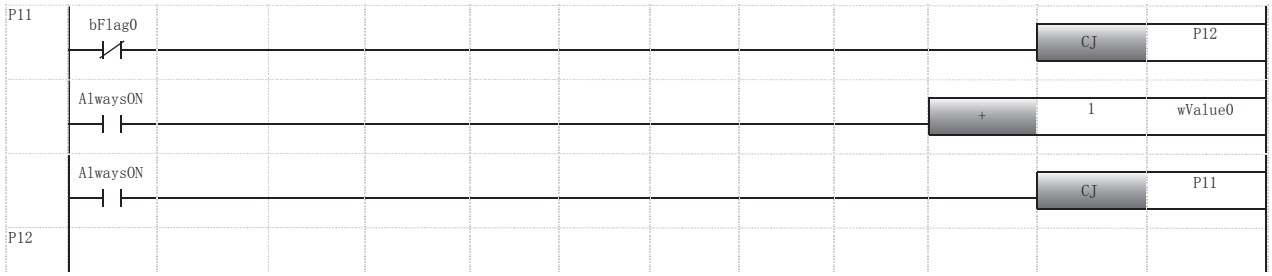
程式範例

bFlag0為TRUE時，wValue0加1。

ST

```
WHILE bFlag0 DO
    wValue0 := wValue0 + 1;
END_WHILE;
```

LD



通過將邏輯運算 （AND、OR等）和比較 （<、>、 =等）的組合表達式用于循環語句的條件式，可表述複雜的結束條件的程式。

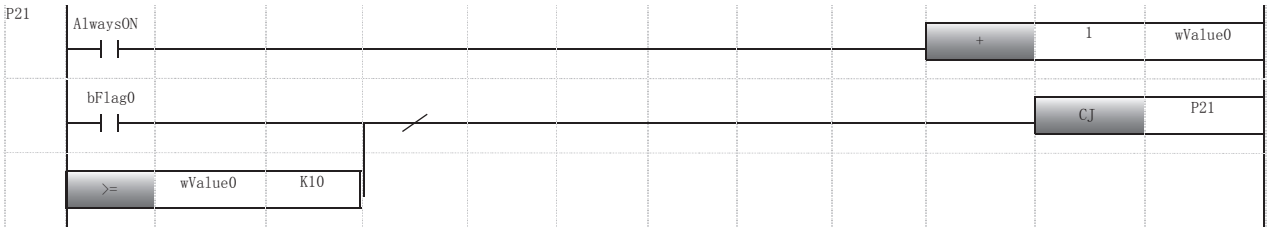
程式範例

bFlag0為TRUE或wValue0小于10時，wValue0加1。

ST

```
REPEAT
    wValue0 := wValue0 + 1;
    UNTIL bFlag0 OR (wValue0 >= 10)
END_REPEAT;
```

LD



重覆執行直到变数達到設定值 （FOR）

整數型变数在達到條件值之前重覆執行的處理可以使用FOR語句進行表述。

FOR語句執行以下處理。

```
FOR 〈變數〉 :=〈初始值（表達式）〉
  TO 〈最終值（表達式）〉
  BY 〈增加值（表達式）〉          —BY（---線內）
                                  可省略
  DO 〈執行語句〉;

END_FOR;
```

根據整數值的結束條件，多次執行執行語句。

設置了初始值的整數型变数在達到最終值之前，重覆執行。

1. 資料的初始化

對作為條件的变数設置初始值。

2. 條件的判斷

变数為最終值時，結束處理。

3. 處理的執行

執行執行語句。

4. 增加值的加法運算

在变数上加上增加值，重覆處理。

要點

FOR語句的初始值、最終值及增加值可指定如下。

- 結果為整數 （INT）值的運算表達式
- 整數 （INT）型的变数
- 返回值為整數 （INT）型的函數調用式

應轉換資料類型以使值變為整數型。

增加值的加法運算處理 （BY〈增加值 （表達式）〉;）在增加值為1時，表述可以省略。

設置為條件的变数在FOR語句結束後，將保持結束時的值。

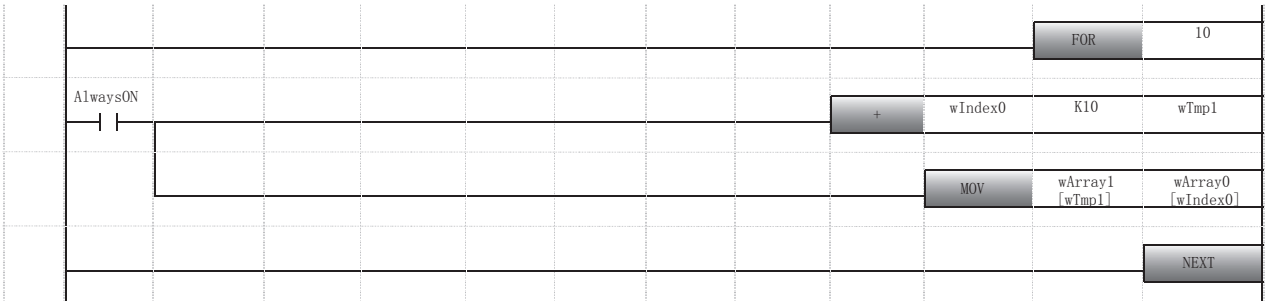
程式範例

在數組wArray0的元素0～10中設置數組wArray1的元素10～20的值。

ST

```
FOR wIndex0 := 0 TO 10 BY 1 DO
    wArray0[ wIndex0 ] := wArray1[ wIndex0 + 10 ];
END_FOR;
```

LD



要點

將數組型資料和循環語句結合，可以對數組的多個元素執行同一資料處理。

4 多種資料類型的處理

本章對在ST語言中處理各資料類型的变数時的注意事項、指定/轉換資料類型進行表述的方法進行說明。
標簽的資料類型在登錄標簽時定義。軟元件的資料類型視軟元件類型而定。

要點

要以不同于定義的資料類型使用值時，應使用通用函數的類型轉換函數。

4.1 BOOL值

BOOL是表述0和1的1位資料類型。
用于表示位軟元件的ON/OFF、執行結果的真 （TRUE）/假 （FALSE）。
位軟元件、設置了“位”的資料類型的標簽按BOOL值的变数處理。

4.2 整數和實數

字軟元件、設置了整數型或實數型的資料類型的標簽按整數或實數的变数處理。
各處理中使用的值的基本資料類型為整數型。要處理小數點以下的值時，使用實數型的变数。

值的範圍

根據变数的不同類型，其有效位數也不同。應根據要執行的運算，指定合適的資料類型的变数。
以下所示為各資料類型的变数在運算中可處理的值的範圍。

資料類型			值的範圍
整數	字[無符號]/位列[16位]	WORD	0～65535
	雙字[無符號]/位列[32位]	DWORD	0～4294967295
	字[有符號]	INT	-32768～32767
	雙字[有符號]	DINT	-2147483648～2147483647
實數	單精度實數	REAL	-2^{128}～-2^{-126}、0、2^{-126}～2^{128}
	雙精度實數	LREAL	-2^{1024}～-2^{-1022}、0、2^{-1022}～2^{1024}

自動進行的類型轉換

在GX Works3的ST語言中，整數或實數型的情況下，在以下處理中即使使用了不同資料類型的变数或常數，也會自動進行類型轉換。

- 賦值語句
- 向函數或FB傳遞輸入參數
- 算術運算式

向大于可處理值的範圍的資料類型轉換時，執行自動類型轉換。

例

代入字[有符號]（INT型）变数和雙字[有符號]（DINT型）变数時

ST	運算結果	
dValue0 := wValue1;	○	INT向DINT的類型轉換會自動進行。
wValue1 := dValue0;	×	DINT向INT的轉換可能會造成資料丟失，因此轉換時會出現錯誤。
wValue1 := DINT_TO_INT(dValue0);	○	DINT向INT的轉換需使用類型轉換函數。 轉換前的值超出INT型的範圍時，會出現運算錯誤。

要點

以下情況時，不會自動進行類型轉換。應使用類型轉換函數。

- 資料大小相同、符號不同的整數型之間的類型轉換
- 向資料丟失類型的類型轉換
- 與整數型、實數型以外的類型轉換

可自動轉換的資料類型

自動類型轉換的資料類型組合如下所示。

轉換前的資料類型	轉換後的資料類型	備注
字[有符號]	雙字[有符號]	自動轉換為符號擴展的值。
	單精度實數	自動轉換為與轉換前相同的值。
	雙精度實數	
字[無符號]/位列[16位]	雙字[有符號]	自動轉換為零擴展的值。
	雙字[無符號]/位列[32位]	
	單精度實數	自動轉換為與轉換前相同的值。
	雙精度實數	
雙字[有符號] 雙字[無符號]/位列[32位]	雙精度實數	
單精度實數		

在向通用函數、通用FB及陳述式傳遞輸入參數時，也會自動進行類型轉換。

輸入參數被定義為泛型資料類型時的自動類型轉換資料類型的組合如下所示。

參數所指定的变数（轉換前）的資料類型	輸入參數的資料類型定義	轉換後的資料類型
字[有符號]	ANY32、ANY32_S	雙字[有符號]
	ANY_REAL、ANY_REAL_32	單精度實數
	ANY_REAL_64	雙精度實數
字[無符號]/位列[16位]	ANY32、ANY32_U	雙字[無符號]/位列[32位]
	ANY_REAL、ANY_REAL_32	單精度實數
	ANY_REAL_64	雙精度實數
雙字[有符號] 雙字[無符號]/位列[32位]	ANY_REAL、 ANY_REAL_64	雙精度實數
單精度實數	ANY_REAL_64	

算術表達式的運算結果的資料類型

ST語言的算術運算表達式（四則運算）的運算結果與運算物件的变数或常数的數量類型相同。（冪乘（**）時，運算結果為實數型。）

要點

運算符的左邊和右邊的資料類型不同時，運算結果按容量較大一側的資料類型處理。

• 運算結果的資料類型的優先級（从高到低）：雙精度實數、單精度實數、雙字、字整數的二元運算中，[有符號]和[無符號]不能同時存在。

注意事項

運算結果超出了資料類型所能處理的值的範圍時，在以後的處理中將無法反映正確的結果（數值）。應事先將運算物件变数的資料類型轉換成可對運算結果進行處理的範圍的資料類型。

例

字[有符號]（INT型）的算術運算時

ST	運算結果	
dValue0 := wValue1 * 10;	×	運算結果超出INT型的範圍（-32768～32767）時，會代入上溢或下溢的運算結果。
dValue1 := INT_TO_DINT(wValue1); dValue0 := dValue1 * 10;	○	運算是以DINT型進行的，因此不會發生上溢或下溢。
dValue1 := INT_TO_DINT(wValue1) * 10;	○	

整數和實數的除法運算

根據变数的類型，除法運算的運算結果可能會有所不同。

整數型变数的除法運算會捨去小數點以下的部分。

實數型变数的除法運算會求取至小數點以下的有效位為止。

• 單精度實數： 有效位數7位（小數點以下6位）

• 雙精度實數： 有效位數15位（小數點以下14位）

例

將表達式“(2 ÷ 10) × 10”的運算結果代入D0時

資料類型		ST	運算結果
整數	字[有符號]	D0 := (2 / 10) * 10;	0
實數	單精度實數	D0:E := (2.0 / 10.0) * 10.0;	2.0

要點

ST語言中，對于字軟元件，可在“:E”等的類型指定中明確指定資料類型進行使用。

（☞88頁 字軟元件的類型指定）

除法運算的餘數（MOD）

整數除法運算的餘數可通過餘數運算（MOD）運算符求取。

例

求5位整數的後2位時

資料類型		ST	運算結果
整數	字[有符號]	D0 := (-32368 MOD 100);	-68

4.3 字元串

字元串型的变数的處理使用字元串處理陳述式和通用函數 （字元串函數）。

程式範例

求字元串的長度。

ST

```
wLen0 := LEN(sString0);
```

字元串的代入

字元串型变数可以使用賦值語句進行表述。

程式範例

在字元串型变数sString0和字元串[Unicode]型变数wsString1中代入字元串“ABC”。

ST

```
sString0 := 'ABC';
wsString1 := "ABC";
```

要點

ASCII字元串常數使用單引號 （’ ）括起來。
Unicode字元串常數使用雙引號 （” ）括起來。

字元串的比較

字元串型变数的比較運算 （比較、一致、不一致）可以使用運算符進行表述。

程式範例

比較字元串，不一致的情況下將sString1代入sString0。

ST

```
IF sString0 <> sString1 THEN
    sString0 := sString1;
END_IF;
```

要點

字元串型变数的情況下，比較運算符 （<、 >、 <=、 >=）會以ASCII或Unicode的編碼值進行比較。
按第一個不同的字元編碼的大小，確定字元串的大小。
比較條件的詳細內容與字元串比較陳述式 （LD$<等）相同。

4.4 時間

時間的資料按時間型变数或時鐘資料 （CPU模組用陳述式所使用的數組資料）處理。

時間型 （TIME）变数

使用時間型变数，能夠以ms單位簡潔明了地用時間資料進行程式。
時間型的常数按以下格式進行表述。

• T#23d23h59m59s999ms

時間單位	d(日)	h(時)	m(分)	s(秒)	ms(毫秒)
範圍	0～24	0～23	0～59	0～59	0～999

時間型变数可在T#-24d20h31m23s648ms～T#24d20h31m23s647ms的範圍內進行設置。
關於時間長度的記述方法的詳細內容，請參閱以下內容。
☞91頁 時間長度的記述方法

時間的代入

時間型变数可以使用賦值語句進行表述。

程式範例

將1小時30分的值代入時間型变数tmData0中。

ST

```
tmData0 := T#1h30m;
```

時間的比較

時間型变数的比較運算 （比較、一致、不一致）可以使用運算符進行表述。

程式範例

時間為1天以上時，結束處理。

ST

```
IF tmData0 >= T#1d THEN
    RETURN;
END_IF;
```

時間的四則運算

時間型变数的乘法運算、除法運算使用通用函數 （時間資料型函數）進行運算。
加減法運算可用運算符進行表述。

程式範例

在時間型变数tmData0上加上10ms，乘以2並將所得值代入tmData0中。

ST

```
tmData0 := MUL_TIME(tmData0 + T#10ms, 2);
```

時鐘資料（日期資料、時間資料）

時鐘資料是CPU模組用陳述式所使用的數組資料。

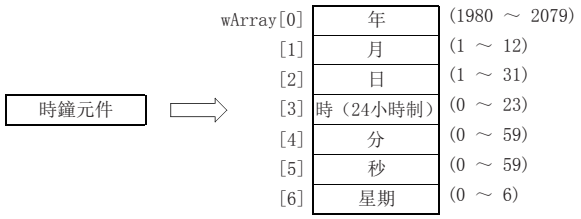
- 時鐘資料： 存儲年、月、日、時、分、秒及星期的數組
- 擴展時鐘資料： 在時鐘資料中增加了毫秒單位的數組
- 日期資料： 僅時鐘資料的年、月、日的數組
- 時間資料： 僅時鐘資料的時、分、秒的數組

時鐘用陳述式和時鐘資料

使用應用陳述式的時鐘用陳述式，可對時鐘資料（字[有符號]資料的數組）進行處理。

程式範例

从CPU模組的時鐘元件中讀取“年、月、日、時、分、秒、星期”。



ST

```
DATERD( TRUE, wArray);
```

日期資料、時間資料的比較

ST中不能使用日期資料、時間資料的比較陳述式。（☞94頁 ST中無法使用的陳述式）

要對日期資料、時間資料進行比較時，應對數組的各元素進行比較或轉換為時間型变数後進行比較。

4.5 數組和結構體

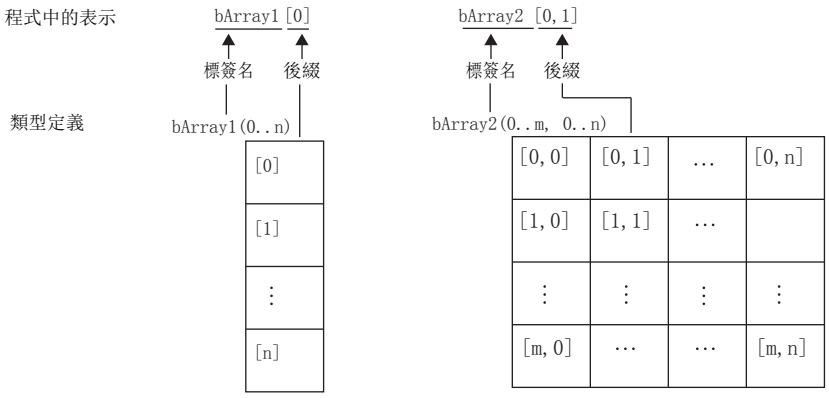
數組和結構體是將多個資料作為一個整體處理的資料格式。

- 數組： 同一資料類型的变数的連續集合。
- 結構體： 不同資料類型的变数的集合。

數組

表示數組型变数整體時，僅使用標簽名。

要表示數組的各元素時，在標簽名後面附上“[ ]”，並在“[ ]”內填上要指定的元素編號。



數組的代入

將數組型变数記述在賦值語句中時，值將會代入附帶元素編號所指定的元素中。

如果沒有附帶元素編號，則可以對數組整體進行賦值 （數組的複製）。

程式範例

在字[有符號]資料的數組型变数wArray1中代入以下內容。

- 元素0： 10
- 元素1： 二維數組wArray2的元素[0,1]

ST

```
wArray1[0] := 10;
wArray1[1] := wArray2[0,1];
```

程式範例

將wArray0的所有元素代入字[有符號]資料的數組型变数wArray1的所有元素中。

（衹有在右邊和左邊的數組的資料類型及資料數完全一致時才可實施。）

ST

```
wArray1 := wArray0;
```

數組的表達式

數組的各元素可以作為定義了資料類型的变数由運算表達式進行指定。（可以進行比較或四則運算等。）

沒有附帶元素編號的數組变数不能用運算表達式進行指定。

ST語言中，可以用表達式來表述數組元素。

程式範例

在字[有符號]資料的數組型变数wArray1中代入以下內容。

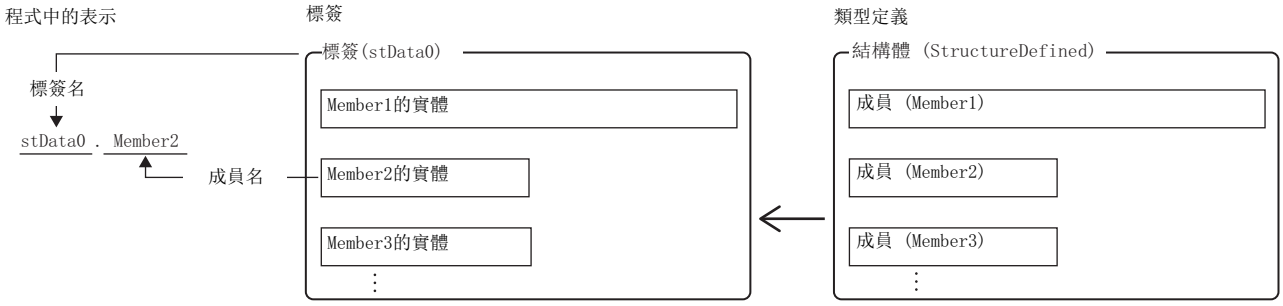
- 元素wIndex0+1：wArray0的元素0和元素1的和

ST

```
wArray1[wIndex0 +1] := wArray0[0] + wArray0[1];
```

結構體

結構體是將多個不同資料類型彙聚起來，作為1個資料類型定義的集合。
結構體的類型定義的名稱及各成員的名稱均可以任意設置，因此非常有利于資料管理。
表示結構體型变数整體時，僅使用標簽名。
要表示結構體的各成員時，在標簽名後面加上“.”再加上成員名。



結構體的代入

將結構體型变数記述在賦值語句中時，值將會代入指定成員中。
如果沒有指定成員，則可以對結構體整體進行賦值 （結構體的複製）。

程式範例

在結構體型变数stData0的成員中代入以下內容。

- Member1(單精度實數)：10.5
- Member2(位)：TRUE

ST

```
stData0.Member1 := 10.5;
stData0.Member2 := TRUE;
```

程式範例

在結構體型变数stData1的所有元素中代入stData0的所有元素。
（僅在右邊和左邊的結構體的資料類型 （結構體型定義）完全一致時才可實施。）

ST

```
stData1 := stData0;
```

組合了結構體和數組的資料類型

ST程式中也可以使用成員中包含了數組的結構體或結構體型的數組。

程式範例

在結構體型变数stData0的數組型成員wArray中代入數組型变数wArray1。（均為字[有符號]、元素數3的數組）

ST

```
stData0.wArray := wArray1;
```

程式範例

對結構體數組型变数stArray0的元素編號1的成員和元素編號0的成員的值進行比較。

ST

```
bResult := stArray0[1].Member1 > stArray0[0].Member1;
```

要點

GX Works3還可進行結構體數組的代入。

5 使用ST表述梯形圖

本章對使用ST語言表述梯形圖程式的梯形圖符號或順序陳述式等的方法進行說明。

5.1 表述觸點、線圈

在ST語言中，對使用了梯形圖的觸點、線圈的回路可以用邏輯與 （AND）、邏輯或 （OR）、邏輯非 （NOT）等邏輯運算和賦值語句進行表述。

常開觸點和線圈

常開觸點和線圈的程式可以用賦值語句進行表述。

程式範例

根據bFlag0的ON/OFF，bResult0進行ON/OFF。

ST

```
bResult0 := bFlag0;
```

LD

要點

賦值語句通過:=表述。將右邊的表達式的結果代入左邊的变数。

常閉觸點 （NOT）

常閉觸點可以用邏輯非 （NOT）的運算符進行表述。

程式範例

bFlag0為ON時，bResult0為OFF，bFlag0為OFF時，bResult0為ON。

ST

```
bResult0 := NOT bFlag0;
```

LD

串聯連接、並聯連接 （AND、OR）

梯形圖中以串聯連接、並聯連接表示的條件，可以用邏輯與 （AND、&）、邏輯或 （OR）的運算符進行表述。

程式範例

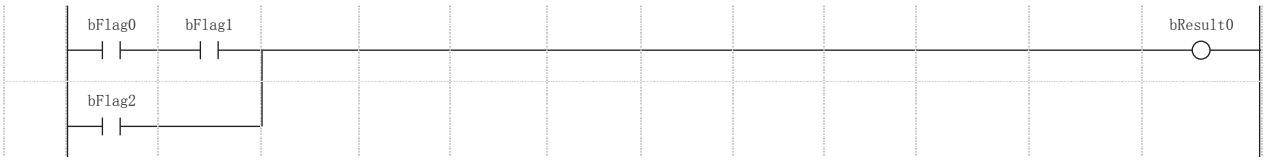
以下任一條件成立時，bResult0為ON。

- 條件1： bFlag0為ON且bFlag1為ON
- 條件2： bFlag2為ON

ST

```
bResult0 := bFlag0 AND bFlag1 OR bFlag2;
```

LD



要點

用1個語句彙總表述多個運算表達式時，將从優先級最高的運算符開始處理。

- 邏輯運算符的優先級 （从高到低）：邏輯與(AND、&)、邏輯異或(XOR)、邏輯或(OR)

有多個優先級相同的運算符時，从最左邊的運算符開始運算。

執行順序比較複雜的觸點、線圈

ST語言還可用于表述複雜的觸點線圈的組合處理。執行順序難以理解時，應使用小括號()，以使優先級簡潔明了。

程式範例

以下條件1及條件2成立時，bResult0為ON。

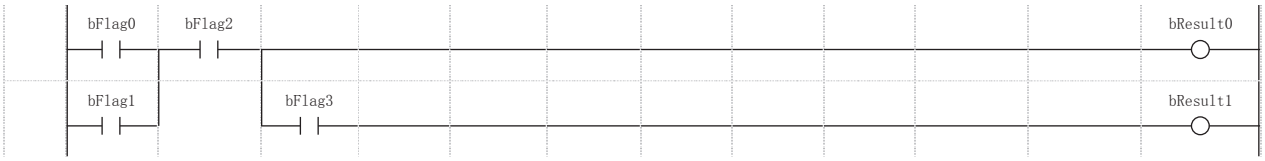
以下條件1、條件2及條件3都成立時，bResult1為ON。

- 條件1： bFlag0或bFlag1為ON
- 條件2： bFlag2為ON
- 條件3： bFlag3為ON

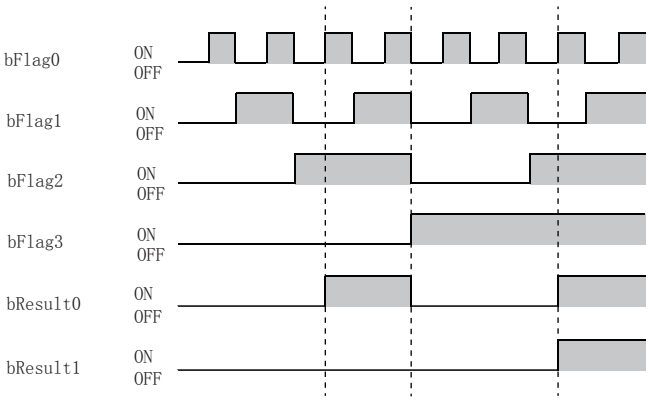
ST

```
bResult0 := (bFlag0 OR bFlag1) AND bFlag2;
bResult1 := bResult0 AND bFlag3;
```

LD



執行了上述程式時，各軟元件的ON/OFF時序如下所示。



5.2 陳述式表述

ST語言中，將可以與梯形圖通用的陳述式作為函數處理。
ST語言所不支持的部分陳述式（☞ 94頁 ST中無法使用的陳述式）可以使用運算符等ST語言特有的格式進行表述。

梯形圖和ST均可以使用的陳述式

原則上，各陳述式在梯形圖和ST中均可以使用。（☞ 12頁 可使用陳述式和函數）
根據各陳述式的定義，作為函數進行表述。

程式範例

bFlag0為OFF→ON時，bResult0為1次掃描ON。

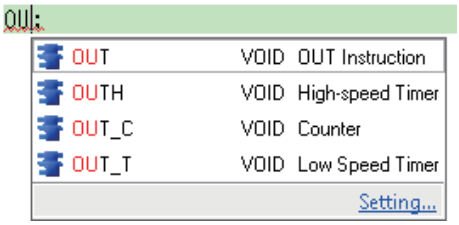
ST

```
PLS(bFlag0, bResult0);
```

LD

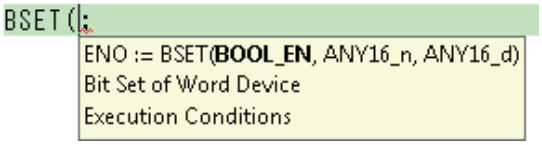
輸入陳述式名時的注意事項

在梯形圖和ST中，有些陳述式雖然功能相同，但陳述式名卻不同。（輸出陳述式的低速定時器陳述式“OUT_T”等）
在ST編輯器中輸入陳述式名後，會一覽顯示以該字元開始的陳述式。
應確認後輸入可在ST中使用的陳述式名。



輸入參數時的注意事項

相同的陳述式在梯形圖和在ST中的參數的順序可能會不同。
在ST編輯器中輸入參數時，會通過工具提示的方式顯示陳述式格式。
應根據工具提示，輸入參數。



關于陳述式的詳細內容，請參照程式手冊。光標停留在陳述式上時按F1後，會顯示該陳述式的頁面。

賦值可使用的陳述式

通用函數的類型轉換函數和賦值語句可使用資料轉換陳述式。
字元串傳送陳述式 （$MOV）等可以使用字元串型標簽的賦值語句進行表述。
☞94頁 賦值可使用的陳述式

程式範例

將單精度實數eValue0轉換成有符號BIN16位資料wValue1。

ST

```
wValue1 := REAL_TO_INT( eValue0 );
```

LD

程式範例

將字元串型变数sString0傳送 （代入）至sString1。

ST

```
sString1 := sString0;
```

LD

可以使用運算符表述的陳述式

基本陳述式的比較運算陳述式、算術運算陳述式可以使用運算符進行表述。
☞94頁 運算符可使用的陳述式

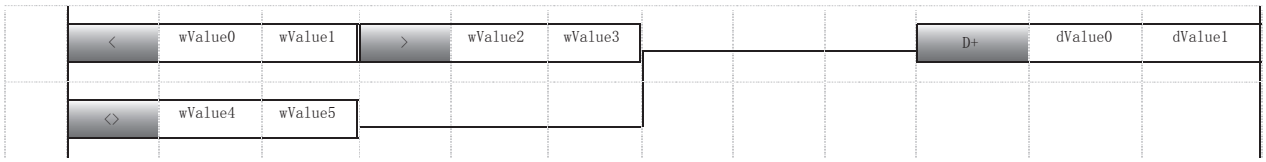
程式範例

根據字[有符號]（INT型）变数的比較結果，加上雙字[有符號]（DINT型）变数的2個值。

ST

```
IF (wValue0 < wValue1 ) AND (wValue2 > wValue3 ) OR (wValue4 <> wValue5 ) THEN
    dValue1 := dValue0 + dValue1;
END_IF;
```

LD



可以使用控制語法、FUN/FB表述的陳述式

循環語句 （☞22頁 循環處理）可以使用結構化陳述式FOR～NEXT進行表述。
不能通過子程式陳述式 （CALL等）或指針分支陳述式 （CJ、SCJ、JMP）對指定了指針的程式進行分支。在ST語言中，應通過選擇語句、函數或FB以使程式結構化。
☞96頁 控制語句、函數等可使用的陳述式

要點

ST語言中，無需END陳述式。（☞96頁 不需要的陳述式）

5.3 聲明、注解的表述

梯形圖的聲明、注解作為注釋進行表述。
ST語言的注釋可以在任意位置進行表述，因此在使用上比聲明、注解更具靈活性。

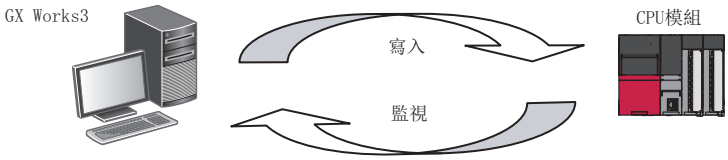
```
(* The processing is performed depending on the ten-key input. *)
IF G_wTenKey <> c_wNONE THEN
        CASE G_wTenKey OF
                0..9 : (* For number-key input (0 to 9) *)
                (* Add the input numeric value to the end of the display value. *)
                IF G_eDecimal = 0.0 THEN
                        (* For integer part *)
                        G_eDisplayValue := (G_eDisplayValue * 10) + G_wTenKey;
                        ELSE
                        (* For after decimal point *)
                        G_eDisplayValue := G_eDisplayValue + (G_eDecimal * G_wTenKey);
                        G_eDecimal := G_eDecimal * 0.1;
                END_IF;
                10: (* For input of decimal point key *)
                G_eDecimal := 0.1;
                11..14: (* For input of addition, subtraction, multiplication, or division key (11 to 14) *)
                (* Retain the operation type *)
                G_wOperation := G_wTenKey - 10;
                (* Move the display value to the previous operation value and then reset the displayed value *)
                G_eLastValue := G_eDisplayValue;
                G_eDisplayValue := 0.0;
                G_eDecimal := 0.0;
                15: (* For equal-key input *)
                (* Add, subtract, multiply, or divide the displayed value to/from the current value. *)
                G_eLastValue := Calculation(G_eLastValue, G_wOperation, G_eDisplayValue);
                (* Assign the rounding result to the display value. *)
                G_eDisplayValue := FractionProcessing(G_eLastValue, G_wSwitch1, G_wSwitch2);
                G_wOperation := 0; (* Clear the operation type *)
                G_eDecimal := 0.0;
        END_CASE;
        (* Clear the key input*)
        G_wTenKey := c_wNONE;
END_IF;
```

6 程式創建步驟

6.1 步驟概要

本節對程式的創建步驟進行說明。

1. 打開ST編輯器。
2. 編輯ST程式。
3. 轉換程式，進行調試。
4. 確認機器的實際動作。



要點

關于GX Works3的操作方法的詳細內容，請參照以下手冊。

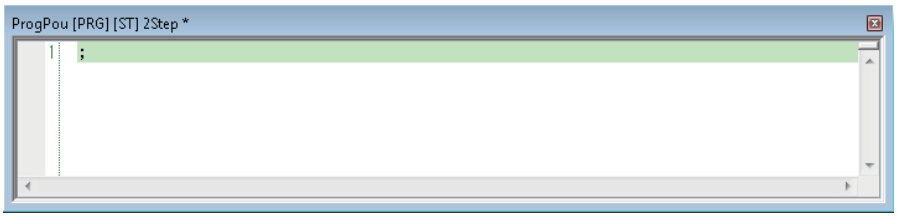
GX Works3操作手冊

6.2 打開ST編輯器

ST程式使用工程工具（GX Works3）的程式功能進行創建。

應在程式語言中選擇“ST”，創建工程或程式。

ST編輯器



6.3 編輯ST程式

對輸入簡單的程式並執行一系列操作的步驟進行說明。

例如：試着輸入以下程式。

程式範例

```
wValue0 := D10 - 123;

IF  wValue0 < 0  THEN
    bFlag0 := TRUE;  (* ON *)
    ELSE
    bFlag0 := FALSE; (* OFF *)
END_IF;

bResult := FunPou(wValue0);
FbPou_1(i_wValue := wValue0,
        o_wValue => wValue1);
```

輸入文本

在ST編輯器中，可通過與一般的文本編輯器相同的操作輸入文本。

操作步驟

```
wValue0 := D10 - 123;
```

1. 輸入“wValue0 := D10 - 123;”。

要點

按Ctrl+Shift+=，可輸入賦值符號 （:=）。

除了通過[Edit （編輯）]菜單進行的操作外，還可通過輸入以下標准鍵進行操作。

- Shift+Delete/Shift+Insert：剪切/粘貼
- Alt+Back space/Ctrl+Back space：撤銷/恢復

編輯器間的複製

可以與一般的文本編輯器等之間進行文本資料的複製/粘貼，可進行如下應用。

- 从其他的文本編輯器或PDF等中複製文本，將其粘貼至ST編輯器中以便引用程式
- 將ST編輯器中創建的程式複製到其他的文本編輯器中，創建檔案

要點

GX Works3中，在ST編輯器上按下Alt的同時拖動時會出現矩形選擇框，可通過該矩形選擇框進行選擇。

輸入控制語法

輸入控制語法 （IF語句）。

操作步驟

```
IF
```

1. 輸入“IF”。

2. 通過Ctrl+F1可顯示語法模板。

```
IF  ?Condition?  THEN
    ?Statement? ;
ELSE
    ?Statement? ;
END_IF;
```

```
IF  wValue0 < 0  THEN
    ?Statement? ;
ELSE
    ?Statement? ;
END_IF;
```

3. 將光標移動到“?條件?”上。
（通過Ctrl+Alt+←可移動光標。）

4. 輸入“wValue0 < 0”作為條件式。

```
IF  wValue0 < 0  THEN
    bFlag0 := TRUE;
ELSE
    bFlag0 := FALSE;
END_IF;
```

5. 將光標移動到“?執行語句?”上。
（通過Ctrl+Alt+→可移動光標。）

6. 輸入“bFlag0 := TRUE;”作為執行語句。

- 即使輸入小寫的“true”，也會自動轉換成大寫的“TRUE”。

同樣，輸入“bFlag0 := FALSE;”。

要點

使用了以下功能。

- [Edit （編輯）]⇨[Display Template （模板顯示）]（Ctrl+F1）
- [Edit （編輯）] ⇨[Mark Template(Left) （模板參數選擇(左)）]/[Mark Template(Right) （模板參數選擇(右)）]（Ctrl+Alt+←/→）

折疊顯示

單擊顯示在控制語法左邊的圖標 （ ⊟ ），可將控制語法折疊顯示 （ ⊞IF...END_IF;）。

輸入注釋

在ST程式中輸入注釋。

操作步驟

```
⊟IF  wValue0 < 0  THEN
     bFlag0 := TRUE;  ON
     ELSE
     bFlag0 := FALSE;
 END_IF;
```

```
⊟IF  wValue0 < 0  THEN
⊟    bFlag0 := TRUE;  (* ON *)
     ELSE
⊟    bFlag0 := FALSE; (* OFF *)
 END_IF;
```

1. 在程式的任意位置 （例如：“bFlag0 := TRUE;”之後），輸入注釋文字。

• 可自由插入空格或TAB。

2. 在注釋文字的前後，需輸入注釋的分割符號（“(*”、“*)”）。

被符號包圍的內容，將作為注釋處理。

同樣，在“FALSE”之後輸入“(* OFF *)”。

要點

GX Works3中，可使用與C語言相同的注釋符號“/**/”和“//”。

行首加上“//”符號，整行即變為注釋。

通過以下功能，可以對選擇範圍以行為單位進行注釋或解除注釋。

• [Edit （編輯）]⇨[Comment Out of Selected Range （選擇範圍的注釋化）] （Ctrl+Shift+S）

• [Edit （編輯）]⇨[Disable Comment Out of Selected Range （選擇範圍的注釋解除）] （Ctrl+Shift+D）

```
⊟IF  wValue0 < 0  THEN
⊟    bFlag0 := TRUE;  (* ON *)
 //  ELSE
 //  bFlag0 := FALSE; (* OFF *)
 END_IF;
```

以行為單位進行注釋的範圍

折疊顯示

單擊顯示在注釋左邊的圖標 （ ⊟ ），可將注釋折疊顯示 （(**)）。

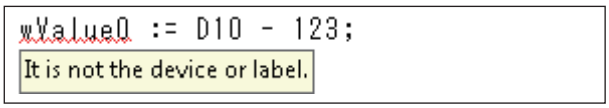
使用標簽

登錄標簽並在ST程式中使用。

登錄標簽

从ST編輯器登錄標簽。

操作步驟

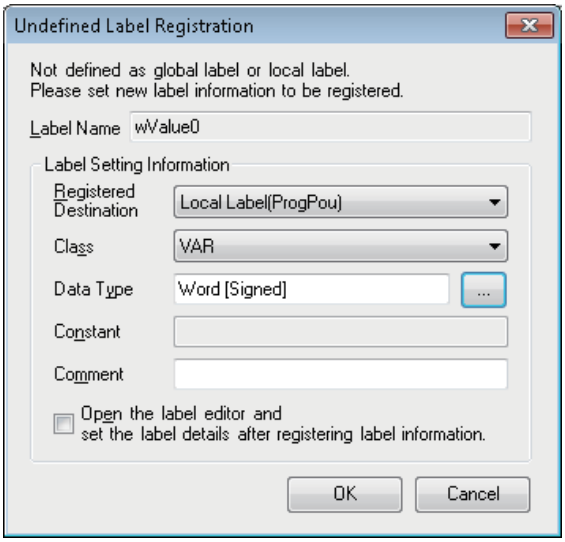


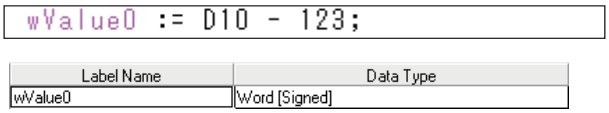
未登錄的標簽名將顯示錯誤位置。

1. 光標移至標簽名上的狀態下按F2，可顯示“Undefined Label Registration （未定義標簽登錄）”畫面。

2. 按如下設置，登錄標簽wValue0。

- 登錄目標標簽： 局部標簽
- 類： VAR
- 資料類型： 字[有符號]





登錄的標簽將以標簽用的顯示顏色顯示。

3. 打開標簽設置畫面，登錄的標簽即被設置。

要點

使用了以下功能。

- [Edit （編輯）]⇨[Register Label （登錄標簽）] （F2）

ST編輯器上的各組態元素的顯示顏色，可通過以下功能進行設置。

- [View （視圖）]⇨[Color and Font （顏色及字體）]

通過標簽編輯器設置

通過標簽編輯器登錄標簽。

操作步驟

Label Name	Data Type
wValue0	Word [Signed]
bFlag0	Bit

1. 顯示“Local Label Setting （局部標簽設置）”畫面。
2. 按如下設置，登錄標簽。
   - 標簽名： bFlag0
   - 資料類型： 位
   - 類： VAR （自動設置。）
3. 打開ST編輯器後，登錄的標簽會以標簽用顯示顏色顯示。

```
IF  wValue0 < 0  THEN
    bFlag0 := TRUE;  (* ON *)
    ELSE
    bFlag0 := FALSE; (* OFF *)
END_IF;
```

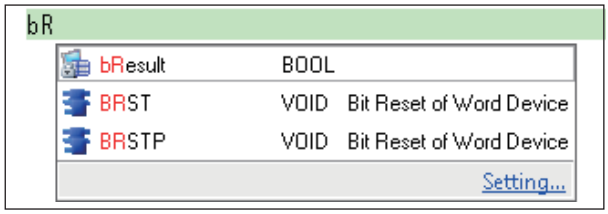
作為在程式例中使用的標簽，對局部標簽作如下設置。

標簽名	資料類型	類
wValue0	字[有符號]	VAR
wValue1	字[有符號]	VAR
bFlag0	位	VAR
bResult	位	VAR

在ST編輯器中使用已登錄的標簽

从已登錄在標簽編輯器的標簽中選擇，在ST編輯器中輸入標簽。

操作步驟



1. 事先登錄位型標簽bResult。
2. 輸入起始字元“bR”後，會一覽顯示以該字元開始的已登錄的標簽名等。
3. 通過↓選擇，按Enter後，選擇的標簽名即被輸入。

創建函數、FB

新建在程式中使用的程式部件 （函數及FB）。

創建函數

在工程中新建函數的資料，對函數程式及其局部標簽進行定義。

■新建資料

在工程中新建函數的資料。

操作步驟

1. 選擇[Project （工程）]⇨[Data Operation （資料操作）]⇨[New data （新建資料）]。
2. 在“New （新建資料）”畫面中對以下設置項目進行設置。

資料類型	項目	內容	程式例中的設定值
函數	資料名	函數調用時的識別符。	FunPou
	程式語言	函數程式的表述語言。	ST
	返回值的類型	對執行完成後返回給調用原的返回值的資料類型進行設置。	位
	使用EN/ENO	選擇“Yes （是）”，參數會帶上EN/ENO。 • EN：設置執行條件的BOOL型輸入參數。 • ENO：返回執行結果的BOOL型輸出參數。	否
	添加目標的FUN檔案	設置用于存儲創建的函數的檔案名。	FUNFILE

要點

創建的函數的設置項目可以在“Properties （屬性）”畫面中確認。
通過以下操作，顯示“Properties （屬性）”畫面。
• 在導航窗口中選擇資料，右鍵單擊，選擇快捷菜單⇨[Properties （屬性）]

■設置參數和內部变数

定義參數和在函數中使用的內部变数。
參數和內部变数通過函數的局部標簽進行設置。應在標簽設置畫面中作如下設置。

標簽名	資料類型	類
i_wValue	字[有符號]	VAR_INPUT
bFlag	位	VAR

要點

類設置為“VAR_INPUT”的局部標簽為輸入參數。
類設置為“VAR_OUTPUT”的局部標簽為輸出參數。
函數調用時的參數順序為局部標簽設置時的定義順序。

創建FB

在工程中新建FB的資料，對FB程式及其局部標簽進行定義。

■新建資料

在工程中新建FB的資料。

操作步驟

1. 選擇[Project （工程）]⇨[Data Operation （資料操作）]⇨[New data （新建資料）]。
2. 在“New （新建資料）”畫面中對以下設置項目進行設置。

資料類型	項目	內容	程式例中的設定值
FB	資料名	要定義的FB的資料類型名。	FbPou
	程式語言	FB程式的表述語言。	ST
	使用EN/ENO	選擇“Yes （是）”，參數會帶上EN/ENO。 • EN：設置執行條件的BOOL型輸入參數。 • ENO：返回執行結果的BOOL型輸出參數。	否
	FB的類型	設置FB的程式本體的存儲目標。可選擇宏類型或子程式類型。	子程式類型
	添加目標的FB檔案	設置用于存儲創建的FB的檔案名。	FBFILE

要點

創建的FB的設置項目可以在“Properties （屬性）”畫面中確認。

通過以下操作，顯示“Properties （屬性）”畫面。

• 在導航窗口中選擇資料，右鍵單擊，選擇快捷菜單⇨[Properties （屬性）]

■設置參數和內部變數

定義參數和在FB中使用的內部變數。

參數和內部變數通過FB的局部標簽進行設置。應在標簽設置畫面中作如下設置。

標簽名	資料類型	類
i_wValue	字[有符號]	VAR_INPUT
o_wValue	字[有符號]	VAR_OUTPUT

要點

類設置為“VAR_INPUT”的局部標簽為輸入參數。

類設置為“VAR_OUTPUT”的局部標簽為輸出參數。

輸入函數

輸入函數。

要點

有返回值的函數可作為表達式處理。（函數調用表達式）
沒有返回值的函數以調用語句表述。（函數調用語句）

```
FunPou(wValue0);  ←— 函數調用語句
```

將函數調用表達式輸入賦值語句

■將返回值代入变数

將返回值代入变数時，函數調用寫在賦值語句的右邊。

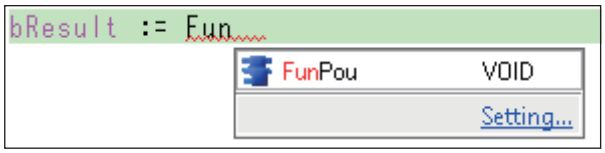
```
bResult :=
```

1. 輸入要代入返回值的变数。
2. 輸入賦值符號“:=”。

■輸入函數名

从已定義的函數中選擇並輸入。

操作步驟

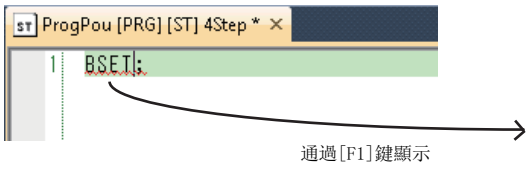



```
bResult := FunPou
```

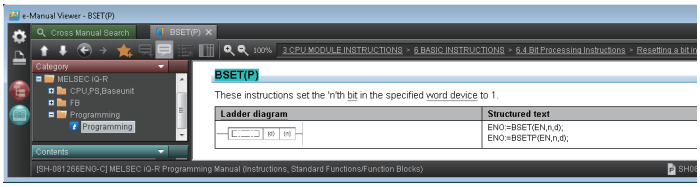
1. 輸入起始字元“Fun”後，會一覽顯示以該字元開始的函數。
2. 通過 ↓ 選擇，按 Enter 後，選擇的函數名即被輸入。

要點

陳述式也可从一覽顯示輸入。（☞ 36頁 陳述式表述）
光標移至陳述式名上的狀態下按 F1，可在e-Manual Viewer中確認陳述式的詳細內容。
（要確認陳述式，需要將相應程式手冊的檔案登錄到e-Manual Viewer中。）



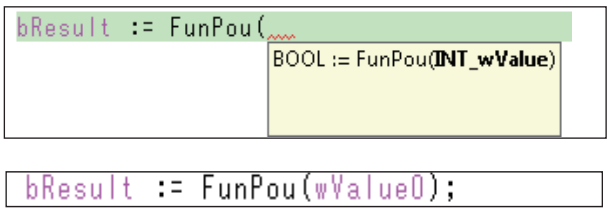
通過[F1]鍵顯示



■輸入參數

根據工具提示，輸入參數。

操作步驟




```
bResult := FunPou(wValue0);
```

1. 輸入“(”後，會在工具提示中顯示陳述式的格式。
2. 根據工具提示，輸入參數。

有多個參數時，用逗號 (,) 隔開。

3. 在參數的末尾輸入“)”。
4. 輸入表示調用語句結束的“;”。

要點

已定義的函數還可以輸入使用了模板顯示的參數。

光標移至函數名上時，通過 Ctrl+F1 可顯示模板。

```
bResult := FunPou( ?INT_wValue? );
```

還可以使用參數選擇功能。

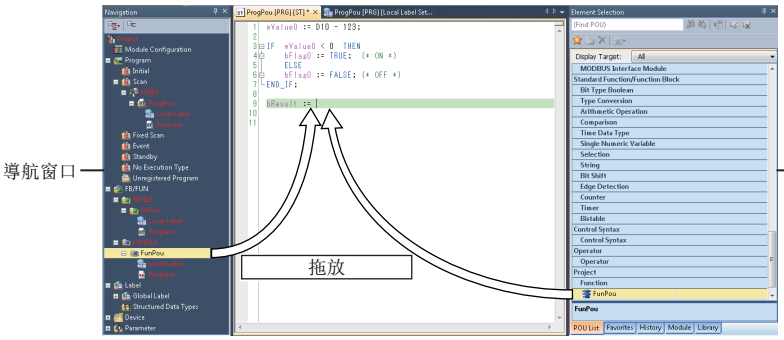
- [Edit (編輯)]⇨[Display Template (模板顯示)] (Ctrl+F1)
- [Edit (編輯)] ⇨[Mark Template(Left) (模板參數選擇(左))]/[Mark Template(Right) (模板參數選擇(右))] (Ctrl+Alt+←/→)

从導航窗口、部件選擇窗口中選擇

可从導航窗口或部件選擇窗口中選擇，輸入函數。

要點

可以从導航窗口或部件選擇窗口拖放至ST編輯器。



輸入FB

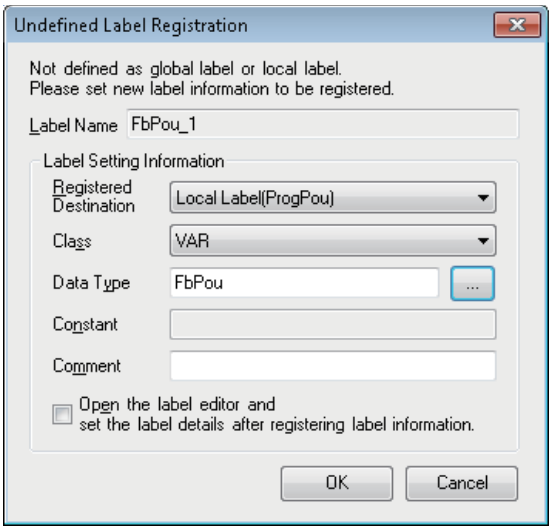
輸入FB。

輸入FB調用語句

■輸入FB的實例名

輸入已定義的FB。

操作步驟



1. 輸入實例名“FbPou_1”（任意）。
2. 光標移至實例名上的狀態下按F2，可顯示“Undefined Label Registration（未定義標簽登錄）”畫面。
3. 進行如下設置，登錄實例。
   - 登錄目標標簽： 局部標簽
   - 類： VAR
   - 資料類型： FbPou

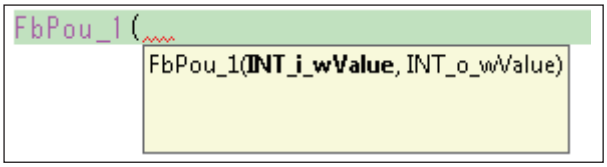
選擇資料類型時，如果類型分類選擇了“FB”，則可以選擇已定義的FB。

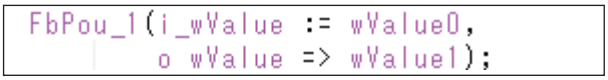
登錄的實例名將以標簽用的顯示顏色顯示。

■輸入參數

根據工具提示，輸入參數。

操作步驟





1. 輸入“(”後，會在工具提示中顯示FB的格式。
2. 根據工具提示，輸入參數。

有多個參數時，用逗號（,）隔開。

3. 在參數的末尾輸入“)”。
4. 輸入表示調用語句結束的“;”。

要點

已定義的FB還可以輸入使用了模板顯示的參數。

光標移至實例名上時，通過Ctrl+F1可顯示模板。

```
FbPou_1(i_wValue:= ?INT? ,o_wValue=> ?INT? );
```

還可以使用參數選擇功能。

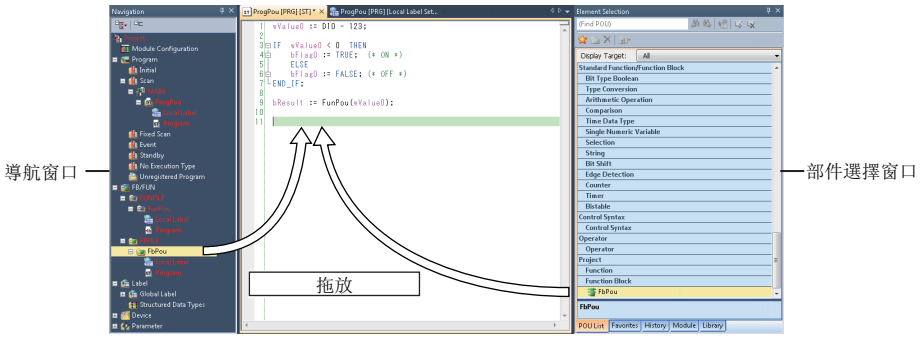
- [Edit（編輯）]⇨[Display Template（模板顯示）]（Ctrl+F1）
- [Edit（編輯）]⇨[Mark Template(Left)（模板參數選擇(左)）]/[Mark Template(Right)（模板參數選擇(右)）]（Ctrl+Alt+←/→）

从導航窗口、部件選擇窗口中選擇

可从導航窗口或部件選擇窗口中選擇，輸入函數FB。

要點

可以从導航窗口或部件選擇窗口拖放至ST編輯器。



6.4 轉換程式、調試

創建的程式必須轉換成可在可程式控制器的CPU模組中執行的代碼 （執行程式）。
如果程式中有不正確的表述，則會在轉換時被檢查出來。應根據顯示的資訊修改程式。

轉換程式

將創建的程式轉換成可執行代碼

操作步驟

1. 執行[Convert （轉換）]⇨[Convert （轉換）] （ F4 ）。

要點

執行轉換 （F4）時，僅對添加、更改的部分進行轉換。
要對包括已轉換的程式在內的所有程式進行轉換時，應執行以下操作。
• [Convert （轉換）]⇨[Rebuild All （全部轉換）] （Shift+Alt+F4）

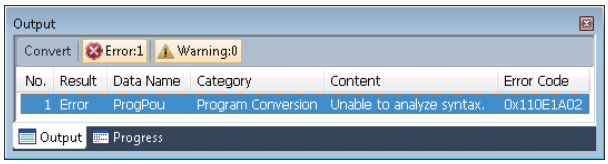
確認錯誤/警告

如果程式中有不正確的表述，則會在轉換時被檢查出來，並會顯示錯誤/警告的資訊。
例如：試着轉換以下程式。

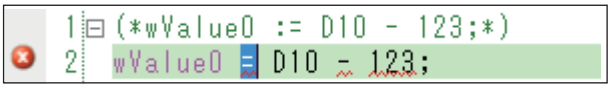
程式範例

```
1  (*wValue0 := D10 - 123;*)
2  wValue0 = D10 - 123;          <-- 賦值符號不同
3
4  IF  wValue0 < 0  THEN
5      bFlag0 := TRUE;  (* ON *)
6      ELSE
7      bFlag0 := FALSE; (* OFF *)
8  (*END_IF;*)
9  END_IF                        <-- 語句結尾沒有“;”
10
11 (*bResult := FunPou( wValue0 );*)
12 FunPou( wValue0 );            <-- 沒有使用返回值
13 FbPou_1(i_wValue := wValue0,
14     (*  o_wValue => wValue1);*)
15         o_wValue := wValue1);  <-- 輸出变数為“:=”
```

操作步驟



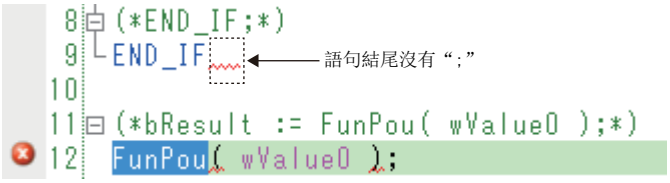
1. 雙擊輸出窗口中顯示的錯誤/警告資訊。



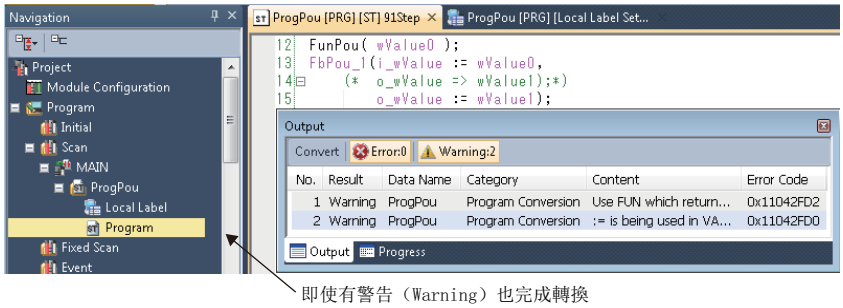
2. 跳轉到錯誤行。根據資訊修改程式。

要點

如果無法對語法作出分析，應確認跳轉行的前後語句。
錯誤處以下劃線顯示。

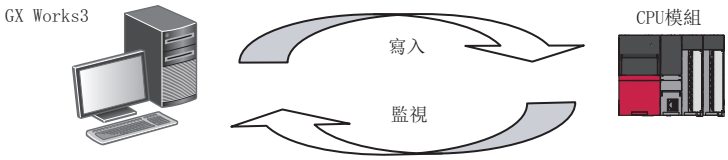


僅含警告（Warning）級別的資訊時，會完成轉換。



6.5 確認機器的實際動作

將轉換後的執行程式寫入到可程式控制器的CPU模組中後執行。監視執行中的程式，確認動作是否與預期一致。



在可程式控制器中執行程式

以下所示為在CPU模組中執行執行程式的步驟。

操作步驟

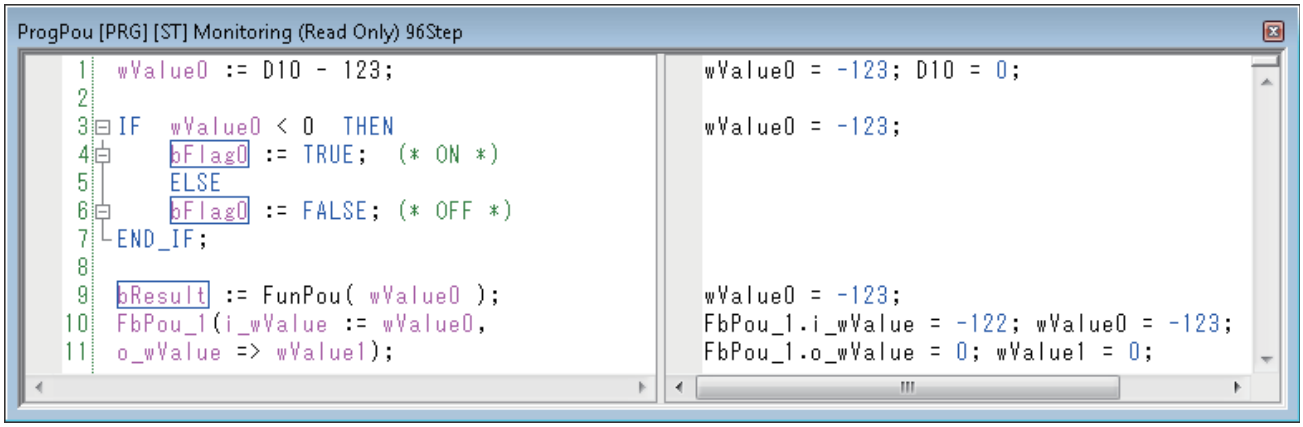
1. 將計算機連接至CPU模組，通過工程工具（GX Works3）設置連接目標。
2. 將CPU模組的動作狀態置為STOP。
3. 選擇[Online（在線）] ⇨[Write to PLC（寫入至可程式控制器）]，寫入“Program（程式）”。
   - 應根據系統或軟元件設置等的變更，寫入“Parameter（參數）”。
   - 使用了全局標簽時，應寫入“Global Label Setting（全局標簽設置）”。
   - 程式中使用了設置有初始值的標簽/軟元件時，應寫入“Label Initial Value（標簽初始值）”、“Global Label Initial Value（全局標簽初始值）”或“Device Initial Value（軟元件初始值）”。
4. 復位CPU模組。
5. 將CPU模組的動作狀態置為RUN。

確認執行中的程式

本項對在程式編輯器中確認執行中程式的狀態的步驟進行說明。

操作步驟

選擇[Online （在線）]⇨ [Monitor （監視）] ⇨[Start Monitoring （監視開始）] （F3）/[Stop Monitoring （監視停止）] （Alt+F3）。



當前值的顯示

以下所示為在ST編輯器上顯示的監視值。

■位類型

位類型的監視值在程式中顯示如下。

- TRUE: bFlag0
- FALSE: bFlag0

■位類型以外

位類型以外的監視值顯示在分割窗口的右側。

要點

將光標移至軟元件/標簽名上，工具提示中即顯示監視值。

當前值的更改

可通過以下方法更改軟元件或標簽的當前值。

操作步驟

1. 在ST編輯器中選擇要更改當前值的軟元件/標簽。
2. 按Shift+Enter。（也可通過Shift+雙擊執行。）

位類型以外時，登錄至監看窗口。應在監看窗口中更改當前值。

■監看窗口中的當前值更改

可通過以下方法確認、更改登錄在監看窗口中的軟元件/標簽的當前值。

操作步驟

1. 選擇[Online （在線）]⇨ [Watch （監看）] ⇨[Start Watching （監看開始）] （Shift+F3）/[Stop Watching （監看停止）] （Shift+Alt+F3）。
2. 監看中，在“Current Value （當前值）”欄中直接輸入要更改的值。

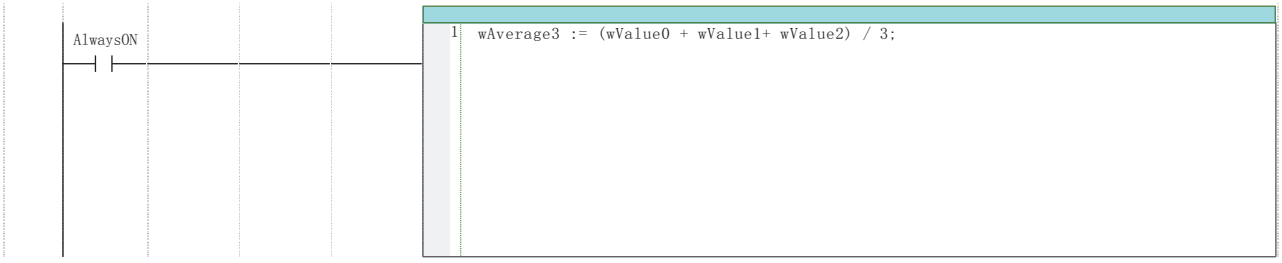
要點

以下任一操作也可以將軟元件/標簽登錄至監看窗口。

- 從ST編輯器拖放到監看窗口
- [Tool （工具）] ⇨[Options （選單）]⇨“Monitor （監視）”⇨“ST Editor （ST編輯器）”⇨“Setting for Automatic Registration to Watch Window （自動登錄至監看窗口的設置）”

6.6 使梯形圖的一部分成為ST （內嵌ST）

要使梯形圖程式的一部分成為ST時，可使用內嵌ST功能。使用了內嵌ST，即可取代梯形圖的陳述式用ST程式進行表述。



操作步驟

1. 選擇[Edit （編輯）]⇨[Inline Structured Text （內嵌ST）]⇨[Insert Inline Structured Text Box （插入內嵌ST框）] （Ctrl+B）。
2. 在內嵌ST框中編輯ST程式。

內嵌ST框中的編輯方法與在ST編輯器中的編輯方法相同。

3. 轉換梯形圖程式。

內嵌ST可作為梯形圖程式的一部分進行編譯 （轉換）。

要點

在梯形圖輸入對話框中輸入“STB”，可以插入內嵌ST框。

備忘錄

第2部分 程式示例

本部分記載了以單純功能為模型的ST程式示例。

7 程式示例的概要

8 計算器處理 （四則運算和條件分支）

9 定位處理 （指數函數、三角函數和結構體）

10 次品分類 （數組和循環處理）

11 運轉時間計測 （時間和字元串）

7 程式示例的概要

以下所示為程式示例的一覽和使用方法。

7.1 程式示例一覽

第2部分以單純功能的應用程式為模型，記載了以下程式示例。

項目	示例內容		參照
計算器處理	四則運算 條件分支	選擇語句 （IF語句） 整數、實數及BOOL值的代入	59頁 初始化程式： Initialization
		選擇語句 （CASE語句） 四則運算	60頁 四則運算 （FUN）：Calculation
		整數的捨去/進位/四捨五入	61頁 化整處理 （FUN）：Rounding
		整數、實數的運算 整數、實數的類型轉換 選擇語句的分層	62頁 尾數處理 （FUN）:FractionProcessing 63頁 計算器程式： Calculator 65頁 含稅計算程式： IncludingTax
定位處理	指數函數 三角函數 結構體	三角函數 （TAN^{-1}）	67頁 根據X、Y坐標計算旋轉角度(FUN)： GetAngle
		結構體型參數 結構體的運算 指數函數 （冪乘、平方根）	68頁 計算X、Y坐標的2點間距離 （FUN）：GetDistance
		三角函數 （COS、SIN） 結構體型返回值	69頁 根據半徑和角度計算X、Y坐標 （FUN）：GetXY
		實數、常數的運算	70頁 電機的陳述式脈衝數的計算 （FB）： PulseNumberCalculation
		結構體的代入 結構體的參數指定	71頁 X、Y坐標的位置控制程式： PositionControl
次品分類	數組 循環處理	二維數組 數組的運算 數組型參數 循環語句和選擇語句的分層	74頁 產品檢查 （FB）：ProductCheck
		包含數組型成員的結構體 結構體數組	75頁 產品資料的分類 （FB）：Assortment
		數組元素的初始化 數組的參數指定 循環語句 （FOR語句、WHILE語句、REPEAT語句）	76頁 產品資料管理程式： DataManagement
運轉時間計測	時間 字元串	時間型資料 定時器的觸點、線圈、當前值	79頁 運轉時間管理程式： OperatingTime
		定時器 BOOL值的運算	80頁 閃爍定時器 （FB）：FlickerTimer 81頁 指示燈的點亮/熄滅程式： LampOnOff
		時鐘資料 （INT型數組資料）	82頁 从秒至時、分、秒的轉換程式： SecondsToTimeArray
		時間型資料 字元串資料	83頁 从時間型至字元串的轉換程式： TimeToString

7.2 通過GX Works3創建程式示例時

本手冊記載的是通過GX Works3創建的ST程式示例。

注意事項

本手冊中記載的程式示例並不保證其實際的動作。
在實際創建程式時，要對應實際系統及要求的運行。
應根據需要，對應所使用的機器，對程式示例中使用的標簽分配軟元件。
應結合所使用的機器及軟元件設置參數。

創建步驟

執行程式示例時，按如下步驟創建程式。

1. 在標簽編輯器中設置全局標簽。（☞ 43頁 通過標簽編輯器設置）

• 使用結構體時，應新建結構體定義並進行設置。

2. 創建程式部件。

• 設置局部標簽。（☞ 43頁 通過標簽編輯器設置）

• 在程式本體中輸入程式示例。（☞ 40頁 編輯器間的複製）

程式內部使用了本手冊中定義的程式部件時，應在同一工程內創建相應的函數、FB。（☞ 44頁 新建資料）

3. 轉換程式。（☞ 50頁 轉換程式、調試）

要點

以PDF或e-Manual格式使用本手冊時，可以通過複製/粘貼的方式方便地引用程式示例。

8 計算器處理 （四則運算和條件分支）

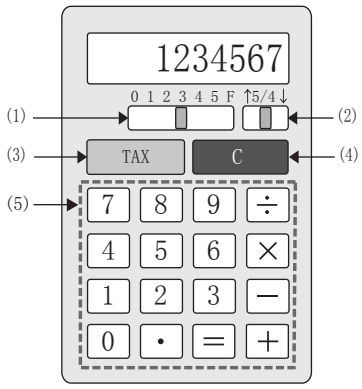
以下所示為在實現市售計算器功能的程式中進行四則運算或條件分支處理的示例。
本程式示例中創建以下所示的程式部件。

資料名	資料類型	內容	參照
Initialization	程式塊	有清除指示時，對变数進行初始化。	59頁 初始化程式： Initialization
Calculation	函數	對2值進行加減乘除運算。	60頁 四則運算 （FUN）: Calculation
Rounding	函數	對整數值進行捨去/進位/四捨五入。	61頁 化整處理 （FUN）: Rounding
FractionProcessing	函數	在指定小數位的下1位進行尾數處理。	62頁 尾數處理 （FUN）:FractionProcessing
Calculator	程式塊	根據輸入，對值繼續運算。	63頁 計算器程式： Calculator
IncludingTax	程式塊	有含稅計算指示時，計算出含稅價格和稅額，顯示含稅價格。	65頁 含稅計算程式： IncludingTax

功能的概要

執行以下處理。

- 輸入了清除鍵時，將對當前值 （到上次為止的計算結果）、顯示值等進行初始化。
- 根據數位鍵的輸入，更新顯示值。小數點以下的值，根據滑動開關的設置進行尾數處理。
- 輸入了含稅計算鍵時，顯示含稅價格。

設備示意圖	編號	名稱		內容
	(1)	小數位指定開關	0～5	指定要顯示的小數點以下的位數。 指定位數以下的值將按化整處理開關指定的方法進行處理。
			浮點數 （F）	不進行尾數處理直接顯示。
	(2)	化整處理開關	進位 （↑）	指定位的下1位為0以外時，指定位+1。
			四捨五入 （5/4）	按指定位的下1位的值進行四捨五入。
			捨去 （↓）	捨去指定位以後的值。
	(3)	含稅計算鍵		計算含稅價格。
	(4)	清除鍵		對當前值 （到上次為止的計算結果）、顯示值進行初始化。
	(5)	數位鍵	0～9、小數點 （.）	輸入數值。
			+、-、*、/	指定四則運算類型。
			=	執行指定的四則運算。

使用的全局標簽

以下所示為本程式示例中使用的全局標簽及結構體定義。
在GX Works3中作如下設置。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	注釋
G_eDisplayValue	單精度實數	VAR_GLOBAL	—	顯示值
G_eLastValue	單精度實數	VAR_GLOBAL	—	當前值 （到上次為止的計算結果）
G_wSwitch1	字[有符號]	VAR_GLOBAL	—	開關的設置值 （0～5： 小數位數/6： 浮點數）
G_wSwitch2	字[有符號]	VAR_GLOBAL	—	開關的設置值 （0：捨去/1：進位/2：四捨五入）
G_bTax	位	VAR_GLOBAL	—	輸入含稅計算鍵
G_bClear	位	VAR_GLOBAL	—	輸入清除鍵
G_wTenKey	字[有符號]	VAR_GLOBAL	—	輸入數位鍵 （0～9： 數值/10： 小數點/11～14： 四則運算/15： =）
G_wOperation	字[有符號]	VAR_GLOBAL	—	運算類型
G_eDecimal	單精度實數	VAR_GLOBAL	初始值：0	小數運算用

■結構體

不使用。

8.1 初始化程式： Initialization

有清除指示時，對变数進行初始化。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	Initialization	ST	初始化程式

程式範例

```
(*有清除指示時，對变数進行初始化。 *)
IF G_bClear THEN
    G_eDisplayValue := 0.0;
    G_eLastValue := 0.0;
    G_wOperation := 0;
    G_eDecimal := 0.0;
    G_bClear := FALSE;
END_IF;
```

要點

首次執行的各变数的值依標簽設置或分配的軟元件的設置而定。

全局標簽可以在工程內所有程式中使用。

IF語句在條件式為真 （TRUE）時執行語句，為假 （FALSE）時不執行。

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	注釋
G_bClear	位	VAR_GLOBAL	—	輸入清除鍵
G_eDisplayValue	單精度實數	VAR_GLOBAL	—	顯示值
G_eLastValue	單精度實數	VAR_GLOBAL	—	當前值 （到上次為止的計算結果）
G_wOperation	字[有符號]	VAR_GLOBAL	—	運算類型
G_eDecimal	單精度實數	VAR_GLOBAL	初始值：0	小數運算用

■局部標簽

不使用。

程式部件

不使用。

8.2 四則運算 （FUN）：Calculation

根據指定的運算類型，對2值進行四則運算。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	返回值的類型	EN/ENO	標題
函數	Calculation	ST	單精度實數	否	四則運算

程式範例

```
(*在值1上加減乘除值2。 *)
CASE i_wOperation OF
    1: (*運算類型為加法運算時： 計算結果=值1+值2*)
    Calculation := i_eValue1 + i_eValue2;
    2: (*減法運算時*)
    Calculation := i_eValue1 - i_eValue2;
    3: (*乘法運算時*)
    Calculation := i_eValue1 * i_eValue2;
    4: (*除法運算時*)
    IF i_eValue2 = 0.0 THEN(*值2為0時，不執行運行。*)
       Calculation := i_eValue1;
       ELSE
       Calculation := i_eValue1 / i_eValue2;
    END_IF;
END_CASE;
```

要點

CASE語句根據整數值的條件，選擇執行語句。條件不一致的語句，不會執行。

選擇語句 （IF語句、CASE語句）可以分層。

变数

在GX Works3中作如下設置，定義標簽。返回值的類型可在函數的屬性中設置。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_eValue1	單精度實數	VAR_INPUT	—	值1
i_wOperation	字[有符號]	VAR_INPUT	—	運算類型
i_eValue2	單精度實數	VAR_INPUT	—	值2

■返回值的類型

識別符	資料類型	內容
Calculation	單精度實數	運算結果

程式部件

不使用。

8.3 化整處理 （FUN）：Rounding

對整數值的1的位進行捨去/進位/四捨五入。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	返回值的類型	EN/ENO	標題
函數	Rounding	ST	雙字[有符號]	否	化整處理

程式範例

```
(*按指定的化整處理方法，對1的位進行處理。 *)
CASE i_wType OF
    0: (* 捨去 *)
    Rounding := i_dValue / 10 * 10; (* 1的位捨去 *)
    1: (* 進位 *)
    Rounding := (i_dValue + 9) / 10 * 10; (* 1的位進位 *)
    2: (* 四捨五入 *)
    Rounding := (i_dValue + 5) / 10 * 10; (* 1的位四捨五入 *);
    ELSE (* 指定值以外時返回輸入值。 *)
    Rounding := i_dValue;
END_CASE;
```

要點

整數的除法運算可以捨去小數點以下的位。

变数

在GX Works3中作如下設置，定義標簽。返回值的類型可在函數的屬性中設置。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_wType	字[有符號]	VAR_INPUT	—	處理方法 （0:捨去/1:進位/2:四捨五入）
i_dValue	雙字[有符號]	VAR_INPUT	—	輸入值

■返回值的類型

識別符	資料類型	內容
Rounding	雙字[有符號]	運算結果

程式部件

不使用。

8.4 尾數處理 （FUN）:FractionProcessing

在指定小數位的下1位進行尾數處理。
在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	返回值的類型	EN/ENO	標題
函數	FractionProcessing	ST	單精度實數	否	尾數處理

程式範例

```
(*在指定小數位的下1位進行尾數處理。 *)
(*位指定超出範圍時，直接返回輸入值。*)
IF (i_wDigits <= 0) OR (i_wDigits > c_wMAX) THEN
    FractionProcessing := i_eValue;
    RETURN; (* 結束處理 *)
END_IF;
(*移動小數點，使指定位變為1的位。 *)
wDigits := i_wDigits + 1;
eValue := i_eValue * (10.0 ** wDigits) - 0.5;
(*對整數的1的位進行指定的化整處理。 *)
dValue := Rounding(i_wType, REAL_TO_DINT(eValue));
(*將值轉換為實數，恢復移動的小數點。 *)
FractionProcessing := DINT_TO_REAL(dValue)/(10.0 ** wDigits);
```

要點

將實數型轉換為整數型時，使用REAL_TO_DINT等的類型轉換函數。
運算表達式的項和參數可以使用類型轉換函數的調用表達式。

变数

在GX Works3中作如下設置，定義標簽。返回值的類型可在函數的屬性中設置。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_eValue	單精度實數	VAR_INPUT	—	輸入值
i_wDigits	字[有符號]	VAR_INPUT	—	顯示的小數位 （小數部分0～5位）
i_wType	字[有符號]	VAR_INPUT	—	處理方法 （0:捨去/1:進位/2:四捨五入）
c_wMAX	字[有符號]	VAR_CONSTANT	常數： 5	小數位指定最大值
wDigits	字[有符號]	VAR	—	處理物件的小數位 （小數部分1～6位）
eValue	單精度實數	VAR	—	實數值 （內部計算用）
dValue	雙字[有符號]	VAR	—	整數值 （內部計算用）

■返回值的類型

識別符	資料類型	內容
FractionProcessing	單精度實數	運算結果

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
Rounding	函數	對整數值進行捨去/進位/四捨五入。	61頁 化整處理 （FUN）：Rounding
REAL_TO_DINT	通用函數	REAL型→DINT型轉換 將REAL型資料轉換成DINT型資料。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)
DINT_TO_REAL	通用函數	DINT型→REAL型轉換 將DINT型資料轉換成REAL型資料。	

8.5 計算器程式：Calculator

根據數位鍵的輸入，更新顯示值。小數點以下的值，根據滑動開關的設置進行尾數處理。
在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	Calculator	ST	計算器程式

程式範例

```
(*輸入了數位鍵後，根據輸入鍵區別處理。*)
IF G_wTenKey <> c_wNONE THEN
    CASE G_wTenKey OF
        0..9 : (* 輸入數位鍵(0～9)時 *)
        (*在顯示值的最後添加輸入數值*)
        IF G_eDecimal = 0.0 THEN
            (* 整數部分時 *)
            G_eDisplayValue := (G_eDisplayValue * 10) + G_wTenKey;
            ELSE
            (* 小數點以下時 *)
            G_eDisplayValue := G_eDisplayValue + (G_eDecimal * G_wTenKey);
            G_eDecimal := G_eDecimal * 0.1;
        END_IF;
        10 : (* 輸入小數點鍵時 *)
        G_eDecimal := 0.1;
        11..14 : (* 輸入加減乘除鍵(11～14)時 *)
        (* 儲存運算類型 *)
        G_wOperation := G_wTenKey - 10;
        (* 將顯示值移動到上次運算值，復位顯示值 *)
        G_eLastValue := G_eDisplayValue;
        G_eDisplayValue := 0.0;
        G_eDecimal := 0.0;
        15: (* 輸入等號鍵時 *)
        (* 在當前值上加減乘除顯示值。 *)
        G_eLastValue := Calculation(G_eLastValue, G_wOperation, G_eDisplayValue);
        (* 將尾數處理的結果代入顯示值。 *)
        G_eDisplayValue := FractionProcessing(G_eLastValue, G_wSwitch1, G_wSwitch2);
        G_wOperation := 0; (* 清除運算類型 *)
        G_eDecimal := 0.0;
    END_CASE;
    (* 清除鍵輸入 *)
    G_wTenKey := c_wNONE;
END_IF;
```

要點

運算符的左邊和右邊的資料類型不同時，運算結果按資料容量較大一側的資料類型處理。（INT型 （字[有符號]）和REAL型 （單精度實數）的運算結果為REAL型 （單精度實數）。）

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	注釋
G_wTenKey	字[有符號]	VAR_GLOBAL	—	輸入數位鍵 （0～9： 數值/10： 小數點/11～14： 四則運算/15： =）
G_eDecimal	單精度實數	VAR_GLOBAL	初始值：0	小數運算用
G_wOperation	字[有符號]	VAR_GLOBAL	—	運算類型
G_eDisplayValue	單精度實數	VAR_GLOBAL	—	顯示值
G_eLastValue	單精度實數	VAR_GLOBAL	—	當前值 （到上次為止的計算結果）
G_wSwitch1	字[有符號]	VAR_GLOBAL	—	開關的設置值 （0～5： 小數位數/6： 浮點數）
G_wSwitch2	字[有符號]	VAR_GLOBAL	—	開關的設置值 （0：捨去/1：進位/2：四捨五入）

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
c_wNONE	字[有符號]	VAR_CONSTANT	常數：-1	無數位鍵輸入

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
Calculation	函數	在當前值上加減乘除輸入值。	60頁 四則運算 （FUN）：Calculation
FractionProcessing	函數	按指定的小數位數選擇捨去/進位/四捨五入，求出值。	62頁 尾數處理 （FUN）:FractionProcessing

8.6 含稅計算程式： IncludingTax

有含稅計算指示時，計算出含稅價格和稅額，顯示含稅價格。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	IncludingTax	ST	含稅計算程式

程式範例

```
(*有含稅計算指示時，顯示含稅價格。 *)
IF G_bTax THEN
    (* 計算稅額。(同時計算小數點以下部分) *)
    eTaxAmount := G_eDisplayValue * c_eTaxRate / 100.0;
    (* 計算含稅價格。(同時計算小數點以下部分) *)
    G_eLastValue := G_eDisplayValue + eTaxAmount;
    (* 將含稅價格設為顯示值。(小數點以下部分四捨五入) *)
    dPrice := REAL_TO_DINT(G_eLastValue);
    G_eDisplayValue := DINT_TO_REAL(dPrice);
    (* 清除鍵輸入 *)
    G_bTax := FALSE;
END_IF;
```

要點

將實數型轉換為整數型時，使用REAL_TO_DINT等的類型轉換函數。

通過REAL_TO_DINT轉換後的資料為將REAL型資料值的小數點以下第1位進行四捨五入後的值。

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	注釋
G_eDisplayValue	單精度實數	VAR_GLOBAL	—	顯示值
G_eLastValue	單精度實數	VAR_GLOBAL	—	當前值 （到上次為止的計算結果）
G_bTax	位	VAR_GLOBAL	—	輸入含稅計算鍵

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
c_eTaxRate	單精度實數	VAR_CONSTANT	常數： 8.0	稅率 （%）
eTaxAmount	單精度實數	VAR	—	稅額
dPrice	雙字[有符號]	VAR	—	含稅價格

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
REAL_TO_DINT	通用函數	REAL型→DINT型轉換 將REAL型資料轉換成DINT型資料。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)
DINT_TO_REAL	通用函數	DINT型→REAL型轉換 將DINT型資料轉換成REAL型資料。	

9 定位處理 （指數函數、三角函數和結構體）

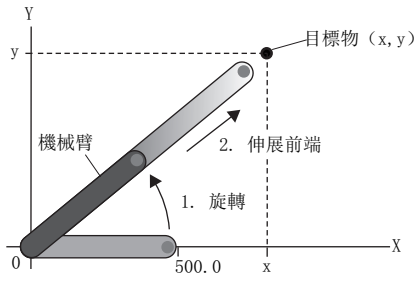
以下所示為在計算X、Y坐標上的位置或設備的旋轉量的程式中，進行了指數函數、三角函數的算術運算及使用了結構體的資料處理的示例。

本程式示例中創建以下程式部件。

資料名	資料類型	內容	參照
GetAngle	函數	針對目標的X、Y坐標，計算旋轉角度。	67頁 根據X、Y坐標計算旋轉角度(FUN)：GetAngle
GetDistance	函數	根據X、Y坐標，計算2點間的距離。	68頁 計算X、Y坐標的2點間距離 （FUN）：GetDistance
GetXY	函數	根據半徑和角度，計算X、Y坐標。	69頁 根據半徑和角度計算X、Y坐標 （FUN）：GetXY
PulseNumberCalculation	FB	根據移動量計算發向電機的陳述式脈衝數。	70頁 電機的陳述式脈衝數的計算 （FB）：PulseNumberCalculation
PositionControl	程式塊	根據想要移動的目標X、Y坐標，計算機械臂的移動角度和長度。	71頁 X、Y坐標的位置控制程式：PositionControl

功能的概要

根據以下步驟計算出機械臂的旋轉角度和機械臂的長度調整用電機的陳述式脈衝數，以便將機械臂的前端移動到目標的X、Y坐標。



1. 將機械臂轉至X、Y坐標所指定的目標物。
2. 通過步進電機伸展機械臂前端直至目標物。

使用的全局標簽

以下所示為本程式示例中使用的全局標簽及結構體定義。

在GX Works3中作如下設置。

■全局標簽

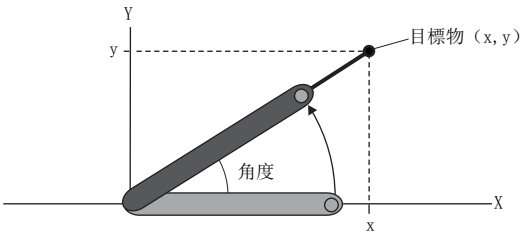
標簽名	資料類型	類	分配/初始值/常數	注釋
G_stTarget	stPosition	VAR_GLOBAL	—	目標物 （X、Y坐標）
G_stArm	stPosition	VAR_GLOBAL	—	機械臂前端 （X、Y坐標）
G_eAngle	單精度實數	VAR_GLOBAL	—	旋轉角度 （度）
G_ePulses	單精度實數	VAR_GLOBAL	—	電機的陳述式脈衝數
G_bOneScanOnly	位	VAR_GLOBAL	分配： SM402	RUN後僅1次掃描ON

■結構體

結構體名	標簽名	資料類型	初始值	注釋
stPosition	eXcoordinate	單精度實數	0.0	X坐標
	eYcoordinate	單精度實數	0.0	Y坐標

9.1 根據X、Y坐標計算旋轉角度(FUN)：GetAngle

計算到目標X、Y坐標為止的旋轉角度 （0～180度）。



角度（弧度）

$$\theta = \mathrm{ATAN}\left(\frac{y}{x}\right)$$

角度（度）

x = 0 時: = 90

x > 0 時: $= \theta \times \frac{180}{\pi}$

x < 0 時: $= \theta \times \frac{180}{\pi} + 180$

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	返回值的類型	EN/ENO	標題
函數	GetAngle	ST	單精度實數	是	根據X、Y坐標計算旋轉角度

程式範例

```
(* 根據X、Y坐標計算出弧度，並將弧度轉換成角度。 *)
(* X坐標為0時，為90度 （不進行計算而結束處理）*)
IF i_eXcoordinate = 0.0 THEN
    GetAngle := 90.0;
    ENO := TRUE;
    RETURN; (* 結束處理 *)
END_IF;
(* 根據X、Y坐標計算出弧度。 *)
eAngleRad := ATAN(i_eYcoordinate / i_eXcoordinate); (* 角度 （弧度） θ = ATAN(Y坐標/X坐標) *)
(* 包含ATAN陳述式所無法處理的資料時，以錯誤結束。 *)
IF SD0 = H3402 THEN (* 錯誤代碼： 3402H(運算異常) *)
    ENO := FALSE;
    RETURN; (* 結束處理 *)
    ELSE
    ENO := TRUE;
END_IF;
(* 將弧度轉換為角度。 *)
GetAngle := eAngleRad * 180.0 / c_ePi; (* 角度 （度） = 角度 （弧度） θ ×180/π *)
(* X坐標為負值時+180度。 *)
IF i_eXcoordinate < 0.0 THEN
    GetAngle := GetAngle + 180.0;
END_IF;
```

要點

單精度實數弧度→角度轉換可使用DEG陳述式。
SD0是錯誤檢查用的軟元件。存儲最新錯誤代碼。

变数

在GX Works3中作如下設置，定義標簽。返回值的類型可在函數的屬性中設置。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_eXcoordinate	單精度實數	VAR_INPUT	—	X坐標
i_eYcoordinate	單精度實數	VAR_INPUT	—	Y坐標
eAngleRad	單精度實數	VAR	—	角度 （弧度）
c_ePi	單精度實數	VAR_CONSTANT	常數： 3.14159	圓周率

■返回值的類型

識別符	資料類型	內容
GetAngle	單精度實數	角度 （度）：0～180度

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
ATAN	通用函數	TAN^{-1}運算 輸出輸入值的TAN^{-1}（反正切）值。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)

9.2 計算X、Y坐標的2點間距離 （FUN）：GetDistance

根據2點的X、Y坐標，計算2點間的距離。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	返回值的類型	EN/ENO	標題
函數	GetDistance	ST	單精度實數	否	2點間的距離計算

程式範例

```
(* 根據X、Y坐標計算2點間的距離。 *)
GetDistance := SQRT((i_stPosition0.eXcoordinate - i_stPosition1.eXcoordinate) ** 2.0
                  + (i_stPosition0.eYcoordinate - i_stPosition1.eYcoordinate) ** 2.0);
```

要點

函數/FB可以指定結構體型的參數。

可以對運算表達式的操作數 （運算物件的值）、賦值語句的左邊和右邊指定結構體成員。

“**”是冪乘的運算符。

变数

在GX Works3中作如下設置，定義標簽。返回值的類型可在函數的屬性中設置。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_stPosition0	stPosition	VAR_INPUT	—	位置0 （X、Y坐標）
i_stPosition1	stPosition	VAR_INPUT	—	位置1 （X、Y坐標）

■返回值的類型

識別符	資料類型	內容
GetDistance	單精度實數	2點間的距離

■結構體

結構體名	標簽名	資料類型	初始值	注釋
stPosition	eXcoordinate	單精度實數	0.0	X坐標
	eYcoordinate	單精度實數	0.0	Y坐標

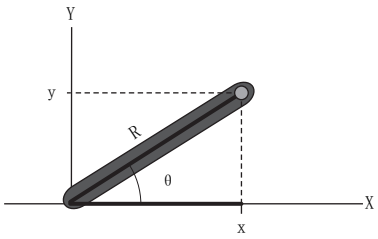
程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
SQRT	通用函數	平方根 輸出輸入值的平方根。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)

9.3 根據半徑和角度計算X、Y坐標 （FUN）：GetXY

根據指定的半徑和角度，計算X、Y坐標。



角度（弧度）	X、Y坐標
θ = 角度（度） × π/180	x = 半徑 R × COS（θ） y = 半徑 R × SIN（θ）

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	返回值的類型	EN/ENO	標題
函數	GetXY	ST	stPosition	否	X、Y坐標的計算

程式範例

```
(* 从度轉換成弧度單位。 *)
eAngleRad := i_eAngle * c_ePi / 180.0; (* 角度 （弧度） θ = 角度 （度） × π /180 *)
(* 計算X、Y坐標。 *)
GetXY.eXcoordinate := i_eRadius * COS(eAngleRad); (* X坐標 = 半徑 × COS( θ ) *)
GetXY.eYcoordinate := i_eRadius * SIN(eAngleRad); (* Y坐標 = 半徑 × SIN( θ ) *)
```

要點

函數無法指定結構體型的返回值。
單精度實數角度→弧度轉換可使用RAD陳述式。

变数

在GX Works3中作如下設置，定義標簽。返回值的類型可在函數的屬性中設置。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_eRadius	單精度實數	VAR_INPUT	—	半徑 （mm）
i_eAngle	單精度實數	VAR_INPUT	—	角度 （度）：0～180度
eAngleRad	單精度實數	VAR	—	角度 （弧度）
c_ePi	單精度實數	VAR_CONSTANT	常數：3.14159	圓周率

■返回值的類型

識別符	資料類型	內容
GetXY	stPosition	X、Y坐標

■結構體

結構體名	標簽名	資料類型	初始值	注釋
stPosition	eXcoordinate	單精度實數	0.0	X坐標
	eYcoordinate	單精度實數	0.0	Y坐標

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
COS	通用函數	COS運算 輸出輸入值的COS （餘弦）值。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)
SIN	通用函數	SIN運算 輸出輸入值的SIN （正弦）值。	

9.4 電機的陳述式脈衝數的計算 （FB）：PulseNumberCalculation

根據移動量計算發向電機的陳述式脈衝數。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	EN/ENO	標題
FB	PulseNumberCalculation	ST	否	陳述式脈衝數的計算

程式範例

```
(* 計算陳述式脈衝數。 *)
IF elround <> 0.0 THEN
    (* 陳述式脈衝數 = 電機分辨率 * ( 目標移動量 / 電機每轉的移動量 ) *)
    o_ePulses := eResolution * ( i_eDistance / elround );
END_IF;
```

变数

在GX Works3中作如下設置，定義標簽。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_eDistance	單精度實數	VAR_INPUT	—	目標移動量 （mm）
o_ePulses	單精度實數	VAR_OUTPUT	—	電機的陳述式脈衝數 （脈衝）
elround	單精度實數	VAR	—	電機每轉的移動量 （mm/rev）
eResolution	單精度實數	VAR	—	電機分辨率 （脈衝/rev）

要點

elround及eResolution在使用側程式的局部標簽設置中設置初始值。

☞ 72頁 局部標簽

程式部件

不使用。

9.5 X、Y坐標的位置控制程式： PositionControl

根據想要移動的目標X、Y坐標，計算移動的角度和長度，以控制機械臂。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	PositionControl	ST	X、Y坐標的位置控制程式

程式範例

```
(* 計算出機械臂的旋轉角度和機械臂的長度調整用電機的陳述式脈衝數，以便將機械臂的前端移動到目標的X、Y坐標。 *)
(* RUN後僅在第1次掃描時為機械臂的坐標設置初始值。 *)
IF G_bOneScanOnly THEN
   G_stArm.eXcoordinate := 500.0;
   G_stArm.eYcoordinate := 0.0;
END_IF;
(* 目標物的位置由其他程式中的全局变数G_stTarget設置。[本程式示例中未記載]*)
(* 計算機械臂對目標X、Y坐標的移動角度。 *)
eTargetAngle := GetAngle( TRUE, bResult1, G_stTarget.eXcoordinate, G_stTarget.eYcoordinate );
eArmAngle := GetAngle( bResult1, bResult2, G_stArm.eXcoordinate, G_stArm.eYcoordinate );
IF bResult2 THEN
   G_eAngle := eTargetAngle - eArmAngle;
   ELSE
   RETURN; (* 錯誤結束 *)
END_IF;
(* 根據機械臂的前端X、Y坐標，計算當前機械臂的長度 （與原點之間的距離）。 *)
eDistance := GetDistance( G_stArm, stOrigin ); (* 輸入变数為結構體型函數調用 *)
(* 計算旋轉後的機械臂前端的X、Y坐標。 *)
eDistance := GetDistance( G_stArm, stOrigin ); (* 返回值為結構體型函數調用 *)
(* 根據目標和機械臂前端的X、Y坐標，計算2點間的距離。 *)
eDistance := GetDistance( G_stTarget, G_stArm );
(* 根據移動量計算發向機械臂長度調整用電機的陳述式脈衝數。 *)
PulseNumberCalculation_1(i_eDistance := eDistance, o_ePulses => G_ePulses ); (* FB調用 *)
(* 指定角度，旋轉機械臂。[本程式示例中未記載] *)
(* 指定電機的陳述式脈衝數，伸展機械臂前端。[本程式示例中未記載] *)
(* 更新機械臂坐標。 *)
G_stArm := G_stTarget;
```

要點

可選擇EN/ENO的有無，創建函數或FB。

在EN中指定了FALSE時，不執行處理。ENO變為FALSE。

FB實例中，輸出变数用“=>”指定。

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	注釋
G_bOneScanOnly	位	VAR_GLOBAL	分配： SM402	RUN後僅1次掃描ON
G_stTarget	stPosition	VAR_GLOBAL	—	目標物 （X、Y坐標）
G_stArm	stPosition	VAR_GLOBAL	—	機械臂前端 （X、Y坐標）
G_eAngle	單精度實數	VAR_GLOBAL	—	旋轉角度 （度）
G_ePulses	單精度實數	VAR_GLOBAL	—	電機的陳述式脈衝數 （脈衝）

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
eTargetAngle	單精度實數	VAR	—	至目標物的角度 （度）：0～180度
eArmAngle	單精度實數	VAR	—	機械臂的角度 （度）：0～180度
bResult1	位	VAR	—	運算結果1
bResult2	位	VAR	—	運算結果2
stOrigin	stPosition	VAR	—	原點 （X、Y坐標 = 0, 0）
eDistance	單精度實數	VAR	—	距離
PulseNumberCalculation_1	PulseNumberCalculation	VAR	初始值 • e1round: 0.2 • eResolution: 3000.0	陳述式脈衝數的計算處理

■結構體

結構體名	標簽名	資料類型	初始值	注釋
stPosition	eXcoordinate	單精度實數	0.0	X坐標
	eYcoordinate	單精度實數	0.0	Y坐標

要點

在GX Works3中，可對每個FB實例的內部变数設置初始值。
應通過對結構體定義設置初始值、或通過程式初始化的方式，初始化結構體型变数。全局標簽時，可對要分配的軟元件設置初始值。

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
GetAngle	函數	針對目標的X、Y坐標，計算角度。	67頁 根據X、Y坐標計算旋轉角度(FUN): GetAngle
GetXY	函數	根據半徑和角度，計算X、Y坐標。	69頁 根據半徑和角度計算X、Y坐標 （FUN）: GetXY
GetDistance	函數	根據X、Y坐標，計算2點間的距離。	68頁 計算X、Y坐標的2點間距離 （FUN）: GetDistance
PulseNumberCalculation	FB	根據移動量計算發向電機的陳述式脈衝數。	70頁 電機的陳述式脈衝數的計算 （FB）: PulseNumberCalculation

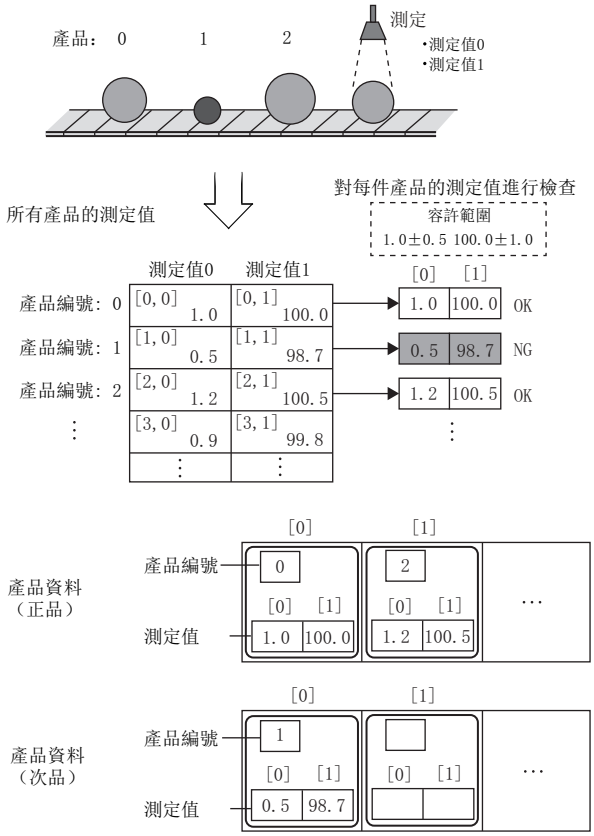
10 次品分類 （數組和循環處理）

以下所示為在要整理多個產品的資料的程式中，使用數組循環處理的示例。
本程式示例中創建以下程式部件。

資料名	資料類型	內容	參照
ProductCheck	FB	根據產品資料，判斷正品、次品。	74頁 產品檢查 （FB）：ProductCheck
Assortment	FB	將產品資料按正品、次品分類。	75頁 產品資料的分類 （FB）：Assortment
DataManagement	程式塊	整理產品資料。	76頁 產品資料管理程式： DataManagement

功能的概要

對測定了多個測定值 （大小、重量等）的產品資料進行檢查，劃分正品、次品。



1. 測定產品。

在本程式示例中，將對8件產品的2種測定值進行測定，並對測得的資料進行處理。請在所有產品的測定值（G_eValueArray）中存儲任意的值。

2. 對每件產品的測定值進行檢查。

- 正品： 所有測定值均在容許範圍內的產品
- 次品： 有測定值超出容許範圍的產品

3. 將資料按正品、次品分開。

4. 完成所有產品的檢查後，獲取正品和次品的資料。

使用的全局標簽

以下所示為本程式示例中使用的全局標簽及結構體定義。
在GX Works3中作如下設置。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	注釋
G_eValueArray	單精度實數 （0..7,0..1）	VAR_GLOBAL	—	所有產品的測定值 （按產品編號順序存儲）
GC_wValueNumber	字[有符號]	VAR_GLOBAL_CONSTANT	常數： 2	每件產品的測定值數
GC_wTotalProduct	字[有符號]	VAR_GLOBAL_CONSTANT	常數： 8	產品總數
G_stProductArray	stProduct(0..7)	VAR_GLOBAL	—	產品資料 （正品）
G_stDefectiveArray	stProduct(0..7)	VAR_GLOBAL	—	產品資料 （次品）

■結構體

結構體名	標簽名	資料類型	初始值	注釋
stProduct	wProductNumber	字[有符號]	—	產品編號
	eValueArray	單精度實數 （0..1）	—	測定值

10.1 產品檢查 （FB）：ProductCheck

判斷值是否在容許範圍內。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	EN/ENO	標題
FB	ProductCheck	ST	否	產品檢查

程式範例

```
(* 檢查數組各元素的值是否在指定的容許範圍內。 *)
FOR wIndex := 0 TO (i_wValueNumber - 1) BY 1 DO
    (* 根據基准值和容許差，求出上下限值。 *)
    eMaxValue := i_eAcceptableArray[wIndex, c_wBasicSize] + i_eAcceptableArray[wIndex, c_wTolerance];
    eMinValue := i_eAcceptableArray[wIndex, c_wBasicSize] - i_eAcceptableArray[wIndex, c_wTolerance];
    (* 檢查上下限值。 *)
    IF (eMaxValue >= i_eValueArray[wIndex]) AND (eMinValue <= i_eValueArray[wIndex]) THEN
        o_bResult := TRUE;
        ELSE
        o_bResult := FALSE;
        EXIT; (* 如有值超出範圍，則中斷結束 *)
    END_IF;
END_FOR;
```

要點

將數組型資料和循環語句結合，可以對數組的多個元素進行同一資料處理。

數組的各元素可以作為定義了資料類型的变数由運算表達式進行指定。

選擇語句 （IF語句、CASE語句）及循環語句 （FOR語句、WHILE語句、REPEAT語句）可以分層。

变数

在GX Works3中作如下設置，定義標簽。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_eValueArray	單精度實數 （0..1）	VAR_INPUT	—	判斷值
i_eAcceptableArray	單精度實數 （0..1,0..1）	VAR_INPUT	—	容許範圍
i_wValueNumber	字[有符號]	VAR_INPUT	—	每件產品的測定值數
o_bResult	位	VAR_OUTPUT	—	檢查結果
c_wBasicSize	字[有符號]	VAR_CONSTANT	常數：0	存儲基准值的元素編號
c_wTolerance	字[有符號]	VAR_CONSTANT	常數：1	存儲容許差的元素編號
eMaxValue	單精度實數	VAR	—	判斷的上限值
eMinValue	單精度實數	VAR	—	判斷的下限值
wIndex	字[有符號]	VAR	—	元素編號

要點

FB內可使用全局資料。

可將數組型資料指定給輸入变数。

程式部件

不使用。

10.2 產品資料的分類 （FB）：Assortment

將產品資料按正品、次品分類存儲。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	EN/ENO	標題
FB	Assortment	ST	否	產品資料的分類

程式範例

```
(* 根據檢查結果劃分正品、次品資料。*)
IF i_bCheck THEN
    (* 正品時 *)
    (* 在產品資料 （正品）中存儲測定值 *)
    o_stProductArray[wTotal] := i_stProduct;
    (* 正品數+1 *)
    wTotal := wTotal + 1;
    ELSE
    (* 次品時 *)
    (* 在產品資料 （次品）中存儲測定值 *)
    o_stDefectiveArray[wDefectiveTotal] := i_stProduct;
    (* 次品數+1 *)
    wDefectiveTotal := wDefectiveTotal + 1;
END_IF;
(* 更新成品率 *)
o_eYieldRatio := INT_TO_REAL(wTotal) / INT_TO_REAL(wTotal + wDefectiveTotal);
```

要點

ST程式中也可以使用成員中包含了數組的結構體或結構體型的數組。

变数

在GX Works3中作如下設置，定義標簽。

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
i_bCheck	位	VAR_INPUT	—	檢查結果
i_stProduct	stProduct	VAR_INPUT	—	檢查完了產品資料
o_stProductArray	stProduct(0..7)	VAR_OUTPUT	—	產品資料 （正品）
o_stDefectiveArray	stProduct(0..7)	VAR_OUTPUT	—	產品資料 （次品）
o_eYieldRatio	單精度實數	VAR_OUTPUT	—	成品率
wTotal	字[有符號]	VAR	—	正品數
wDefectiveTotal	字[有符號]	VAR	—	次品數

■結構體

結構體名	標簽名	資料類型	初始值	注釋
stProduct	wProductNumber	字[有符號]	—	產品編號
	eValueArray	單精度實數 （0..1）	—	測定值

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
INT_TO_REAL	通用函數	INT型→REAL型轉換 將INT型資料轉換成REAL型資料。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)

10.3 產品資料管理程式：DataManagement

將產品按正品、次品分開，分別為每件產品存儲測定值。
在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	DataManagement	ST	產品資料管理程式

程式範例

```
(* 檢查產品資料，劃分正品和次品。 *)
(* 設置檢查的容許範圍。 *)
Check_1.i_eAcceptableArray[0,0] := 1.0;(* 測定值0的基准值 *)
Check_1.i_eAcceptableArray[0,1] := 0.5;(* 測定值0的容許差 *)
Check_1.i_eAcceptableArray[1,0] := 100.0; (* 測定值1的基准值 *)
Check_1.i_eAcceptableArray[1,1] := 1.0;(* 測定值1的容許差 *)
Check_1.i_wValueNumber := GC_wValueNumber; (* 每件產品的測定值數 *)
(* 通過循環處理檢查所有產品的測定值。 *)
REPEAT
    (* 一次處理3件資料。 *)
    wDataEnd := wProductNumber + 3;
    (* 剩餘資料不足3件時，處理到最後。*)
    IF wDataEnd > GC_wTotalProduct THEN
        wDataEnd := GC_wTotalProduct;
    END_IF;
    (* 重覆是否為3件資料的判斷、分類處理。 *)
    WHILE wProductNumber < wDataEnd DO
        (* 獲取處理物件的測定值。 *)
        FOR wIndex := 0 TO (GC_wValueNumber - 1) BY 1 DO
            eValueArray[wIndex] := G_eValueArray[wProductNumber, wIndex];
        END_FOR;
        (* 根據測定值判斷正品、次品。*)
        Check_1(i_eValueArray := eValueArray, o_bResult => bResult);
        (* 根據檢查結果bResult劃分正品、次品資料。*)
        stProductData.wProductNumber := wProductNumber;
        stProductData.eValueArray := eValueArray;
        Assortment_1(i_bCheck := bResult, i_stProduct := stProductData);
        (* 進行到下一個產品編號。 *)
        wProductNumber := wProductNumber + 1;
    END_WHILE;
    (* 以下情況時，結束處理。 *)
    UNTIL ((Assortment_1.o_eYieldRatio < c_eLimit) (* 成品率低于判斷標准 *)
    OR(wProductNumber >= GC_wTotalProduct)) (* 或是產品編號超過產品總數 *)
END_REPEAT;
(* 獲取按正品、次品劃分的產品資料。 *)
G_stProductArray := Assortment_1.o_stProductArray;
G_stDefectiveArray := Assortment_1.o_stDefectiveArray;
```

要點

無法為數組各個元素設置不同的初始值。要設置不同值時，應通過程式進行設置。
FB時可在調用語句的前後指定參數。
選擇語句（IF語句、CASE語句）及循環語句（FOR語句、WHILE語句、REPEAT語句）可以分層。

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	注釋
G_eValueArray	單精度實數 （0..7,0..1）	VAR_GLOBAL	—	所有產品的測定值 （按產品編號順序存儲）
GC_wValueNumber	字[有符號]	VAR_GLOBAL_CONSTANT	常數： 2	每件產品的測定值數
GC_wTotalProduct	字[有符號]	VAR_GLOBAL_CONSTANT	常數： 8	產品總數
G_stProductArray	stProduct(0..7)	VAR_GLOBAL	—	產品資料 （正品）
G_stDefectiveArray	stProduct(0..7)	VAR_GLOBAL	—	產品資料 （次品）

■局部標簽

標簽名	資料類型	類	初始值/常數	注釋
wProductNumber	字[有符號]	VAR	—	產品編號 （內部循環處理用）
wDataEnd	字[有符號]	VAR	—	資料終端 （內部循環處理用）
wIndex	字[有符號]	VAR	—	元素編號 （內部循環處理用）
eValueArray	單精度實數 （0..1）	VAR	—	測定值
bResult	位	VAR	—	檢查結果 （內部循環處理用）
stProductData	stProduct	VAR	—	檢查完了產品資料
c_eLimit	單精度實數	VAR_CONSTANT	常數： 0.8	成品容許率
Check_1	ProductCheck	VAR	—	產品檢查處理
Assortment_1	Assortment	VAR	—	產品資料的分類處理

■結構體

結構體名	標簽名	資料類型	初始值	注釋
stProduct	wProductNumber	字[有符號]	—	產品編號
	eValueArray	單精度實數 （0..1）	—	測定值

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
ProductCheck	FB	根據產品資料，判斷正品、次品。	74頁 產品檢查 （FB）：ProductCheck
Assortment	FB	將產品資料按正品、次品分類。	75頁 產品資料的分類 （FB）：Assortment

11 運轉時間計測 （時間和字元串）

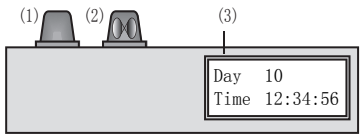
以下所示為在根據運轉時間點亮指示燈以顯示運轉時間的設備的程式中，進行定時器及時間資料處理的示例。
本程式示例中創建以下所示的程式部件。

資料名	資料類型	內容	參照
OperatingTime	程式塊	以秒為單位對設備的運轉時間進行計數。	79頁 運轉時間管理程式： OperatingTime
FlickerTimer	FB	使輸出信號交互閃爍。	80頁 閃爍定時器 （FB）：FlickerTimer
LampOnOff	程式塊	根據設備的狀態，點亮/熄滅指示燈。	81頁 指示燈的點亮/熄滅程式： LampOnOff
SecondsToTimeArray	程式塊	根據以秒為單位的時間，求出時、分、秒。	82頁 从秒至時、分、秒的轉換程式：SecondsToTimeArray
TimeToString	程式塊	將運轉時間 （時間型）轉換為顯示用字元串。	83頁 从時間型至字元串的轉換程式：TimeToString

功能的概要

執行以下處理。

1. 以秒為單位對設備的運轉時間進行計數。
2. 根據動作狀態和運轉時間，判斷設備狀態，點亮/熄滅指示燈。
3. 將運轉時間轉換為以時、分、秒為單位。
4. 將運轉時間轉換為畫面顯示用的字元串。

設備示意圖	編號	名稱	內容	
	(1)	運轉中指示燈	設備運轉中點亮。 熄滅： 停止中，點亮： 運轉中	運轉時間超過3周以上時，交互閃爍。
	(2)	警告指示燈	運轉時間超過1周以上時，點亮。 熄滅： 正常，點亮： 警告	
	(3)	顯示畫面	顯示運轉時間 （日、時、分、秒）。	

使用的全局標簽

以下所示為本程式示例中使用的全局標簽及結構體定義。
在GX Works3中作如下設置。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	內容
G_bOperatingStatus	位	VAR_GLOBAL	—	動作狀態 （TRUE：運轉中，FALSE：停止中）
G_tmTime	時間	VAR_GLOBAL	—	總運轉時間 （～T#24d20h31m23s647ms）
G_bOperationLamp	位	VAR_GLOBAL	—	運轉中指示燈 （TRUE：點亮，FALSE：熄滅）
G_bWarningLamp	位	VAR_GLOBAL	—	警告指示燈 （TRUE：點亮，FALSE：熄滅）
G_dSeconds	雙字[有符號]	VAR_GLOBAL	—	運轉時間 （0～86399秒）
G_wTimeArray	字[有符號](0..2)	VAR_GLOBAL	—	運轉時間 （[0]：時，[1]：分，[2]：秒）
G_sDisplayedCharacters	字元串(32)	VAR_GLOBAL	—	運轉時間顯示畫面的顯示字元
G_bOneScanOnly	位	VAR_GLOBAL	分配： SM402	RUN後僅1次掃描ON

■結構體

不使用。

11.1 運轉時間管理程式：OperatingTime

以秒為單位對設備的運轉時間進行計數。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	OperatingTime	ST	運轉時間管理程式

程式範例

```
(* 按秒對運轉時間進行計數。 *)
bResult := OUT_T(G_bOperatingStatus, td1sTimer, 10); (* 定時器的設定值1000ms *)
IF td1sTimer.S THEN (* 1秒後 *)
    (* 運轉時間 （TIMER型）計時 *)
    G_tmTime := G_tmTime + T#1000ms;
    IF G_tmTime < T#0ms THEN (* 發生溢出時 *)
        G_tmTime := T#0ms; (* 清零 *)
    END_IF;
    (* 運轉時間 （以秒為單位）計時 *)
    G_dSeconds := G_dSeconds +1;
    IF G_dSeconds >= 86400 THEN
        G_dSeconds := 0; (* 第24小時清零 *)
    END_IF;
    (* 定時器的復位 （當前值：0，觸點：OFF）*)
    RST(TRUE, td1sTimer.N);
    RST(TRUE, td1sTimer.S);
END_IF;
```

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	內容
G_bOperatingStatus	位	VAR_GLOBAL	—	運轉狀態 （TRUE：運轉中，FALSE：停止中）
G_tmTime	時間	VAR_GLOBAL	—	總運轉時間 （～T#24d20h31m23s647ms）
G_dSeconds	雙字[有符號]	VAR_GLOBAL	—	運轉時間 （0～86399秒）

■局部標簽

標簽名	資料類型	類	初始值/常數	內容
bResult	位	VAR	—	定時器執行的ENO
td1sTimer	定時器	VAR	—	1秒計測用定時器

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
OUT_T	陳述式	低速定時器陳述式 到OUT陳述式為止的運算結果為ON時，線圈置ON，執行定時器的計測。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)

11.2 閃爍定時器 (FB)：FlickerTimer

輸入信號為ON時，使輸出信號0、1交互閃爍。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	EN/ENO	標題
FB	FlickerTimer	ST	是	閃爍定時器

程式範例

```
(* 使輸出信號0、1交互閃爍。(觸發器梯形圖) *)
(* 求低速定時器的設定值。 *)
wInterval := TIME_TO_INT(i_tmInterval) / 100;
(* 定時器1為OFF時，定時器0在設置時間ms後置ON *)
bResult := OUT_T(NOT tdTimer1.S, tdTimer0, wInterval);
(* 定時器0从OFF變為ON時，定時器1在設置時間ms後置ON *)
bResult := OUT_T(tdTimer0.S, tdTimer1, wInterval);
(* 定時器0為ON時，輸出信號0置ON *)
o_bOutputSignal0 := tdTimer0.S;
(* 定時器0為OFF時，輸出信號1置ON *)
o_bOutputSignal1 := NOT tdTimer0.S;
```

要點

應將定時器的設定值設置成超過掃描時間+定時器的時限設置值。定時器的時限設置通過工程工具的參數進行設置。(默認： 100ms)

变数

在GX Works3中作如下設置，定義標簽。

■局部標簽

標簽名	資料類型	類	初始值/常數	內容
i_tmInterval	時間	VAR_INPUT	—	切換間隔
o_bOutputSignal0	位	VAR_OUTPUT	初始值： TRUE	輸出信號0
o_bOutputSignal1	位	VAR_OUTPUT	初始值： TRUE	輸出信號1
bResult	位	VAR	—	定時器執行的ENO
wInterval	字[有符號]	VAR	—	低速定時器設定值
tdTimer0	定時器	VAR	—	定時器0
tdTimer1	定時器	VAR	—	定時器1

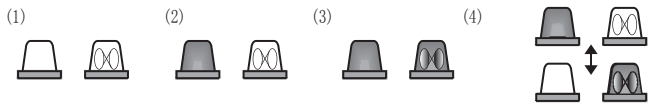
程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
OUT_T	陳述式	低速定時器陳述式 到OUT陳述式為止的運算結果為ON時，線圈置ON，執行定時器的計測。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)
TIME_TO_INT	通用函數	TIME型→INT型轉換 將TIME型資料轉換成INT型資料。	

11.3 指示燈的點亮/熄滅程式：LampOnOff

根據設備的狀態，點亮/熄滅指示燈。



編號	運轉中指示燈	警告指示燈	設備的狀態		備注
(1)	熄滅	熄滅	正常	停止中	運轉前
(2)	點亮	熄滅		運轉中	運轉時間0～7天
(3)	點亮	點亮	警告		運轉時間1周以上
(4)	交互閃爍		異常		運轉時間3周以上

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	LampOnOff	ST	指示燈的點亮/熄滅程式

程式範例

```
(* 根據運轉狀態，對運轉中指示燈進行ON/OFF。*)
G_bOperationLamp := G_bOperatingStatus;
(* 總運轉時間為1周 （7天）以上時，警告指示燈ON。 *)
G_bWarningLamp := G_tmTime >= T#7d;
(* 總運轉時間為3周 （21天）以上時， *)
(* 運轉中指示燈和警告指示燈交互閃爍。 *)
FlickerTimer_1(EN := G_tmTime >= T#21d, i_tmInterval := T#1000ms);
IF FlickerTimer_1.ENO THEN
    G_bOperationLamp := FlickerTimer_1.o_bOutputSignal0;
    G_bWarningLamp := FlickerTimer_1.o_bOutputSignal1;
END_IF;
```

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	內容
G_bOperatingStatus	位	VAR_GLOBAL	—	動作狀態 （TRUE：運轉中，FALSE：停止中）
G_bOperationLamp	位	VAR_GLOBAL	—	運轉中指示燈 （TRUE：點亮，FALSE：熄滅）
G_bWarningLamp	位	VAR_GLOBAL	—	警告指示燈 （TRUE：點亮，FALSE：熄滅）
G_tmTime	時間	VAR_GLOBAL	—	總運轉時間 （～T#24d20h31m23s647ms）

■局部標簽

標簽名	資料類型	類	初始值/常數	內容
FlickerTimer_1	FlickerTimer	VAR	—	閃爍定時器

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
FlickerTimer	FB	使輸出信號交互閃爍。	80頁 閃爍定時器 （FB）：FlickerTimer

11.4 从秒至時、分、秒的轉換程式： SecondsToTimeArray

根據以秒為單位的時間，求出時、分、秒。時、分、秒的值存儲在數組型資料 （與時鐘用陳述式SEC2TIME的參數的資料格式相同）中。

在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	SecondsToTimeArray	ST	从秒至時、分、秒的轉換程式

程式範例

```
(* 根據以秒為單位的時間，求出時、分、秒。 *)
G_wTimeArray[0] := GET_INT_ADDR( G_dSeconds / 3600); (* 小時 *)
G_wTimeArray[1] := GET_INT_ADDR((G_dSeconds MOD 3600) / 60); (* 分 *)
G_wTimeArray[2] := GET_INT_ADDR((G_dSeconds MOD 3600) MOD 60); (* 秒 *)
```

要點

MOD是餘數運算的運算符。

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	內容
G_dSeconds	雙字[有符號]	VAR_GLOBAL	—	運轉時間 （0～86399秒）
G_wTimeArray	字[有符號](0..2)	VAR_GLOBAL	—	運轉時間 （[0]: 時，[1]: 分，[2]: 秒）

■局部標簽

不使用。

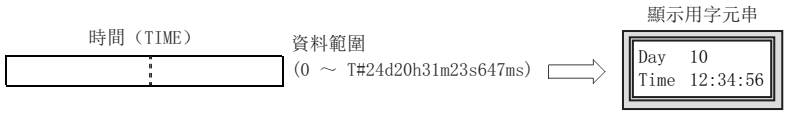
程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
GET_INT_ADDR	通用函數	無需類型轉換 輸入变数按INT型輸出。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)

11.5 从時間型至字元串的轉換程式： TimeToString

將運轉時間 （時間型）轉換為顯示用字元串。



在GX Works3中作如下設置，創建程式部件。

資料類型	資料名	程式語言	標題
程式塊	TimeToString	ST	从時間型至字元串的轉換程式

程式範例

```
(* 將運轉時間轉換為顯示用字元串。*)
(* 初始化內部变数。*)
IF G_bOneScanOnly THEN (* RUN後僅在第1次掃描時執行*)
(* 初始化時間的單位 （日、時、分、秒的ms數）。 *)
   dUnitArray[0] := TIME_TO_DINT(T#1d); (* 日 *)
   dUnitArray[1] := TIME_TO_DINT(T#1h); (* 小時 *)
   dUnitArray[2] := TIME_TO_DINT(T#1m); (* 分 *)
   dUnitArray[3] := TIME_TO_DINT(T#1s); (* 秒 *)
(* 指定要轉換為字元串的數值的數位。 *)
   wDigitArray[0] := 2; (* 按2位轉換為字元串 *)
   wDigitArray[1] := 0; (* 無小數點 *)
END_IF;
sTimeString := ''; (* 初始化字元串 *)
(* 將時間 （TIME）型資料轉換為ms數。 *)
dTime := TIME_TO_DINT(G_tmTime);
(* 按日、時、分、秒的順序處理。 *)
FOR wIndex := 0 TO 3 BY 1 DO
(* 將時間 （ms單位）轉換為日/時/分/秒。 *)
   wTime := DINT_TO_INT(dTime /dUnitArray[wIndex]);
   dTime := dTime MOD dUnitArray[wIndex];
(* 將日/時/分/秒的值轉換為2位的字元串。 *)
   STR(TRUE, wDigitArray, wTime, sItemString);
   (* 設置要添加的字元串。 *)
   CASE wIndex OF
      0 : sAdd := 'Day ';
      1 : sAdd := '$nTime ';
      2,3:sAdd := ':';
   END_CASE;
   (* 添加字元串。 *)
   sTimeString := CONCAT(sTimeString, sAdd, sItemString);
END_FOR;
(* 將創建的字元串設置為顯示字元串。 *)
G_sDisplayedCharacters := sTimeString;
```

要點

使用通用函數的TIME_TO_STRING時，以ms為單位的值會轉換成字元串。本程式中，先求出按日、時、分、秒分開的值，再轉換為字元串。

“$n”是換行代碼。

变数

在GX Works3中作如下設置，定義標簽。

■全局標簽

標簽名	資料類型	類	分配/初始值/常數	內容
G_tmTime	時間	VAR_GLOBAL	—	總運轉時間 （～T#24d20h31m23s647ms）
G_bOneScanOnly	位	VAR_GLOBAL	分配： SM402	RUN後僅1次掃描ON
G_sDisplayedCharacters	字元串(32)	VAR_GLOBAL	—	運轉時間顯示畫面的顯示字元

■局部標簽

標簽名	資料類型	類	初始值/常數	內容
dUnitArray	雙字[有符號](0..3)	VAR	—	單位 （日、時、分、秒的ms數）
wDigitArray	字[有符號](0..1)	VAR	—	數位指定 （[0]：所有位數，[1]：小數位數）
sTimeString	字元串(32)	VAR	—	創建的字元串 （日、時、分、秒）
dTime	雙字[有符號]	VAR	—	時間 （ms單位）
wIndex	字[有符號]	VAR	—	元素編號
wTime	字[有符號]	VAR	—	時間 （日/時/分/秒）
sItemString	字元串(32)	VAR	—	日/時/分/秒的2位數值字元串
sAdd	字元串(32)	VAR	—	添加的字元串

程式部件

關于使用的程式部件，請參照以下內容。

資料名	資料類型	內容	參照
TIME_TO_DINT	通用函數	TIME型→DINT型轉換 將TIME型資料轉換成DINT型資料。	MELSEC iQ-R 程式手冊(指令/通用FUN/通用FB篇)
DINT_TO_INT	通用函數	DINT型→INT型轉換 將DINT型資料轉換成INT型資料。	
STR	陳述式	在BIN16位資料的指定位置添加小數點並將其轉換為字元串。	
CONCAT	通用函數	字元串的合併 合併並輸出字元串。	

備忘錄

附錄

附1　ST語言規格

以下所示為ST語言中使用的構成單位的類型及內容。

語句

以下所示為語句的類型和表述方法。

類型			表述方法	內容
賦值語句			〈變數〉 :=〈表達式〉; 代入結果	將右邊的評價結果代入左邊的變數。
子程式控制語句		調用語句	〈識別符〉(參數1, 參數2, ???);	調用函數、FB等。
		RETURN語句	RETURN;	中途結束程式。

控制語法	選擇語句	IF	IF 〈條件式1〉 THEN
			〈執行語句1〉;
			ELSIF 〈條件式2〉 THEN
			〈執行語句2〉;
			ELSIF（---線內）可多個
			ELSE
			〈執行語句3〉;
			ELSIF, ELSE（---線內）可省略
			END_IF;
				根據BOOL值的條件，選擇執行語句。
				條件式1為真 （TRUE）時，執行執行語句1。
				條件式1為假 （FALSE）時，對“ELSIF”的條件進行判斷。
				條件式2為真 （TRUE）時，執行執行語句2。
				“IF”及“ELSIF”的所有條件式均為假 （FALSE）時，執行“ELSE”後的執行語句3。

		CASE	CASE 〈條件式〉 OF 〈整數值1〉 : 〈執行語句1〉; 〈整數值2〉..〈整數值3〉 : 〈執行語句2〉; 可多個 ELSE 〈執行語句3〉; ELSE（---線內）可省略 END_CASE;	根據整數值的條件，選擇執行語句。 條件式的結果與整數值1一致時，執行執行語句1。 要通過範圍指定要判斷的整數值時，使用“..”表述。條件式結果的值在整數值2～整數值3的範圍內時，執行執行語句2。 與所有的整數值或範圍都不一致時，執行“ELSE”後的執行語句3。
	循環語句	FOR	FOR 〈變數〉 :=〈初始值（表達式）〉 TO 〈最終值（表達式）〉 BY 〈增加值（表達式）〉 BY（---線內）可省略 DO 〈執行語句〉; END_FOR;	根據整數值的結束條件，多次執行執行語句。 執行執行語句直到設置了初始值的整數型變數變為最終值為止。每執行1次執行語句，即在變數上加上增加值。
		WHILE	WHILE 〈條件式〉 DO 〈執行語句〉; 循環直至變為假（FALSE） END_WHILE;	根據BOOL值的結束條件，多次執行執行語句。 條件式為真 （TRUE）時，執行執行語句。反覆執行直到條件式的結果變為假 （FALSE）為止。
		REPEAT	REPEAT 〈執行語句〉; 循環直至變為真（TRUE） UNTIL 〈條件式〉 END_REPEAT;	根據BOOL值的結束條件，多次執行執行語句。 執行執行語句後，對條件進行判斷。 反覆執行直到條件式的結果變為真 （TRUE）為止。
	循環語句的中斷		EXIT;	中斷循環語句。
空語句			;	不進行任何處理。

要點

函數和FB總計可以進行32次分層。

IF語句、CASE語句、FOR語句、WHILE語句、REPEAT語句總計可以進行128次分層。

運算符

以下所示為運算符的類型和表述方法。

類型		運算符	優先順序（从高到低的順序）	示例	
				一般算式表述	ST
符號的反轉		-	1	B = - A	eValueB := - eValueA;
邏輯運算	邏輯非	NOT		$B = \overline{A}$	bFlagB := NOT bFlagA;
冪乘		**	2	$B = C^A$	eValueB := eValueC ** eValueA;
四則運算	乘法運算	*	3	A × B = C	eValueC := eValueA * eValueB;
	除法運算	/		A ÷ B = C ... D	eValueC := eValueA / eValueB;
	餘數運算	MOD			eValueC := eValueA MOD eValueB;
	加法運算	+	4	A + B = C	eValueC := eValueA + eValueB;
	減法運算	-		A - B = C	eValueC := eValueA - eValueB;
比較運算	大于、小于	>、<	5	A > B	bFlag := eValueA > eValueB;
	大于等于、小于等于	>=、<=		A ≤ B	bFlag := eValueA <= eValueB;
	等于	=	6	A = B	bFlag := eValueA = eValueB;
	不等于	<>		A ≠ B	bFlag := eValueA <> eValueB;
邏輯運算	邏輯與	AND、&	7	A ∧ B	bFlag := eValueA AND eValueB;
	邏輯異或	XOR	8	A ∀ B	bFlag := eValueA XOR eValueB;
	邏輯或	OR	9	A ∨ B	bFlag := eValueA OR eValueB;

要點

1個表達式中最多可使用1024個運算符。

優先順序

將多個運算表達式表述在1個表達式中時，將从優先級高的運算符開始處理。

有多個優先級相同的運算符時，从左邊的運算符開始運算。

小括號()內的運算表達式將優先進行運算。

注釋

以下所示為注釋的類型和表述方法。

類型	符號	內容	示例
多行注釋	(* *)	从開始符號到結束符號為止的內容作為注釋處理。	(* 注釋 *)
	/* */		/* 注釋 */
單行注釋	//	从開始符號到行尾為止的內容作為注釋處理。	// 注釋

要點

IEC 61131-3中定義的ST的注釋符號僅有(**)，但在GX Works3中可以使用與C語言等相同的符號 (/**/、//)來表述注釋。

GX Works2中，僅可使用帶有(**)的注釋。

附

軟元件

可與梯形圖一樣指定軟元件。帶#號時可以指定本地軟元件。
可使用軟元件的指定（數位指定、位指定、間接指定）及變址修飾。

限制事項

ST程式中無法使用指針。

定時器、計數器的觸點/線圈/當前值

在ST語言中，可指明觸點/線圈/當前值以使用定時器、計數器等軟元件。
沒有指定時，會根據使用的陳述式自動判斷觸點/線圈/當前值。
觸點、線圈按位型處理，當前值按以下資料類型處理。

軟元件	表示方法				當前值的資料類型
	無指定	觸點	線圈	當前值	
定時器	T	TS	TC	TN	字[無符號]/位列[16位]
累積定時器	ST	STS	STC	STN	
計數器	C	CS	CC	CN	
長定時器	LT	LTS	LTC	LTN	雙字[無符號]/位列[32位]
長累積定時器	LST	LSTS	LSTC	LSTN	
長計數器	LC	LCS	LCC	LCN	

字軟元件的類型指定

字軟元件可指明資料類型進行使用。

資料類型	軟元件類型指定符	示例	說明
字[無符號]/位列[16位]	:U	D0:U	將D0的值按WORD型16位的值處理。
雙字[無符號]/位列[32位]	:UD	D0:UD	將D0、D1的值按DWORD型32位的值處理。
字[有符號]	（無）	D0	將D0的值按INT型16位的值處理。
雙字[有符號]	:D	D0:D	將D0、D1的值按DINT型32位的值處理。
單精度實數	:E	D0:E	將D0、D1的值按REAL型32位的值處理。
雙精度實數	:ED	D0:ED	將D0～D3的值按LREAL型64位的值處理。

對進行了數位指定或間接指定的軟元件，不可添加軟元件類型指定符。

■字軟元件的類型轉換

將沒有指定類型的字軟元件指定給了函數或FB的參數時，將按參數定義的資料類型處理。（通用函數、通用FB及陳述式時也一樣。）
指定給輸入參數時會自動進行類型轉換，因此根據對參數的指定方法結果會有所不同。

例

BIN32位資料傳送陳述式（DMOV）時（輸入參數、輸出參數為ANY32型的陳述式）

- D0 = 16#ABCD
- D1 = 16#1234
- 分配了D0的字[有符號]型標簽G_wLabel

ST	內容	傳送來的值
bResult := DMOV(TRUE, D0, D10);	將存儲在D0、D1中的32位資料傳送到D10、D11。	16#1234ABCD
bResult := DMOV(TRUE, D0:UD, D10:UD);		
bResult := DMOV(TRUE, D0:U, D10);	存儲在D0中的字[有符號]/位列[16位]型的整數值會被自動轉換為雙字[有符號]型，零擴展的值會被傳送到D10、D11。	16#0000ABCD
bResult := DMOV(TRUE, G_wLebel, D10);	存儲在D0中的字[有符號]型的整數值會被自動轉換為雙字[有符號]型的整數值，符號擴展的值會被傳送到D10、D11。	16#FFFFABCD
bResult := DMOV(TRUE, D0:U, D10:U);	輸出变数無法自動轉換，因此會出現D10:U轉換錯誤。	—

在運算表達式中使用未指定類型的字軟元件時，將會執行从字[有符號]型資料的自動轉換。
☞26頁 可自動轉換的資料類型

標簽

可與梯形圖一樣指定標簽。

可以使用標簽的位指定 （例： Lbl.3）、數位指定 （例： K4Lbl）。

限制事項

ST程式中無法使用指針型標簽。

常數

以下所示為常數的表述方法。

類型		表示方法	表示示例	帶“_”的表示示例*1
BOOL值		以TRUE或FALSE表述。	TRUE、FALSE	
		用1或0的值表述。可使用整數的各種表示方法。	2#0、8#1、0、H1	
整數	2進制數	2進制數前加上“2#”。	2#0010、2#01101010	2#111_1111_1111_1111
	8進制數	8進制數前加上“8#”。	8#2、8#152、8#377	8#7_7777
	10進制數	直接輸入10進制數。	2、106、-1	32_767
		10進制數前加上“K”。	K2、K106、K-1	—*2
	16進制數	16進制數前加上“16#”。	16#2、16#6A、16#FF	16#7F_FF
		16進制數前加上“H”。	H2、H6A、HFF	—*2
實數	小數表示	直接輸入實數。	1200.0、0.012、-0.1	3.14_159
		實數前加上“E”。	E1200、E0.012、E-0.1	—*2
	指數表示	尾數部和指數之間加上“E”。 “mEn”表示尾數部m乘以10的n次冪。	1.2E3、1.2E-2、-1.0E-1	2.99_792_458E8
		尾數部前面加上“E”，尾數部和指數之間加上“+”/“-”。 “Em+n”表示尾數部m乘以10的n次冪。*3	E1.2+3、E1.2-2、E-1.0-1	—*2
字元串	ASCII Shift JIS	字元串使用單引號（’）括起來。	’ABC’	
	Unicode	字元串使用雙引號（″）括起來。	″ABC″	
時間		前面加上“T#”或“TIME#”。	T#1h、T#1d2h3m4s5ms、TIME#1h	

*1 在整數、實數的表示中，使用下劃線（_）區隔數值，可使程式更易讀懂。在程式的處理中，通過下劃線（_）進行的數值區隔將被忽略。

*2 前面加了K、H、E時，無法使用下劃線（_）。

*3 在前面加有“E”的實數表示中，如果在值後面再加上“+”/“-”，則會作為指數處理。
要表述為算術運算時，應在數值和運算符之間插入空格或制表符。
（例：“E1.2+3”表示1200.0，“E1.2 +3”表示4.2。）

指定資料類型時

指定以下值時可以標明資料類型。不指定時，將自動判斷資料類型。

不區分資料類型名的大寫字母/小寫字母。不能與使用了K、H、E的常數的表示同時使用。

常數	可指定的資料類型		表示示例	帶“_”的表示示例
BOOL值	位	BOOL	BOOL#TRUE、BOOL#FALSE、BOOL#0、BOOL#1	
整數	字[無符號]/位列[16位]	UINT	UINT#2#01101010、UINT#8#152、UINT#106、UINT#16#6A	UINT#2#0110_1010
		WORD*1	WORD#2#01101010、WORD#8#152、WORD#106、WORD#16#6A	WORD#2#0110_1010
	雙字[無符號]/位列[32位]	UDINT	UDINT#2#11111111111111111、UDINT#8#77777、UDINT#32767、UDINT#16#7FFF	UDINT#16#7F_FF
		DWORD*1	DWORD#2#11111111111111111、DWORD#8#77777、DWORD#32767、DWORD#16#7FFF	DWORD#16#7F_FF
	字[有符號]	INT	INT#2#01101010、INT#8#152、INT#106、INT#16#6A	INT#2#0110_1010
	雙字[有符號]	DINT	DINT#2#11111111111111111、DINT#8#77777、DINT#32767、DINT#16#7FFF	DINT#16#7F_FF
實數	單精度實數	REAL	REAL#2.34、REAL#1.0E6	REAL#3.14_159
	雙精度實數	LREAL	LREAL#-2.34、LREAL#1.001E16	LREAL#1.00_1E16

*1 四則運算公式的操作數(運算物件值)及函數、FB的調用語句及函數調用表達式的ANY_NUM型的參數時，不能使用“WORD#”及“DWORD#”。應使用“UINT#”或“UDINT#”。

字元串型的常數中使用“$”時

在字元串型常數中要指定換行等時，使用“$”表示。

類型	表示
美元符號 （$）	$$、$24
單引號 （’）	$’、$27
雙引號 （″）	$″、$22
換行 （Line feed）	$L、$l、$0A
換行 （Newline）	$N、$n、$0D $0A
翻頁	$P、$p、$0C
回車符	$R、$r、$0D
制表符	$T、$t、$09
ASCII碼對應字元	$[ASCII碼(2數位的16進制數)]

時間長度的記述方法

時間型的常數按以下格式進行表述。

- T#23d23h59m59s999ms或TIME#23d23h59m59s999ms

時間單位	d(日)	h(時)	m(分)	s(秒)	ms(毫秒)
範圍	0～24	0～23	0～59	0～59	0～999

時間單位需按d、h、m、s、ms的順序連續。

可以在#的後面附加符號(-)，作為有符號的值進行記述。

（例： T#-24d20h31m23s648ms）

■時間單位的省略

前後的時間單位可以省略。

（例： T#1m2s）

中途的時間單位不能省略。

(例：不能記述為“T#1m2ms”。應記述為“T#1m0s2ms”。)

對於最後的時間單位，可以通過使用了小數點的表示(無符號實數的10進制數表示)進行記述。

(例： “T#1.234s”被處理為“T#1s234ms”。)

ms(毫秒)單位無法以小數點表示。小數點以下將被捨去。

(例： “T#1.234ms”被處理為“T#1ms”。)

■設置範圍

使用符號(-)，省略了前面的時間單位時，可記述的值的範圍如下所示。

時間單位	d(日)	h(時)	m(分)	s(秒)	ms(毫秒)
無省略	-24～24	0～23	0～59	0～59	0～999
省略d	—	-596～596	0～59	0～59	0～999
省略d、h	—	—	-35791～35791	0～59	0～999
省略d、h、m	—	—	—	-2147483～2147483	0～999
省略d、h、m、s	—	—	—	—	-2147483648～2147483647

時間型變數(☞29頁 時間型 （TIME） 變數)中可以在以下範圍內設置時間長度。

- T#-24d20h31m23s648ms～T#24d20h31m23s647ms
- T#-596h31m23s648ms～T#596h31m23s647ms
- T#-35791m23s648ms～T#35791m23s647ms
- T#-2147483s648ms～T#2147483s647ms
- T#-2147483648ms～T#2147483647ms

函數和FB

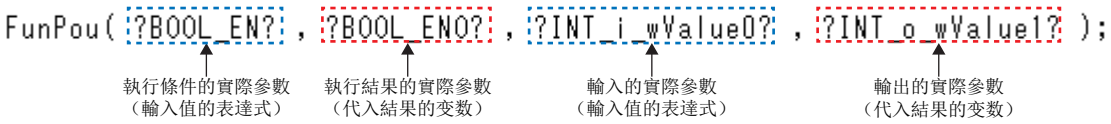
函數和FB是對程式中調用的子程式進行了定義的程式部件。
定義的函數可在程式塊、FB及其他函數中使用。
定義的FB可在程式塊及其他FB中通過創建實例來使用。

參數

本部分對調用語句中的參數的表述方法進行說明。

■實際參數的參數指定

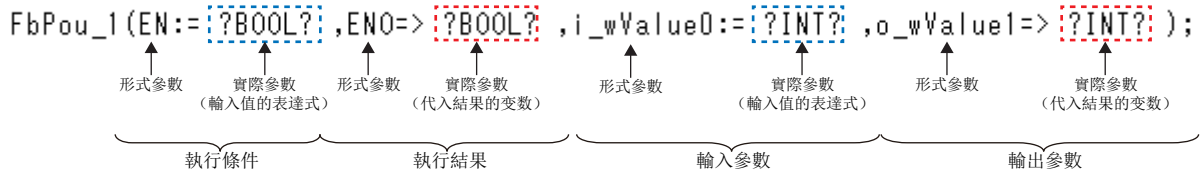
在調用語句的括號中，使用逗號（,）區隔列出參數。
函數的模板按此格式顯示。（☞47頁 輸入參數）



參數的順序是函數或FB定義的局部標簽設置中的定義順序。
如果創建時設置為使用EN/ENO，則第1參數指定為EN，第2參數指定為ENO。

■向形式參數中分配實際參數的參數指定

在調用語句的括號中，使用逗號（,）區隔列出形式參數和實際參數。
FB的模板按此格式顯示。（☞48頁 輸入參數）

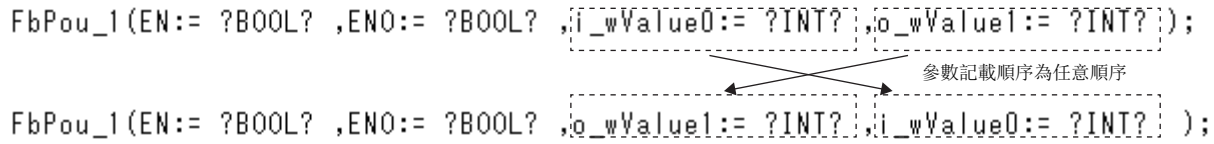


形式參數名是函數或FB定義的局部標簽設置中指定的變數名。
EN/ENO指定“EN”、“ENO”作為形式參數。

要點

向形式參數中分配實際參數時，按以下格式指定。
- 輸入參數、輸入輸出參數及EN時：“<形式參數名>:=<表達式>”
- 輸出參數及ENO時：“<形式參數名>=><變數>”

向形式參數中分配實際參數時，可任意更改參數的表述順序。

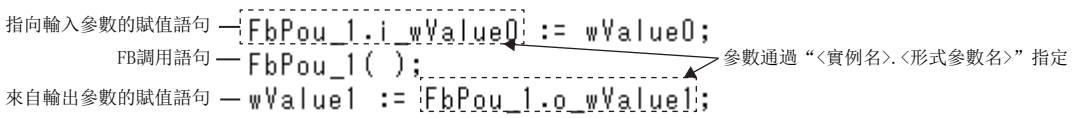


要點

向形式參數中分配實際參數並指定參數時，可省略參數的表述。

■調用語句前後的參數指定

FB可在調用語句的前後指定參數。表述時應將向輸入參數的賦值語句寫在調用語句之前，來自輸出參數的賦值語句寫在調用語句之後。



返回值

有返回值的函數會在執行結束後將值返回到調用源。

■返回值的資料類型

返回值的資料類型通過“New Data （新建資料）”畫面或“Properties （屬性）”畫面設置。

沒有設置資料類型時，將作為沒有返回值的函數處理。

沒有返回值的函數以調用語句表述。（函數調用語句）

■在返回值中設置值

在函數中，將作為返回值返回的值設置到將函數名作為識別符的變數中。

程式範例

將輸入參數i_wValue0～i_wValue2的和作為函數FunAdd的返回值返回。

ST （函數FunAdd的程式）

```
FunAdd := i_wValue0 + i_wValue1 + i_wValue2;
```

■在調用源使用返回值

有返回值的函數可作為表達式處理。（函數調用表達式）

在函數的調用源程式中，可將函數調用表達式作為返回值的資料類型的變數記述運算表達式或條件語句。

程式範例

在Average3中代入wValue0～wValue2的平均值。

將輸入參數wValue0～wValue2的和作為函數FunAdd的返回值返回。

ST （調用源的程式）

```
Average3 := FunAdd( wValue0, wValue1, wValue2) / 3;
```

EN/ENO的思路

可選擇EN/ENO的有無，創建函數及FB。

帶上EN （使能輸入）、ENO （使能輸出），可對執行處理進行控制。

- EN：設置執行條件。
- ENO：輸出執行結果。

EN/ENO為BOOL型。

帶EN的函數及FB，僅在EN的執行條件為TRUE時執行。

以下所示為與EN和ENO狀態所對應的輸出變數及返回值。

EN	ENO	輸出变数、返回值	備注
TRUE	TRUE	運算輸出值	正常結束
	FALSE	不定值	執行處理中，ENO中代入了FALSE時。 輸出的值視實際裝置而定。
FALSE	FALSE	• 函數：調用時的值 • FB：上次的結果	不執行處理，直接結束。 也不執行對輸入变数、輸出变数的賦值。

■在ENO中設置值

要在函數/FB內設置ENO的值時，應在变数“ENO”中代入BOOL值。

程式範例

BCD陳述式中出現了運算錯誤時，將中斷函數/FB的處理，並錯誤結束。

ST （函數/FB的程式）

```
ENO := BCD(EN, wValue0, DO);
IF ENO = FALSE THEN
    RETURN;
END_IF;
```

附2　ST中無法使用的陳述式

在ST中使用運算符或控制語句表述以下在梯形圖中使用的陳述式。

賦值可使用的陳述式

類型	陳述式符號	對應的類型轉換函數
單精度實數→有符號BIN16位資料	FLT2INT(P)	REAL_TO_INT(_E)
單精度實數→無符號BIN16位資料	FLT2UINT(P)	—(REAL_TO_DINT(_E), DINT_TO_WORD(_E))
單精度實數→有符號BIN32位資料	FLT2DINT(P)	REAL_TO_DINT(_E)
單精度實數→無符號BIN32位資料	FLT2UDINT(P)	—(REAL_TO_DINT(_E), DINT_TO_DWORD(_E))
雙精度實數→有符號BIN16位資料	DBL2INT(P)	LREAL_TO_INT(_E)
雙精度實數→無符號BIN16位資料	DBL2UINT(P)	—(LREAL_TO_DINT(_E), DINT_TO_WORD(_E))
雙精度實數→有符號BIN32位資料	DBL2DINT(P)	LREAL_TO_DINT(_E)
雙精度實數→無符號BIN32位資料	DBL2UDINT(P)	—(LREAL_TO_DINT(_E), DINT_TO_DWORD(_E))
有符號BIN16位資料→無符號BIN16位資料轉換	INT2UINT(P)	INT_TO_WORD(_E)
有符號BIN16位資料→有符號BIN32位資料轉換	INT2DINT(P)	INT_TO_DINT(_E)*1
有符號BIN16位資料→無符號BIN32位資料轉換	INT2UDINT(P)	INT_TO_DWORD(_E)
無符號BIN16位資料→有符號BIN16位資料轉換	UINT2INT(P)	WORD_TO_INT(_E)
無符號BIN16位資料→有符號BIN32位資料轉換	UINT2DINT(P)	WORD_TO_DINT(_E)*1
無符號BIN16位資料→無符號BIN32位資料轉換	UINT2UDINT(P)	WORD_TO_DWORD(_E)*1
有符號BIN32位資料→有符號BIN16位資料轉換	DINT2INT(P)	DINT_TO_INT(_E)
有符號BIN32位資料→無符號BIN16位資料轉換	DINT2UINT(P)	DINT_TO_WORD(_E)
有符號BIN32位資料→無符號BIN32位資料轉換	DINT2UDINT(P)	DINT_TO_DWORD(_E)
無符號BIN32位資料→有符號BIN16位資料轉換	UDINT2INT(P)	DWORD_TO_INT(_E)
無符號BIN32位資料→無符號BIN16位資料轉換	UDINT2UINT(P)	DWORD_TO_WORD(_E)
無符號BIN32位資料→有符號BIN32位資料轉換	UDINT2DINT(P)	DWORD_TO_DINT(_E)
字元串傳送	$MOV(P)	—*1
Unicode對應字元串傳送	$MOV(P)_WS	—*1
有符號BIN16位資料→單精度實數轉換	INT2FLT(P)	INT_TO_REAL(_E)*1
無符號BIN16位資料→單精度實數轉換	UINT2FLT(P)	—(WORD_TO_INT(_E), INT_TO_REAL(_E))*1
有符號BIN32位資料→單精度實數轉換	DINT2FLT(P)	DINT_TO_REAL(_E)
無符號BIN32位資料→單精度實數轉換	UDINT2FLT(P)	—(DWORD_TO_DINT(_E), DINT_TO_REAL(_E))
雙精度實數→單精度實數轉換	DBL2FLT(P)	LREAL_TO_REAL(_E)
有符號BIN16位資料→雙精度實數轉換	INT2DBL(P)	INT_TO_LREAL(_E)*1
無符號BIN16位資料→雙精度實數轉換	UINT2DBL(P)	—(WORD_TO_DINT(_E), DINT_TO_LREAL(_E))*1
有符號BIN32位資料→雙精度實數轉換	DINT2DBL(P)	DINT_TO_LREAL(_E)*1
無符號BIN32位資料→雙精度實數轉換	UDINT2DBL(P)	—(DWORD_TO_DINT(_E), DINT_TO_LREAL(_E))*1
單精度實數→雙精度實數轉換	FLT2DBL(P)	REAL_TO_LREAL(_E)*1

*1　在字元串的傳送，或為可自動轉換的資料類型 （☞ 26頁 自動進行的類型轉換）時，可以祇通過代入進行表述。

運算符可使用的陳述式

算術運算符可使用的陳述式

類型	陳述式符號
BIN16位加法運算	+(P)(_U) [2個操作數時]
BIN16位減法運算	-(P)(_U) [2個操作數時]
BIN32位加法運算	D+(P)(_U) [2個操作數時]
BIN32位減法運算	D-(P)(_U) [2個操作數時]
BIN16位乘法運算	*(P)(_U)
BIN16位除法運算	/(P)(_U)
BIN32位乘法運算	D*(P)(_U)
BIN32位除法運算	D/(P)(_U)

類型	陳述式符號
BCD4位加法運算	B+(P) [2個操作數時]
BCD4位減法運算	B-(P) [2個操作數時]
BCD8位加法運算	DB+(P) [2個操作數時]
BCD8位減法運算	DB-(P) [2個操作數時]
BCD4位乘法運算	B*(P)
BCD4位除法運算	B/(P)
BCD8位乘法運算	DB*(P)
BCD8位除法運算	DB/(P)
BIN16位塊資料加法運算	BK+(P)(_U)
BIN16位塊資料減法運算	BK-(P)(_U)
BIN32位塊資料加法運算	DBK+(P)(_U)
BIN32位塊資料減法運算	DBK-(P)(_U)
字元串的合併	$+(P) [2個操作數時]
單精度實數加法運算	E+(P) [2個操作數時]
單精度實數減法運算	E-(P) [2個操作數時]
雙精度實數加法運算	ED+(P) [2個操作數時]
雙精度實數減法運算	ED-(P) [2個操作數時]
單精度實數乘法運算	E*(P)
單精度實數除法運算	E/(P)
雙精度實數乘法運算	ED*(P)
雙精度實數除法運算	ED/(P)
時鐘資料的加法運算	DATE+(P)
時鐘資料的減法運算	DATE-(P)
擴展時鐘資料的加法運算	S(P).DATE+
擴展時鐘資料的減法運算	S(P).DATE-

邏輯運算符、比較運算符可使用的陳述式

類型	陳述式符號
運算開始、串聯連接、並列連接	LD、LDI、AND、ANI、OR、ORI
梯形圖塊串聯連接、並列連接	ANB、ORB
BIN16位資料比較	LD□(_U)、AND□(_U)、OR□(_U)
BIN32位資料比較	LDD□(_U)、ANDD□(_U)、ORD□(_U)
BIN16位塊資料比較	BKCMP□(P)(_U)
BIN32位塊資料比較	DBKCMP□(P)(_U)
16位資料邏輯與	WAND(P) [2個操作數時]
32位資料邏輯與	DAND(P) [2個操作數時]
16位資料邏輯或	WOR(P) [2個操作數時]
32位資料邏輯或	DOR(P) [2個操作數時]
16位資料邏輯異或	WXOR(P) [2個操作數時]
32位資料邏輯異或	DXOR(P) [2個操作數時]
16位資料邏輯異或非	WXNR(P) [2個操作數時]
32位資料邏輯異或非	DXNR(P) [2個操作數時]
字元串比較	LD$□、AND$□、OR$□
單精度實數比較	LDE□、ANDE□、ORE□
雙精度實數比較	LDED□、ANDED□、ORED□
日期比較	LDDT□、ANDDT□、ORDT□
時間比較	LDTM□、ANDTM□、ORTM□

控制語句、函數等可使用的陳述式

類型	陳述式符號
結構化陳述式	FOR、NEXT
指針分支	CJ、SCJ、JMP
跳轉至END	GOEND
从中斷程式返回	IRET
FOR～NEXT強制結束	BREAK(P)
子程式調用	CALL(P)
从子程式返回	RET
子程式的輸出OFF調用	FCALL(P)
程式檔案間子程式調用	ECALL(P)
程式檔案間子程式輸出OFF調用	EFCALL(P)
子程式調用	XCALL

不需要的陳述式

類型	陳述式符號
順控程式結束	END
無處理 （NOP）	NOP、NOPLF

索引

B

BOOL 25

C

CASE 21,86

E

EN/ENO 44,45,93

F

FB. 12,45,48,92
FOR 24,86

I

IF. 19,86

R

REPEAT 22,86
RETURN 86
RETURN語句 13

S

ST編輯器 39

W

WHILE 22,86

三畫

子程式控制語句 13,86

四畫

內嵌 53
分隔符 11
分層 13,86

八畫

変数 11
函數 12,44,46,92
注釋 11,41,87
空語句 13
表達式 14
返回值 93
返回值的類型 44

十一畫

參數 44,45,92
常數 11,90
控制語法 13,40,86
軟元件 88

十二畫

循環語句 13,22,86

十三畫

運算符. 11,87

十四畫

監視. 52
語句. 13,86

十五畫

標記. 11
標簽. 42,89
調用語句 13,86
賦值語句 13,15,86

十六畫

選擇語句 13,19,86

十八畫

轉換. 50

十九畫

類型轉換 26

索

備忘録

修訂記錄

*本手冊編號在封底的左下角。

修訂日期	*手冊編號	修訂內容
2015年5月	SH(NA)-081466CHT-A	第一版
2016年9月	SH(NA)-081466CHT-B	■第二版 部分修改
2018年2月	SH(NA)-081466CHT-C	■第三版 部分修改

日文手冊編號： SH-081445-E

商標

Unicode is either a registered trademark or a trademark of Unicode, Inc. in the United States and other countries.

The company names, system names and product names mentioned in this manual are either registered trademarks or trademarks of their respective companies.

In some cases, trademark symbols such as '™' or '®' are not specified in this manual.

SH(NA)-081466CHT-C(1802)STC
MODEL: R-ST-GUIDE-CHT

MITSUBISHI ELECTRIC CORPORATION

HEAD OFFICE : TOKYO BUILDING, 2-7-3 MARUNOUCHI, CHIYODA-KU, TOKYO 100-8310, JAPAN
NAGOYA WORKS : 1-14 , YADA-MINAMI 5-CHOME , HIGASHI-KU, NAGOYA , JAPAN