20 機能をプログラミングしたい

（部品を使わないプログラミング）

この章では、GP-Pro EX の「スクリプトを使ったプログラミング」についての基本的な説明と、スクリプトの作成方法について説明します。
まず「20.1 設定メニュー」（20-2 ページ）をお読みいただき、目的に合った説明ページへ読み進んでください。

20.1 設定メニュー

D スクリプトはお客様自身でプログラムできる簡易言語です。この機能を使うと、GP 内部で演算を行ったり、未対応の周辺機器と通信させたりできます。

⚠警告

D スクリプト／グローバル D スクリプトでは、人命や重大な物的損傷にかかわる制御は決して行わないでください。

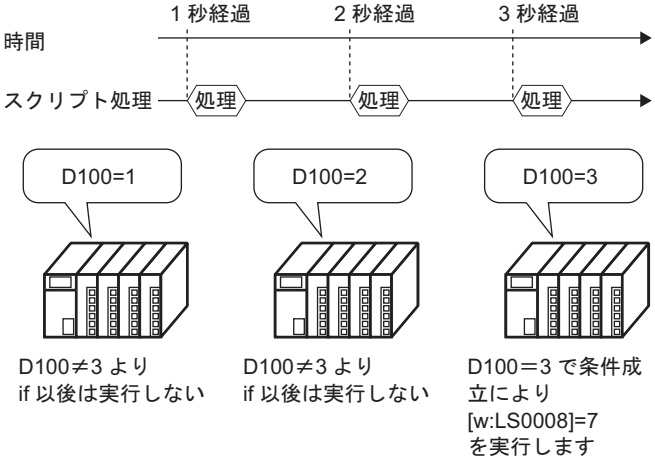
MEMO

- D スクリプトはベース画面に対して設定します。そのベース画面を表示中に条件をみてプログラムを実行します。
- グローバル D スクリプトは表示画面に関係なく GP が運転中、条件をみてプログラムを実行します。
- 拡張スクリプトは、より高度な通信プログラムを作成するために使用します。
- スクリプト以外に、ロジックプログラムを使用した制御もできます。

☞「28.1 設定メニュー」（28-2 ページ）

条件付きで演算したい

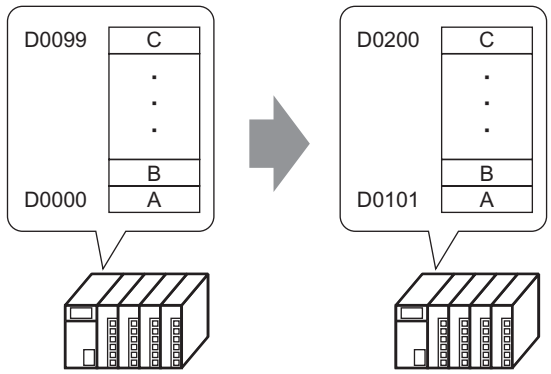
3 秒後に自動で画面番号 7 に画面が切り替わるスクリプトを作成します。



☞ **設定手順**（20-6 **ページ**）
☞ **詳細**（20-5 **ページ**）

データをまとめてコピーしたい

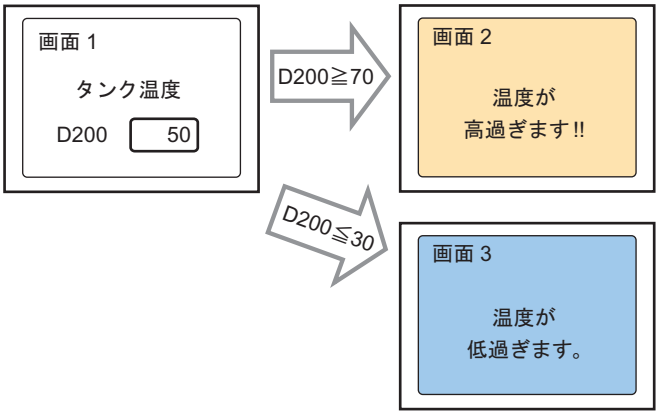
ビットアドレス M0100 の立ち上がりを検出し、接続機器に格納されているデータを他のアドレスへコピーするスクリプトを作成します。



☞ **設定手順**（20-13 **ページ**）
☞ **詳細**（20-12 **ページ**）

エラーが発生すると警告を出したい

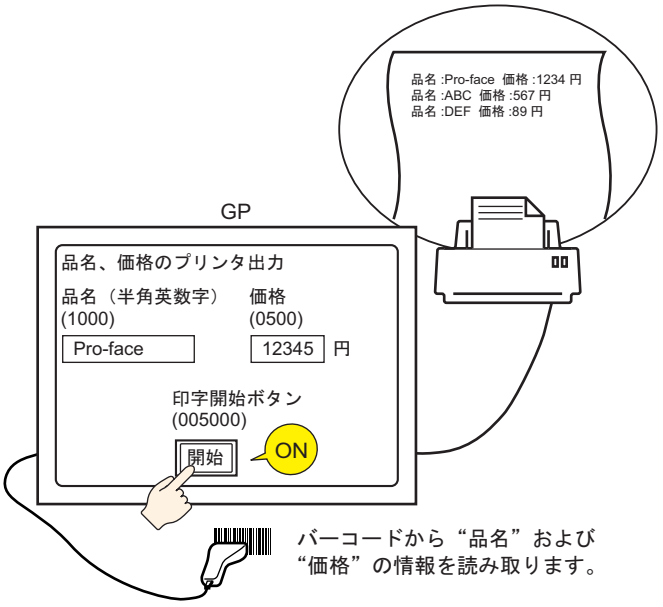
温度管理のシステムにおいて、接続機器からのエラービット M0001 を検出し、温度情報格納アドレス D200 が 70 度以上の場合と 30 度以下の場合にそれぞれの警告メッセージの表示を行います。また、エラーを検出した回数もカウントするスクリプトを作成します。



☞ 設定手順（20-18 ページ）
☞ 詳細（20-17 ページ）

対応していない周辺機器と通信させたい

バーコードを USB に接続し読み取ったデータを、COM1 に接続したシリアルプリンタへ出力する拡張スクリプトを作成します。



☞ 設定手順（20-34 ページ）
☞ 詳細（20-21 ページ）

20.2 条件付きで演算したい

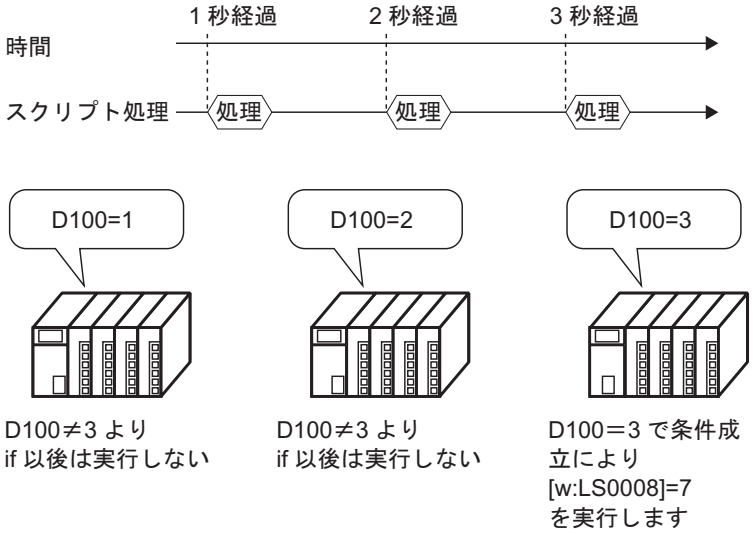
MEMO

・設定内容の詳細は設定ガイドを参照してください。

☞「20.8.1 D スクリプト / 共通設定 [グローバル D スクリプト設定] の設定ガイド」(20-48 ページ)

動作

3 秒後に自動で画面番号 7 に画面が切り替わるスクリプトを作成します。

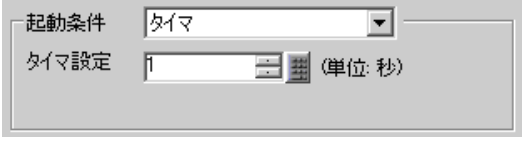


使用する命令

命令	動作概要
代入（=）	左辺に右辺の値を代入します。
加算（+）	ワードデバイスのデータと定数の加算を実行します。
if **（）**	if に続く（）内の条件式が成立時、if（）より後の処理を実行します。
等しい（==）	左辺と右辺の値を比較します。左辺 = 右辺ならば真となります。
LS0008	格納された値の画面番号に切り替えます。 ☞**「付録** 1.4.2 **システムデータエリア」**(A-9 **ページ)**

起動条件

下記のようにタイマを選択し、[タイマ設定] を 1 秒に設定します。



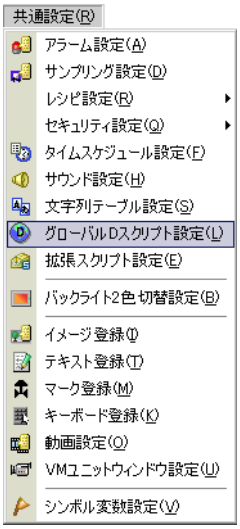
完成スクリプト

実行式　実行式を広く見せる　アドレス入力

```
0001 [w:[PLC1]D00100]=[w:[PLC1]D00100]+1
0002 if([w:[PLC1]D00100]==3)
0003 {
0004     [w:[#INTERNAL]LS0008]=7
0005 }
0006 endif
0007
```

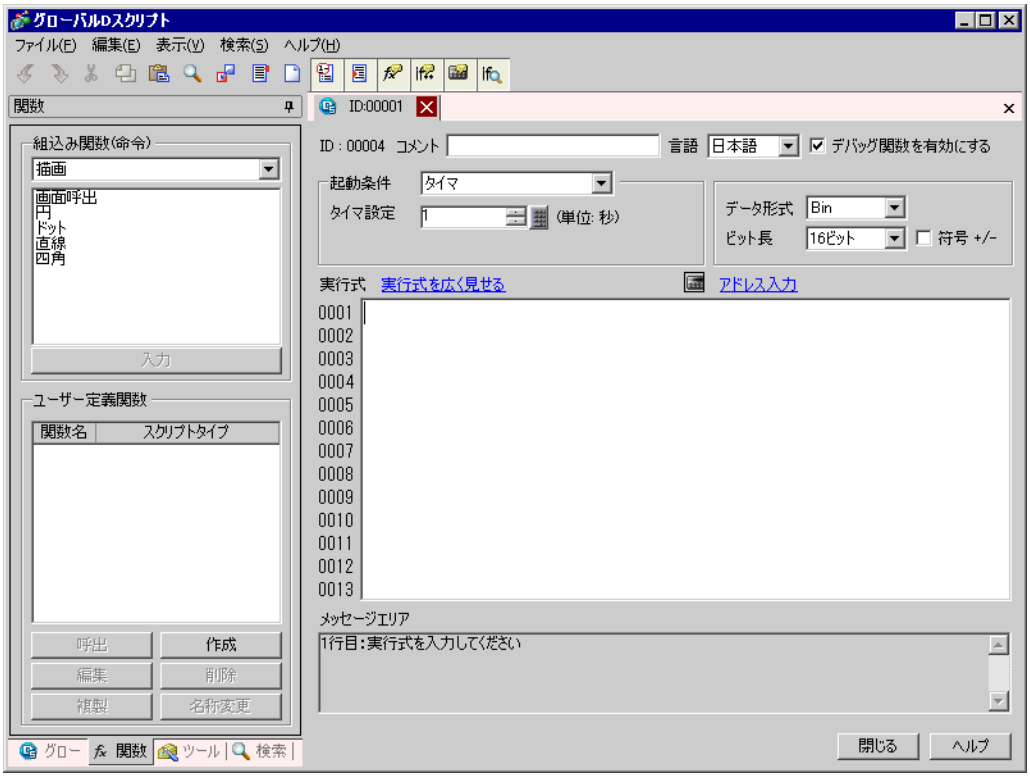
作成手順

1 [共通設定 (R)] メニューの [グローバル D スクリプト設定 (L)] をクリックします。

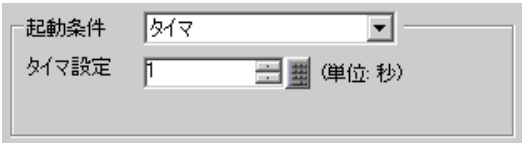


2 [作成] をクリックします。既にスクリプトを登録している場合は、ID 番号を指定して [編集] をクリックします。

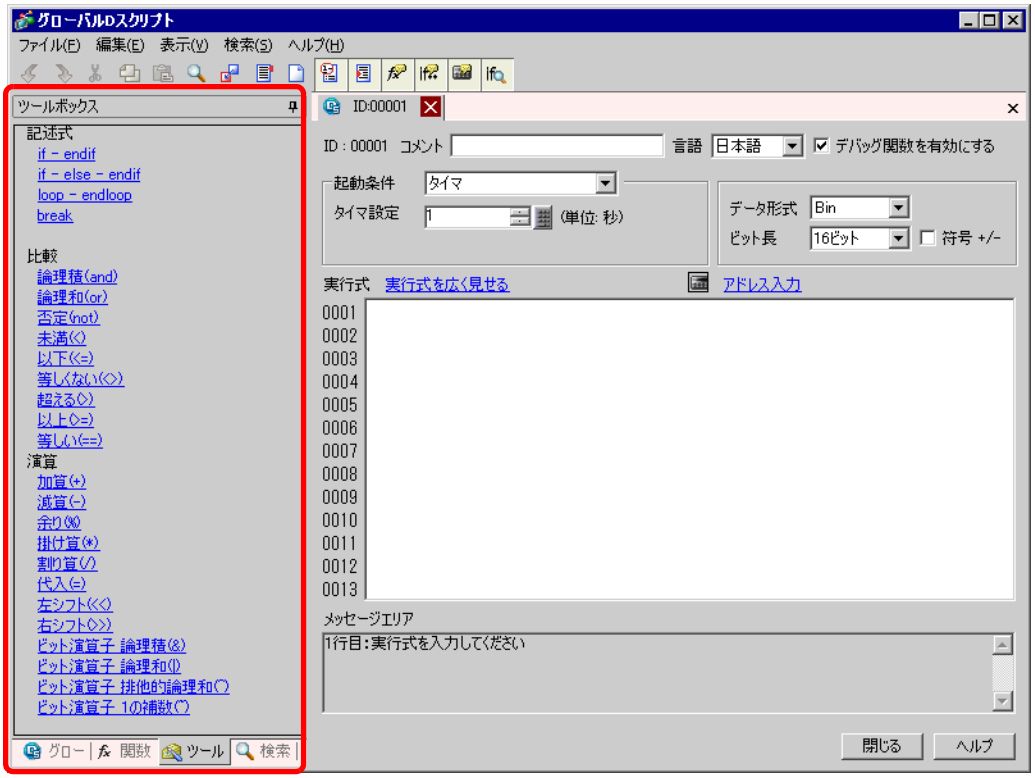
3 [グローバル D スクリプト] ダイアログボックスが表示されます。



4 スクリプトの起動条件（トリガ）で [タイマ] を選択し、[タイマ設定] を 1 秒に指定します。

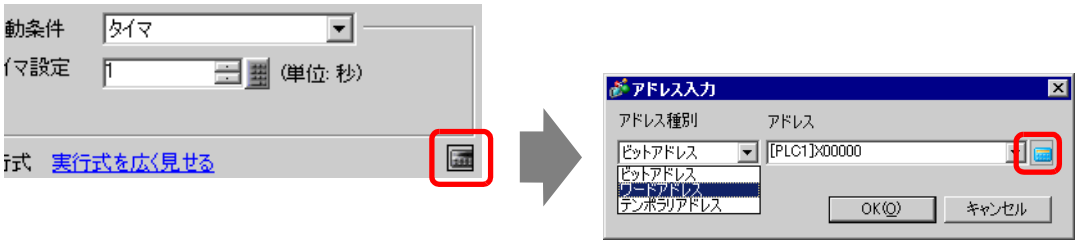


5 [ツールボックス] タブをクリックします。ツールボックスは、スクリプトで使用できる命令をクリックするだけで簡単に配置させることができます。

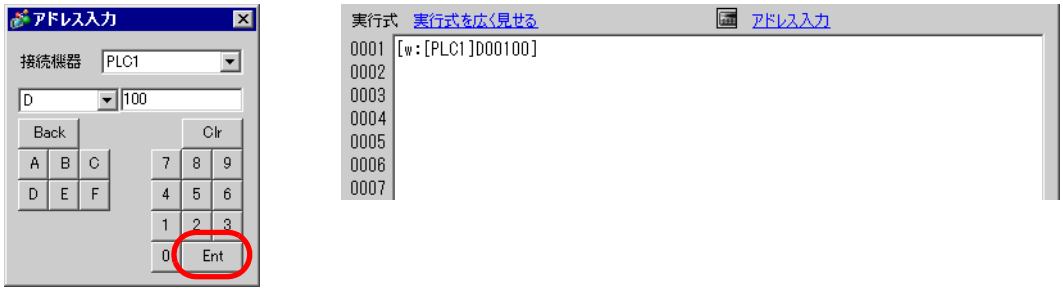


6 1 行目のスクリプトを作成します。1 行目の動作は、D00100 の初期値を 0 とすると、処理毎に 1 ずつ増加して格納されるカウント動作です。

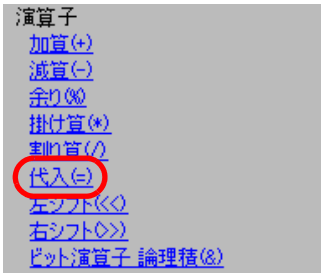
をクリックし、[ワードアドレス] を選択して、 をクリックします。

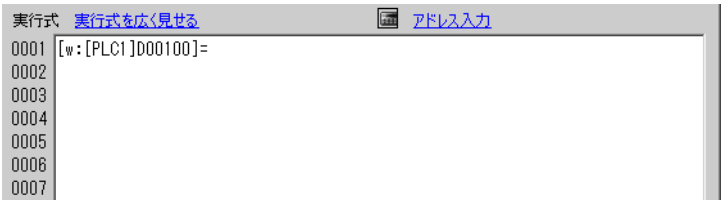


7 D00100 を入力して、[ENT] をクリックします。

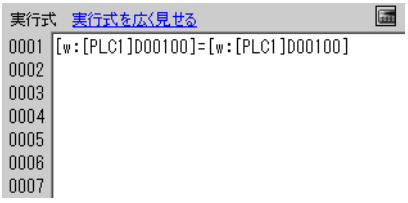


8 ツールボックスの [代入（=）] をクリックします。

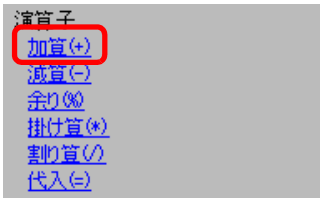


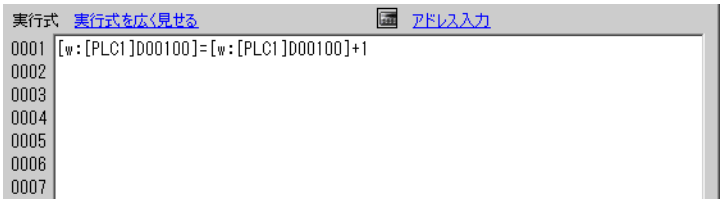


9 手順 6 ～ 7 と同様に D00100 を配置します。



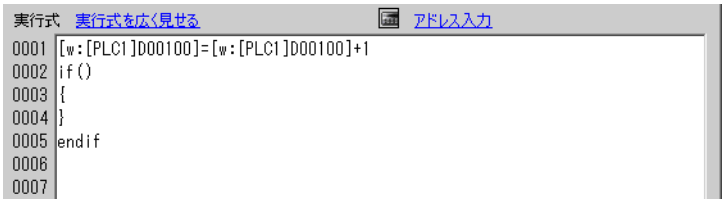
10 加算（+）をクリックし、「1」を入力して 1 行目は完成です。



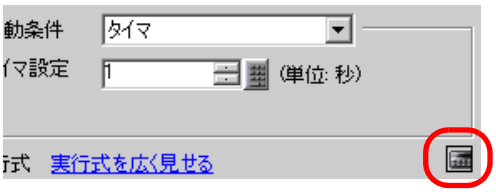


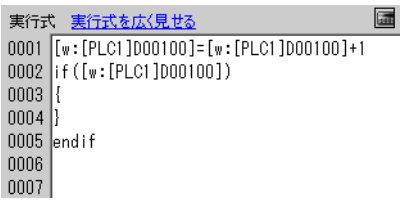
11 2 行目のスクリプトを作成します。2 行目の動作は、if に続く（）内の条件式が成立時、if（）より後の処理を実行します。

[if - endif] をクリックします。

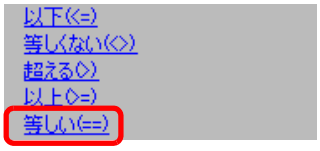


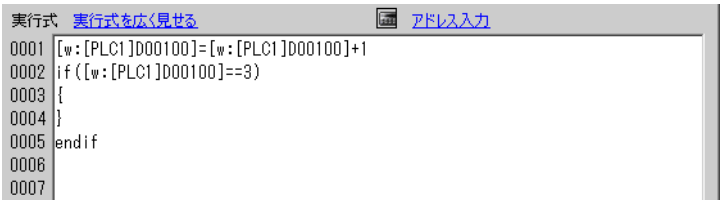
12 if に続く（）内の条件式を作成します。条件式は、D00100 に格納されている値と「3」を比較し、等しい場合に条件成立となります。
カーソルを（）内に合わせ、手順 6 ～ 7 と同様に D00100 を配置します。



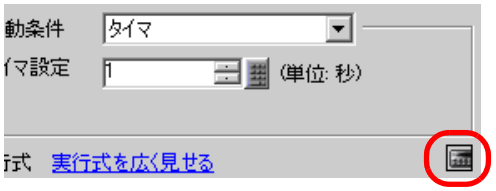


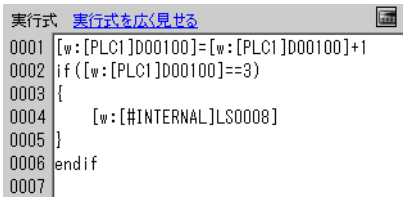
13 [等しい（==）] をクリックし、「3」を入力して 2 行目は完成です。



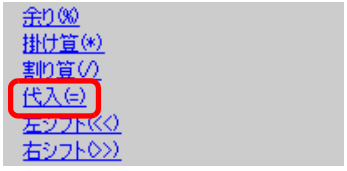


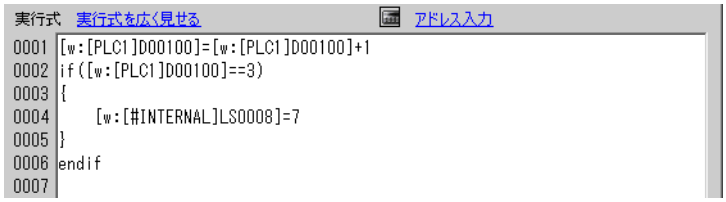
14 カーソルを {} 内に合わせ、改行します。手順 6 ～ 7 と同様に LS0008 を配置します。



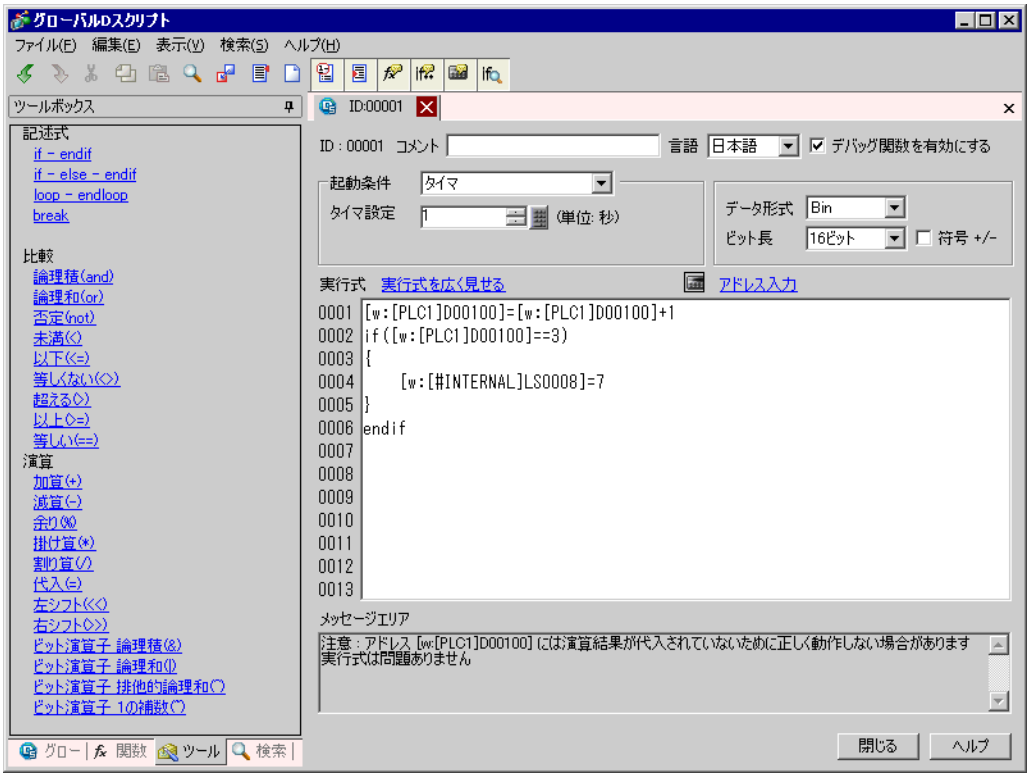


15 [代入（=）] をクリックし、「7」を入力します。





16 完成です。



MEMO

- 文字列選択時に［Ctrl］キー＋［Shift］キー＋［→］キー／［←］キーを押すと、テキストブロックの最後まで選択できます。
- ［Ctrl］キー＋［F4］キーを押すと、現在選択している画面を閉じます。
- ［Esc］キーを押すと、スクリプトを上書き保存／破棄して終了します。

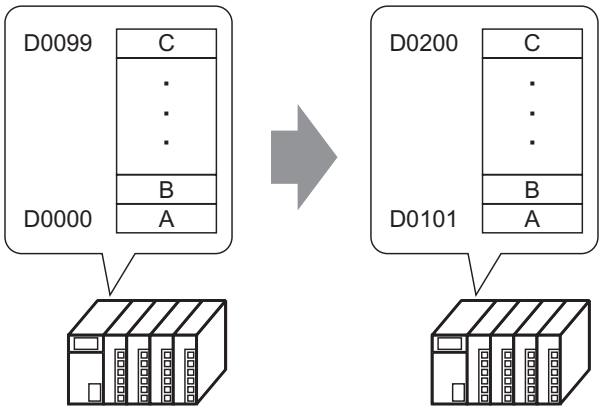
20.3 データをまとめてコピーしたい

MEMO
- 設定内容の詳細は設定ガイドを参照してください。

☞「20.8.1 D スクリプト / 共通設定 [グローバル D スクリプト設定] の設定ガイド」(20-48 ページ)

動作

ビットアドレス M0100 の立ち上がりを検出し、接続機器に格納されているデータを他のアドレスへコピーするスクリプトを作成します。

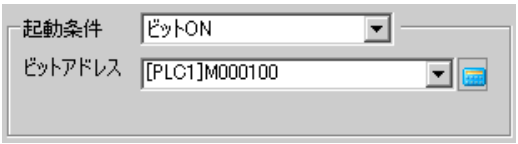


使用する命令

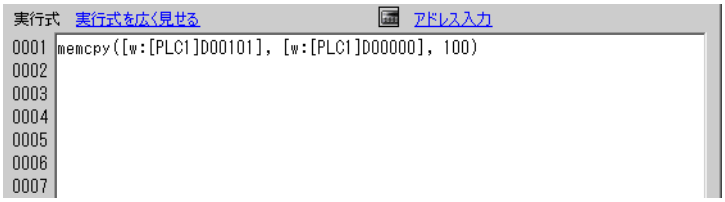
命令	動作概要
メモリコピー memcpy()	デバイスに格納されている値を一括コピーします。 コピー元ワードアドレスからアドレス数分のデータをコピー先ワードアドレスにコピーします。 **[書式]** memcpy([コピー先ワードアドレス],[コピー元ワードアドレス], アドレス数)

起動条件

下記のように立ち上がりを選択し、[ビットアドレス] を M000100 に設定します。

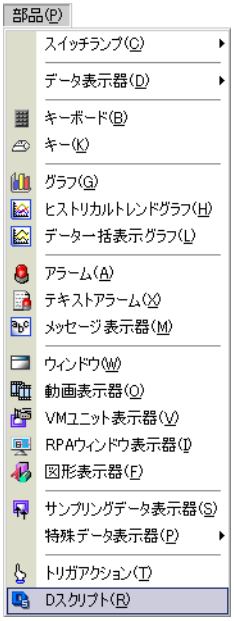


完成スクリプト

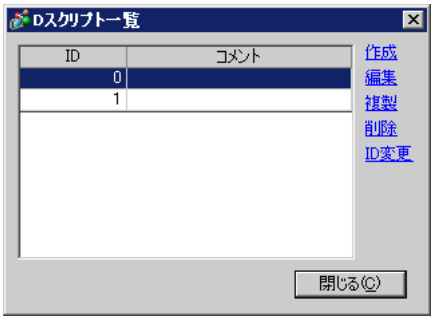


作成手順

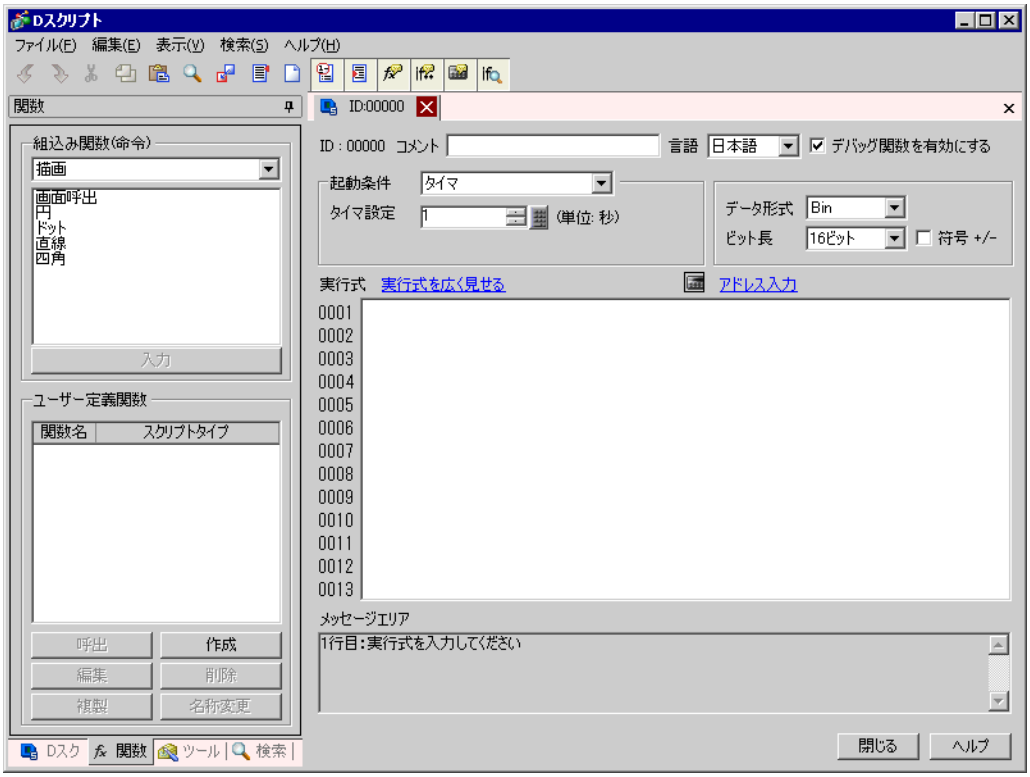
1 [部品 (P)] メニューの [D スクリプト (R)] をクリックするか、 をクリックします。



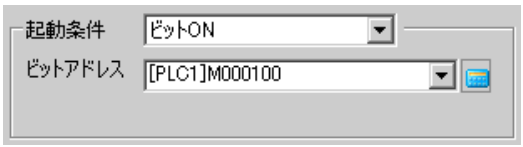
2 [作成] をクリックします。既に登録されている場合、ID 番号が表示されます。



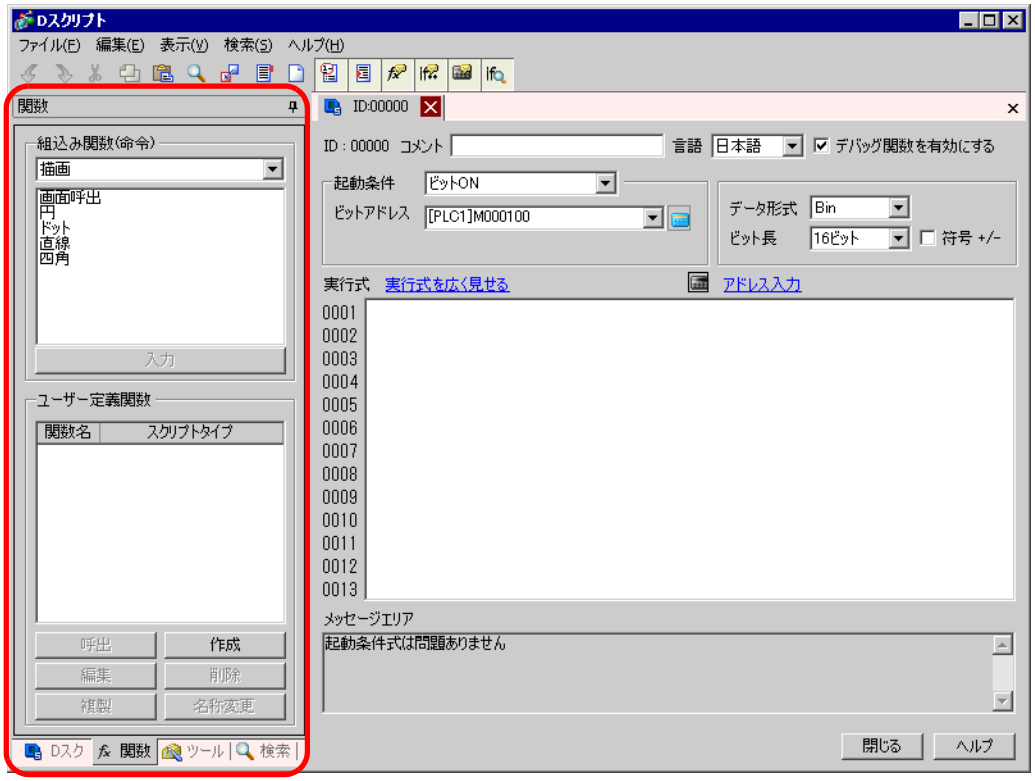
3 [D スクリプト] ダイアログボックスが表示されます。



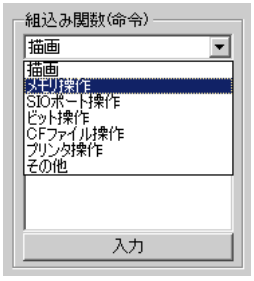
4 スクリプトの起動条件（トリガ）で [ビット ON] を選択し、[ビットアドレス] は M000100 を指定します。



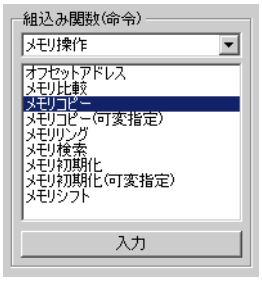
5 [関数] タブをクリックします。組み込み関数（命令）は、スクリプトで使用できる命令をクリックするだけで簡単に配置させることができます。

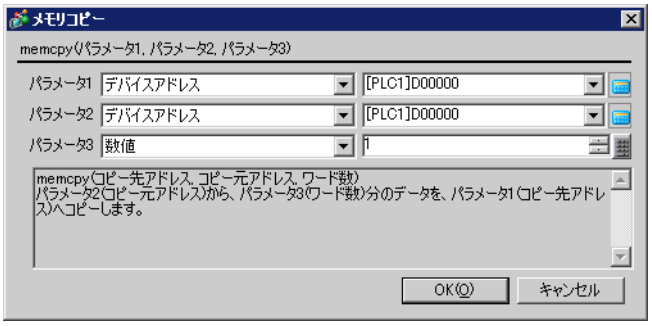


6 [組み込み関数（命令）] のプルダウンメニューから [メモリ操作] を選択します。

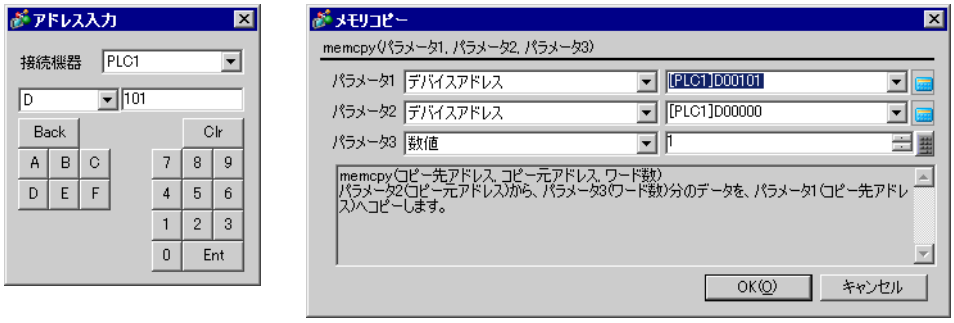


7 [メモリコピー] をダブルクリックし、以下ダイアログボックスでコピー先のアドレス、コピー元のアドレス、アドレス数を指定します。 をクリックします。

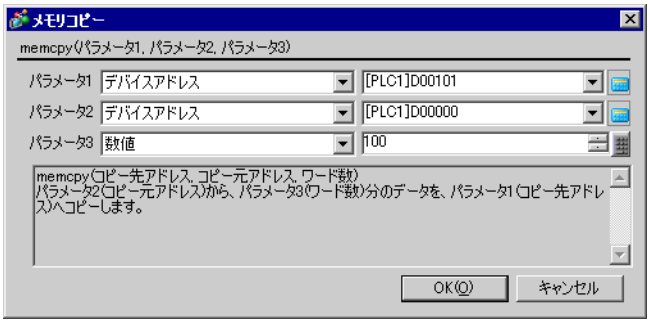




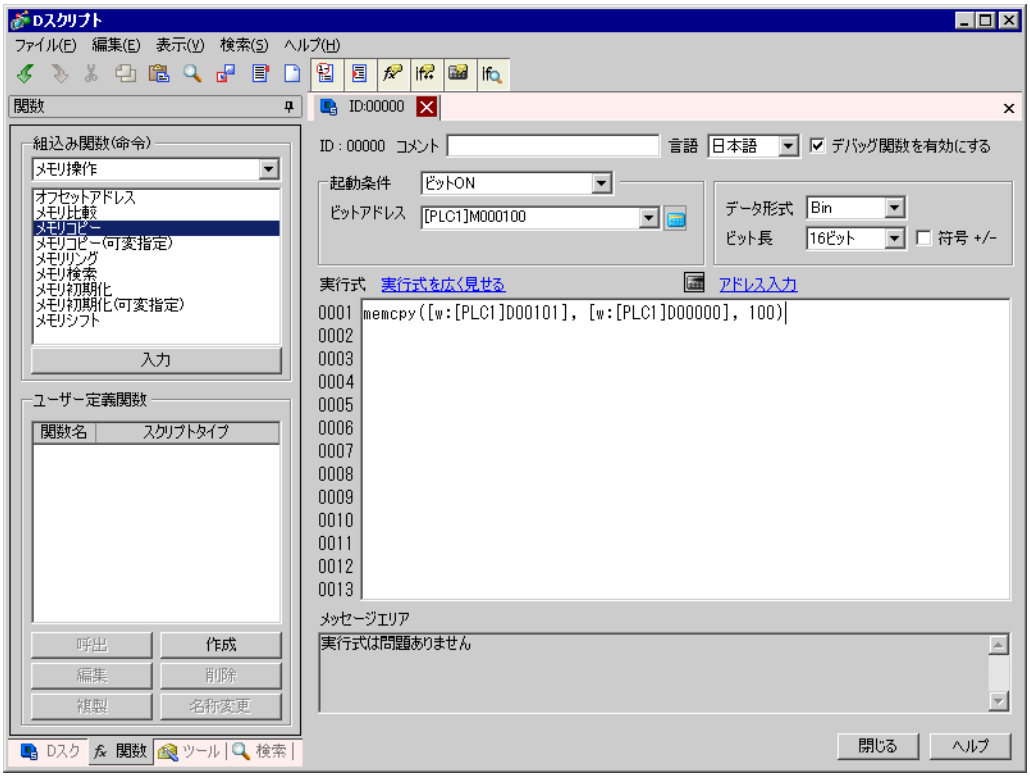
8 D00101 を入力して、[ENT] をクリックします。



9 アドレス数に 100 を入力し、手順 8 と同様にコピー元ワードアドレスに D00000 を指定して [OK] をクリックします。



10 完成です。



20.4 エラーが発生すると警告を出したい

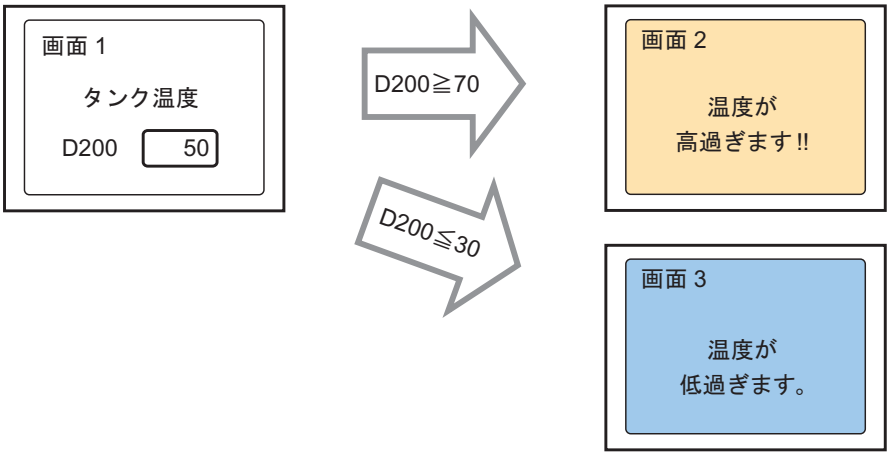
MEMO

・設定内容の詳細は設定ガイドを参照してください。

☞「20.8.1 D スクリプト / 共通設定 [グローバル D スクリプト設定] の設定ガイド」(20-48 ページ)

動作

温度管理のシステムにおいて、接続機器からのエラービット M0001 を検出し、温度情報格納アドレス D200 が 70 度以上の場合と 30 度以下の場合にそれぞれの警告メッセージの表示を行います。また、エラーを検出した回数もカウントするスクリプトを作成します。



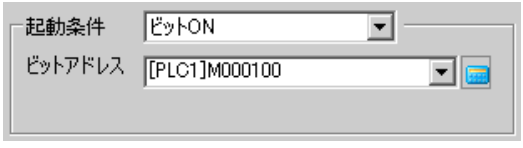
D200 が 70 度以上になる度にカウントし、その回数を格納するアドレス　：LS0300
D200 が 30 度以下になる度にカウントし、その回数を格納するアドレス　：LS0301
警告画面を表示するための画面番号を格納するアドレス　：LS0302

使用する命令

命令	動作概要
If ()	if に続く（）内の条件式が成立時、if（）より後の処理を実行します。
以上（>=）	N1>=N2（N1 ≧ N2）ならば真となります。
代入（=）	左辺に右辺の値を代入します。
加算（+）	ワードデバイスのデータと定数の加算を実行します。
以下（<=）	N1<=N2（N1 ≦ N2）ならば真となります。

起動条件

下記のように立ち上がりを選択し、[ビットアドレス] を M000100 に設定します。



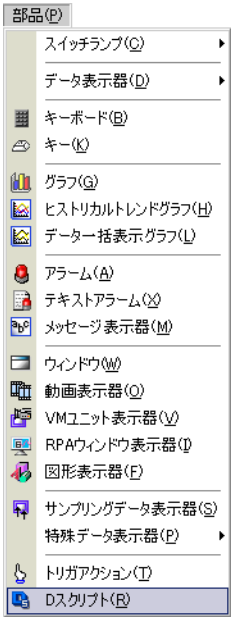
完成スクリプト

実行式　実行式を広く見せる　アドレス入力

```
0001 if([w:[PLC1]D00200]>=70)                          //温度が70度以上の場合
0002 {
0003     [w:[#INTERNAL]LS0302]=100                     //70度以上の警告メッセージ画面番号100代入
0004     [w:[#INTERNAL]LS0300]=[w:[#INTERNAL]LS0300]+1 //エラー回数カウントアップ
0005 }
0006 endif
0007
0008 if([w:[PLC1]D00200]<=30)                          //温度が30度以下の場合
0009 {
0010     [w:[#INTERNAL]LS0302]=101                     //30度以下の警告メッセージ画面番号101代入
0011     [w:[#INTERNAL]LS0301]=[w:[#INTERNAL]LS0301]+1 //エラー回数カウントアップ
0012 }
0013 endif
0014
0015
0016
0017
```

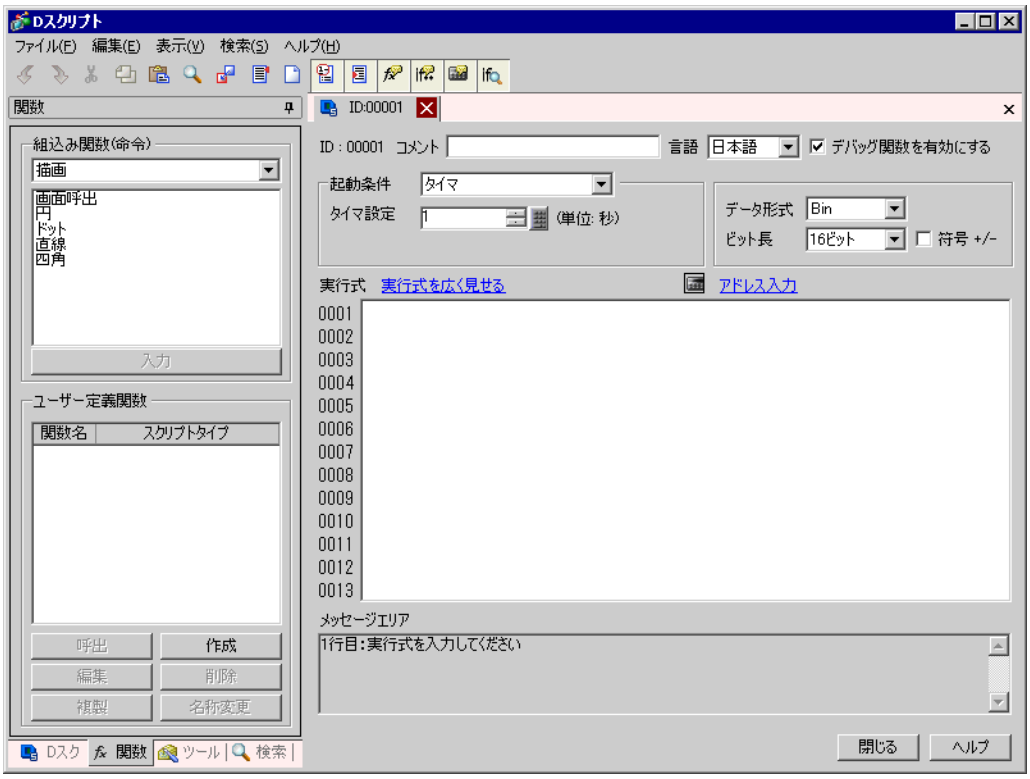
作成手順

1 [部品] メニューの [D スクリプト (R)] をクリックするか、 をクリックします。



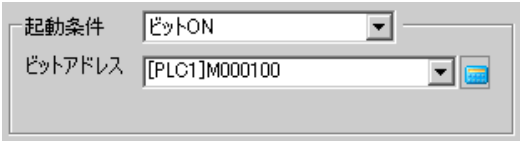
2 [作成] をクリックします。既に登録されている場合、ID 番号が表示されます。

3 [D スクリプト] ダイアログボックスが表示されます。

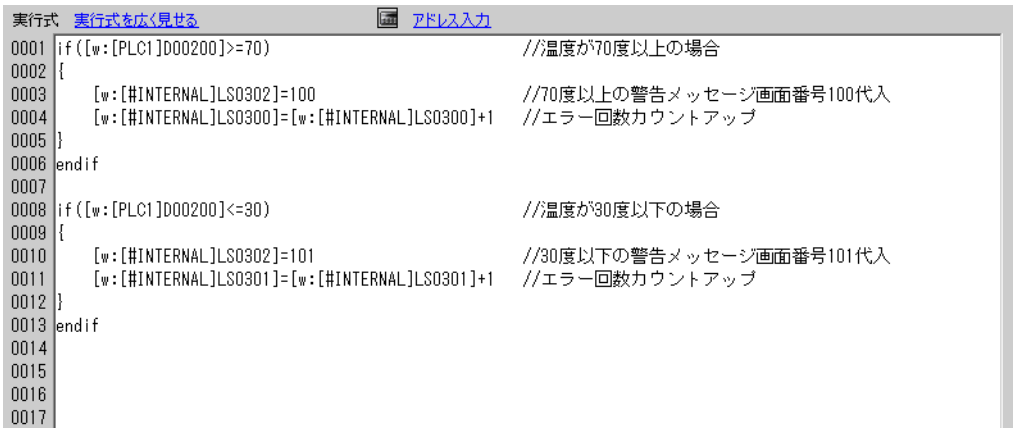


4 コメントを設定します。「警告表示」と入力します。

5 スクリプトの起動条件（トリガ）で [ビット ON] を選択し、[ビットアドレス] は M00100 を指定します。



6 コマンド、記述式、定数入力から実行部にプログラムを記述して完成です。



MEMO

- 文字列選択時に［Ctrl］キー＋［Shift］キー＋［→］キー／［←］キーを押すと、テキストブロックの最後まで選択できます。
- ［Ctrl］キー＋［F4］キーを押すと、現在選択している画面を閉じます。
- ［Esc］キーを押すと、スクリプトを上書き保存／破棄して終了します。

20.5 対応していない周辺機器と通信させたい

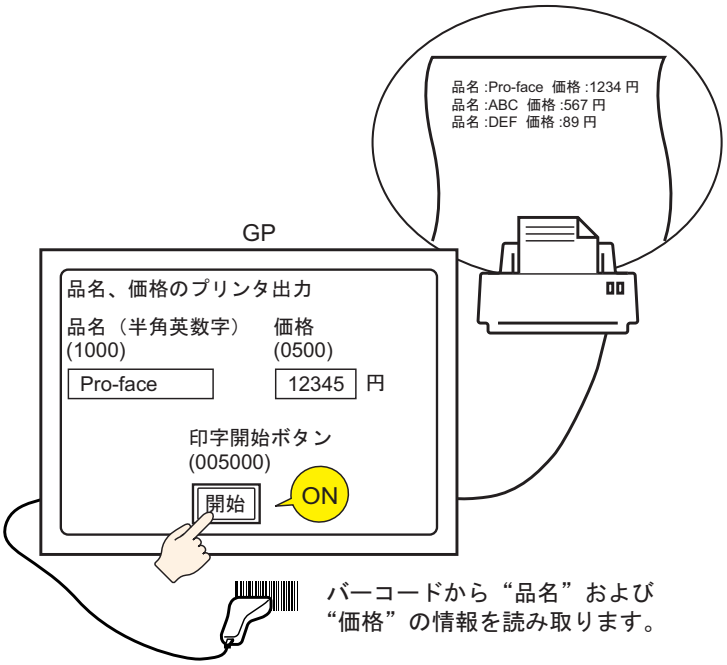
MEMO

・ 設定内容の詳細は設定ガイドを参照してください。

☞「20.8.1 D スクリプト / 共通設定 [グローバル D スクリプト設定] の設定ガイド」(20-48 ページ)

■ 動作

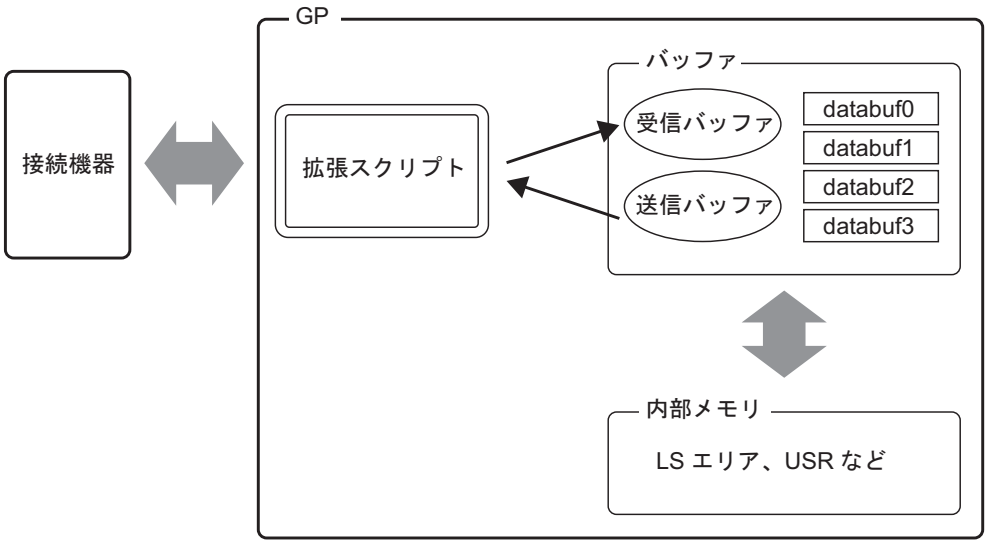
バーコードを USB に接続し読み取ったデータを、COM1 に接続したシリアルプリンタへ出力する拡張スクリプトを作成します。



■ 拡張スクリプトのしくみ

拡張スクリプトは、GP に内蔵されたシリアルポートと接続された入出力機器との通信専用のスクリプトです。

拡張スクリプトの処理は、下図のように送信バッファ / 受信バッファを通して databuf0 ～ databuf3 へデータを格納して処理を行います。databuf はアドレス（番地）で区分けされていないので、接続機器からのデータを編集する場合は、内部メモリに格納してから行うようにしてください。



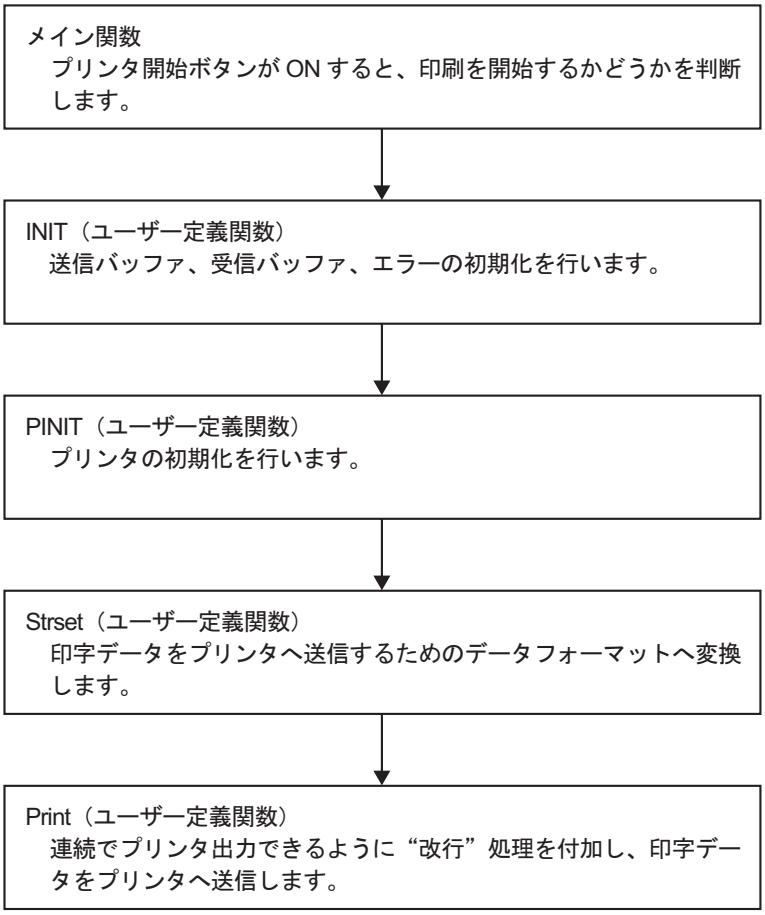
受信バッファ / 送信バッファ

接続機器との通信に関して、データ送受信の有無をリアルタイムで判別する bit 型のメモリ領域になります。

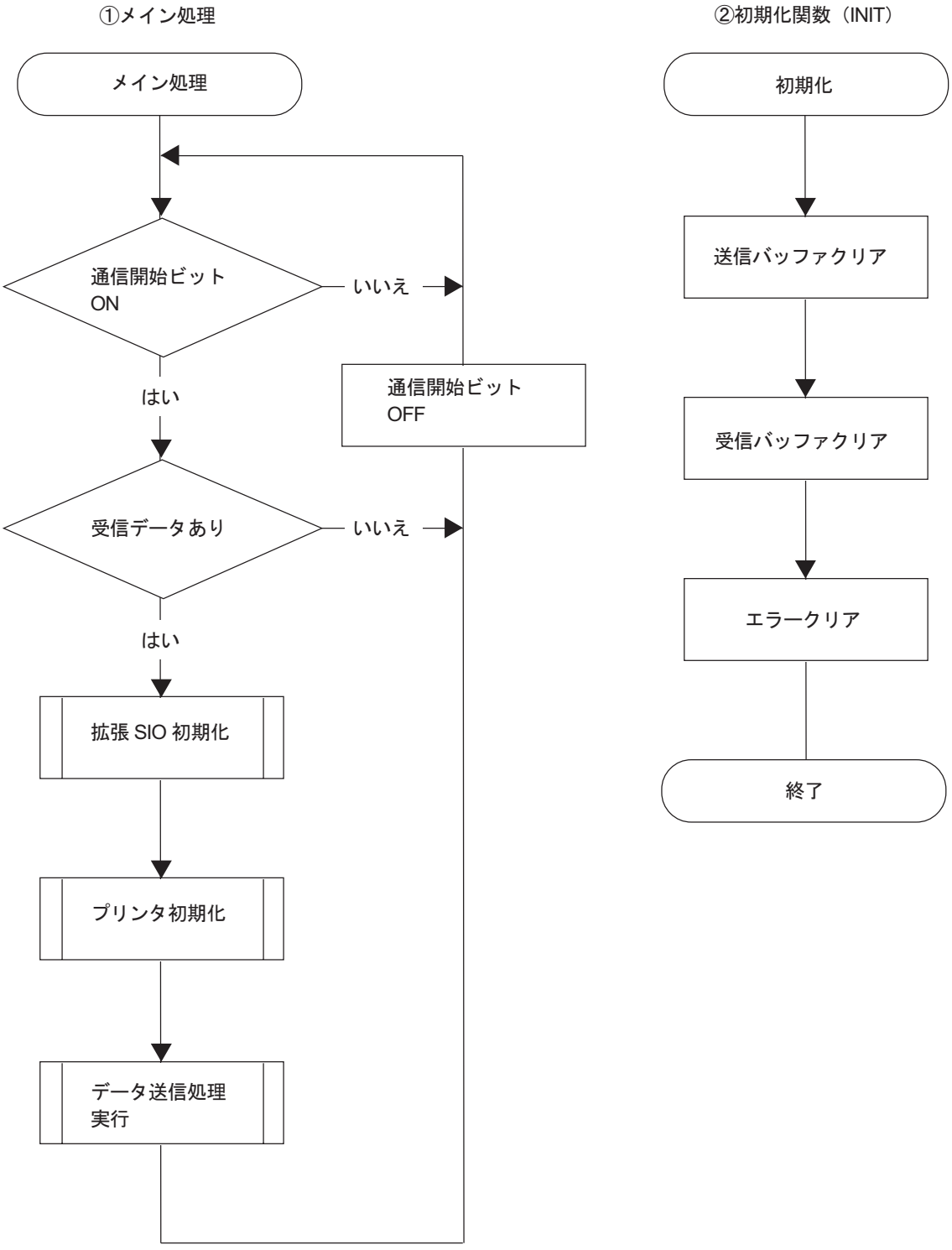
databuf0 ～ databuf3

データ格納場所としての byte 型（8bit）のメモリ領域になります。各バッファのサイズは、1K バイトです。

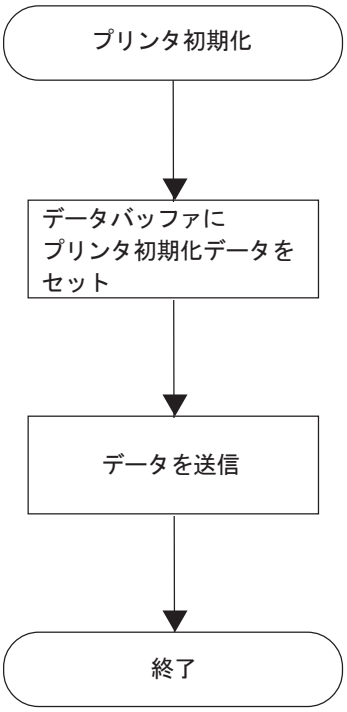
■ スクリプト処理の流れ


メイン関数
プリンタ開始ボタンが ON すると、印刷を開始するかどうかを判断します。
INIT（ユーザー定義関数）
送信バッファ、受信バッファ、エラーの初期化を行います。
PINIT（ユーザー定義関数）
プリンタの初期化を行います。
Strset（ユーザー定義関数）
印字データをプリンタへ送信するためのデータフォーマットへ変換します。
Print（ユーザー定義関数）
連続でプリンタ出力できるように“改行”処理を付加し、印字データをプリンタへ送信します。

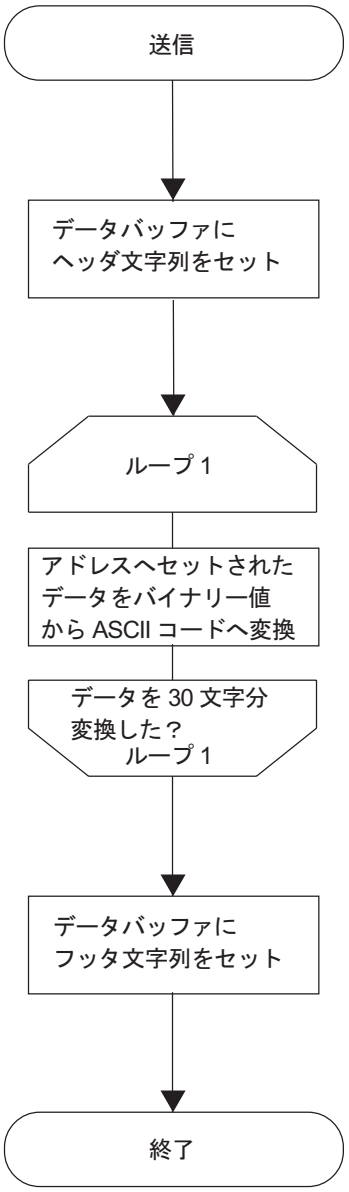
■ フローチャート


①メイン処理
メイン処理
通信開始ビット
ON
いいえ
はい
通信開始ビット
OFF
受信データあり
いいえ
はい
拡張 SIO 初期化
プリンタ初期化
データ送信処理
実行
②初期化関数（INIT）
初期化
送信バッファクリア
受信バッファクリア
エラークリア
終了

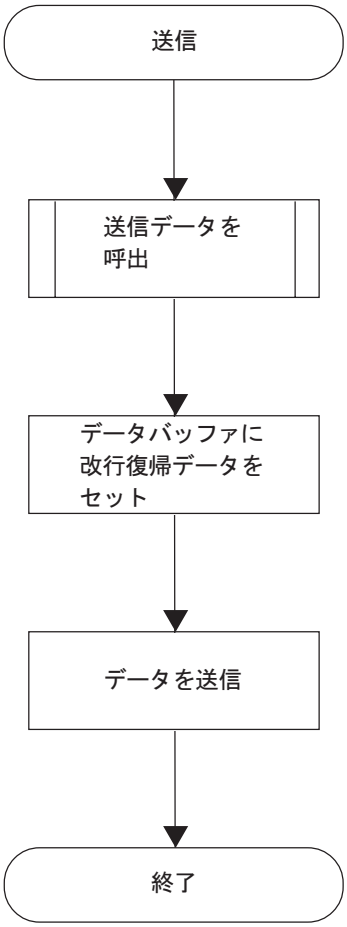
③プリンタ初期化関数（PINIT）



④文字列関数（Strset）



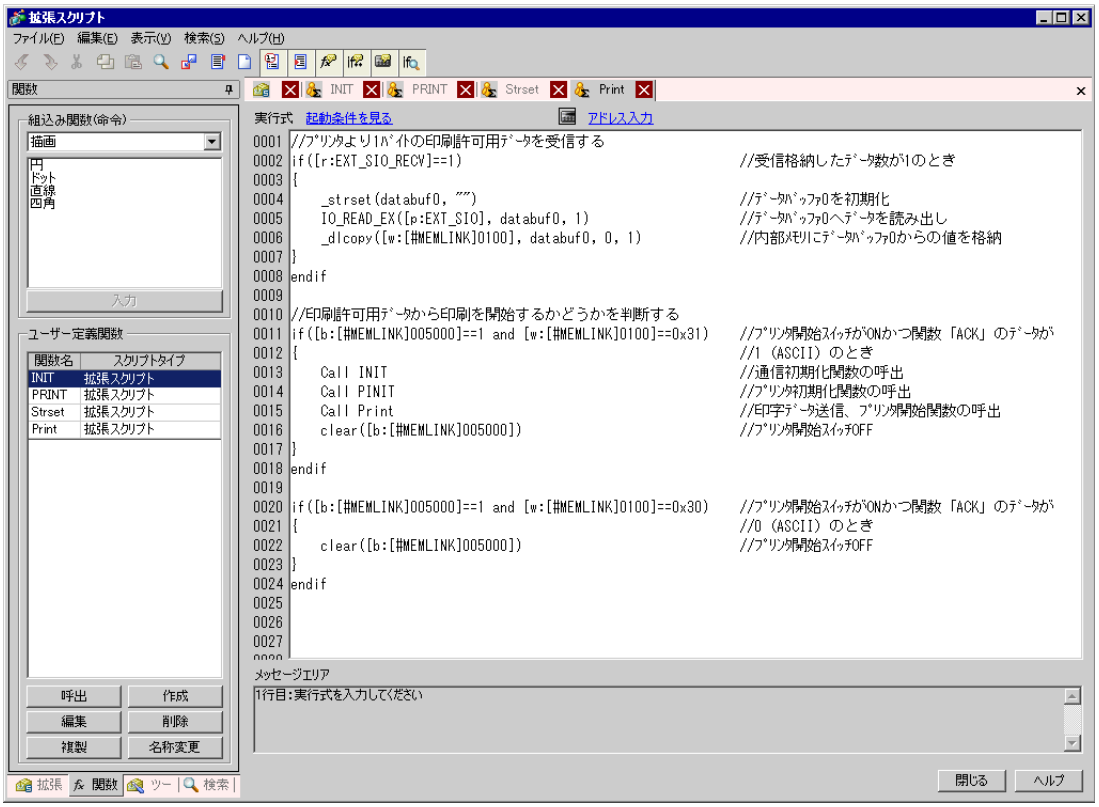
⑤送信関数（Print）



■ スクリプトの動作概要

◆メイン関数

完成スクリプト



動作概要

プリンタ開始ボタン（内部メモリ 005000）が ON すると、プリンタからの 1 バイトの印刷許可用データから印刷を開始するかどうかを判断します。

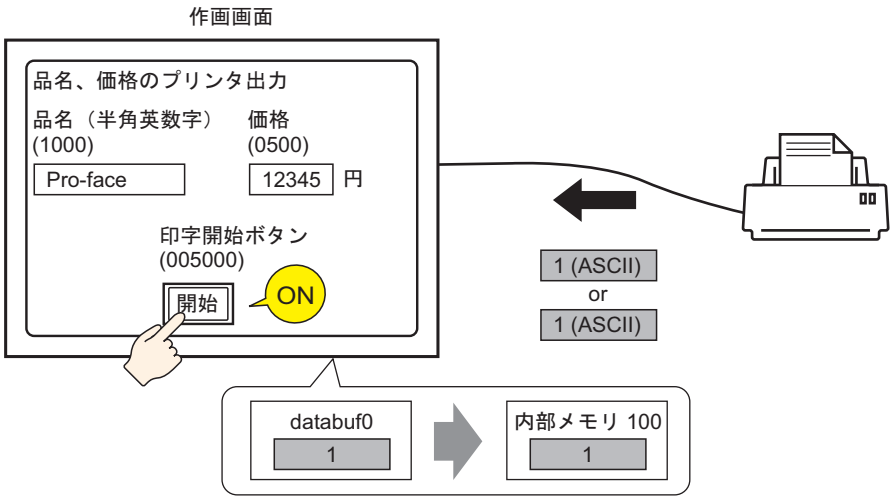
印刷許可用データは、プリンタの仕様例として以下の動作を行うものとします。

印刷準備 OK：0x31（ASCII コードの “1”）を接続機器に対して送信する。

印刷準備 NG：0x30（ASCII コードの “0”）を接続機器に対して送信する。

印刷許可用データを databuf0 に受け取った GP は、以降のスクリプト処理でこのデータを利用しやすい内部メモリ 100 に移動させます。

内部メモリ 100 が 0x31（ASCII コードの “1”）の場合は印刷を開始し、0x30（ASCII コードの “0”）の場合は始めに戻って 0x31 のデータを受け取るまでこの処理を繰り返します。



◆INIT（ユーザー定義関数）

完成スクリプト

実行式　実行式を広く見せる　アドレス入力

```
0001 [c:EXT_SIO_CTRL00]=1    //送信ﾊﾞｯﾌｧｸﾘｱ
0002 [c:EXT_SIO_CTRL01]=1    //受信ﾊﾞｯﾌｧｸﾘｱ
0003 [c:EXT_SIO_CTRL02]=1    //ｴﾗｰｸﾘｱ
0004
```

動作概要

送信バッファ、受信バッファ、エラーの初期化を行います。

◆PINIT（ユーザー定義関数）

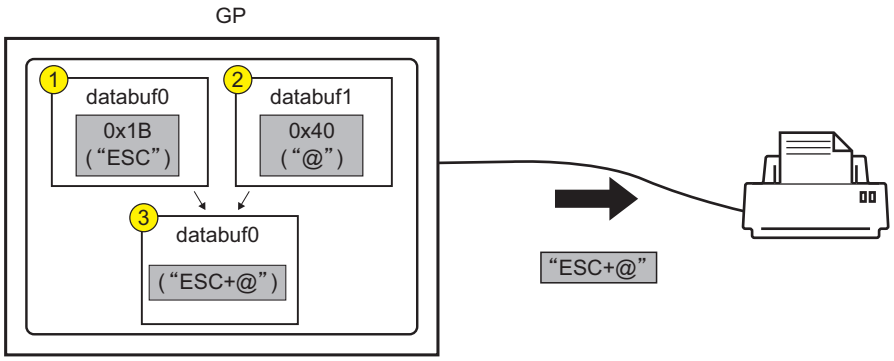
完成スクリプト

実行式　実行式を広く見せる　アドレス入力

```
0001 //ﾌﾟﾘﾝﾀ初期化 ( ESC/Pｺﾏﾝﾄﾞ 「ESC + @」 )
0002
0003 _strset(databuf0, "")           //ﾃﾞｰﾀﾊﾞｯﾌｧ0をｸﾘｱ
0004 _strset(databuf0, 0x1B)         //ASCIIｺｰﾄﾞ"ESC"をｾｯﾄ
0005 _strset(databuf1, "")           //ﾃﾞｰﾀﾊﾞｯﾌｧ1をｸﾘｱ
0006 _strset(databuf1, 0x40)         //ASCIIｺｰﾄﾞ"@"をｾｯﾄ
0007 _strcat(databuf0, databuf1)     //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0008 _strlen([t:0000], databuf0)     //ﾃﾞｰﾀの長さを数値化しﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽへ格納
0009
0010 //ｼﾘｱﾙﾎﾟｰﾄよりﾃﾞｰﾀ送信
0011
0012 IO_WRITE_EX([p:EXT_SIO], databuf0, [t:0000])    //ﾊﾞｯﾌｧ0のﾃﾞｰﾀをﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽの値分送信
0013
```

動作概要

プリンタの初期化を行います。ESC/P コマンド “ESC+@” をプリンタに送信します。



◆Strset（ユーザー定義関数）

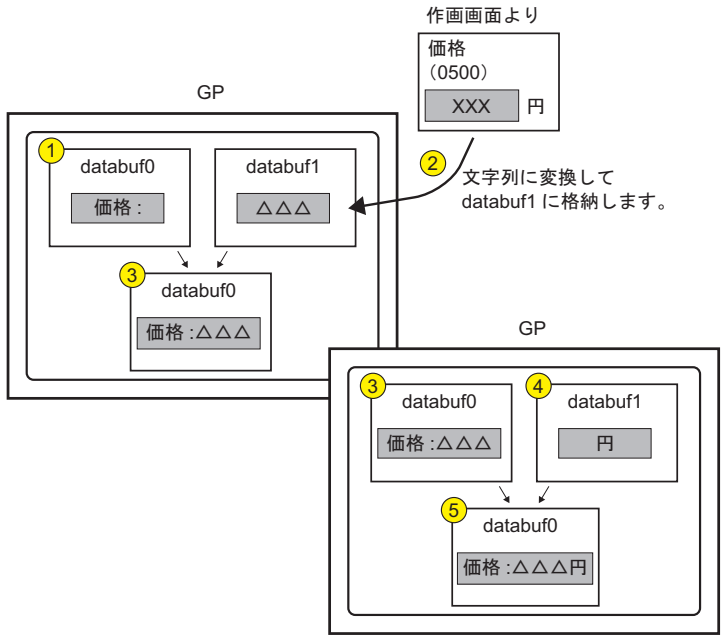
完成スクリプト

実行式　実行式を広く見せる　アドレス入力

```
0001 //文字列”価格 ： ”と”円”を付加する
0002 _strset(databuf0, "")                          //ﾃﾞｰﾀﾊﾞｯﾌｧ0を初期化
0003 _strset(databuf0, "価格 ： ")                  //ﾃﾞｰﾀﾊﾞｯﾌｧ0へ文字列を格納
0004 _bin2decasc(databuf0, [w:[#MEMLINK]0500])      //数値を文字列に変換してﾃﾞｰﾀﾊﾞｯﾌｧ1へ格納
0005 _strcat(databuf0, databuf1)                    //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0006 _strset(databuf1, "")                          //ﾃﾞｰﾀﾊﾞｯﾌｧ1を初期化
0007 _strset(databuf1, "円")                        //ﾃﾞｰﾀﾊﾞｯﾌｧ1へ文字列を格納
0008 _strcat(databuf0, databuf1)                    //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0009
0010 //ﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽの初期化
0011 [t:0001]=0
0012 [t:0002]=0
0013
0014 //内部ﾒﾓﾘにﾜｰﾄﾞ単位で連続格納されている文字列をﾊﾞｲﾄ単位で格納し直す（30文字分）
0015 loop()
0016 {
0017     [w:[#MEMLINK]2000]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] >> 8        //上位ﾊﾞｲﾄを下位ﾊﾞｲﾄへ置き換えて格納
0018     [w:[#MEMLINK]2001]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] & 0xFF      //上位ﾊﾞｲﾄを消去して次ｱﾄﾞﾚｽへ格納
0019     [t:0001]=[t:0001]+1                                                 //ｱﾄﾞﾚｽｵﾌｾｯﾄ +1
0020     [t:0002]=[t:0002]+2                                                 //ｱﾄﾞﾚｽｵﾌｾｯﾄ +2
0021     if([t:0001]==15)                                                    //2ﾜｰﾄﾞへ2ﾊﾞｲﾄずつ格納することを15回繰り返したら抜ける
0022     {
0023         break
0024     }
0025     endif
0026 }
0027 endloop
0028 _ldcopy(databuf2, [w:[#MEMLINK]2000], 30)       //内部ﾒﾓﾘ2000～2030のﾃﾞｰﾀをﾃﾞｰﾀﾊﾞｯﾌｧへ文字列として格納
0029
0030 //文字列”品名 ： ”を付加する
0031 _strset(databuf1, "")                          //ﾃﾞｰﾀﾊﾞｯﾌｧ1を初期化
0032 _strset(databuf1, "品名 ： ")                  //ﾃﾞｰﾀﾊﾞｯﾌｧ1へ文字列を格納
0033 _strcat(databuf1, databuf2)                    //ﾃﾞｰﾀﾊﾞｯﾌｧ1の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ2を結合
0034
0035 //品名に価格の文字列をくっつける
0036 _strcat(databuf1, databuf0)                    //ﾃﾞｰﾀﾊﾞｯﾌｧ1へﾃﾞｰﾀﾊﾞｯﾌｧ0の値を結合
```

動作概要

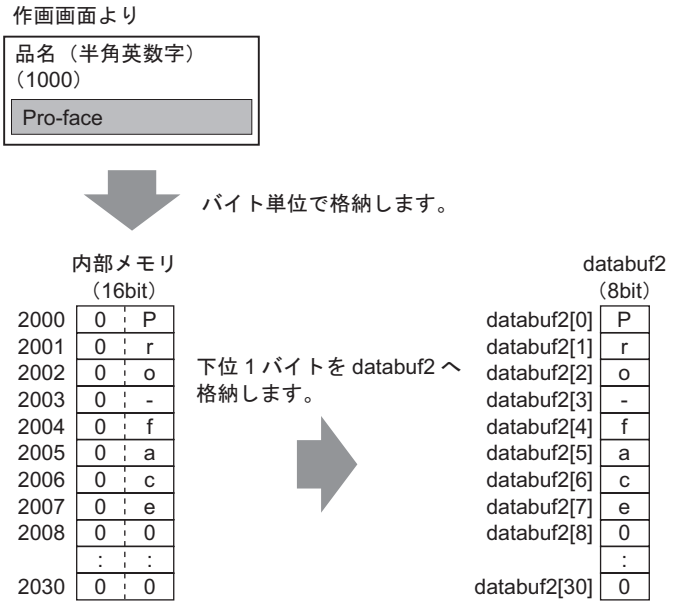
1 内部メモリ 0500 に格納されている価格データへ文字列“価格：”と“円”を付加します。



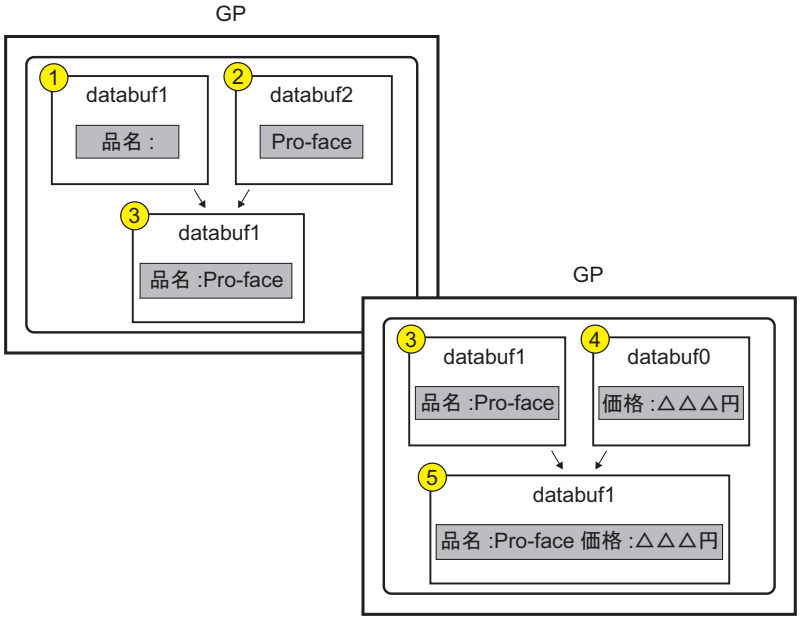
2 印字データをプリンタへ送信するためのデータフォーマットへ変換します。内部メモリ 1000 に連続で格納されている文字列データ（品名）をバイト単位に区切り、下位 1 バイトの文字列データとして内部メモリ 2000 ～ 2030 に格納します。
関数 _ldcopy を使って連続するワードアドレスの下位 1 バイトを順に databuf2 に格納します。

MEMO

- 使用する関数 _ldcopy の動作について、ワード単位で格納されているデータは下位 1 バイトのみバッファに格納し、上位バイトのデータは無視されます。



3 databuf2 へ文字列 “ 品名：” と “ 価格 ” を付加します。



◆Print（ユーザー定義関数）

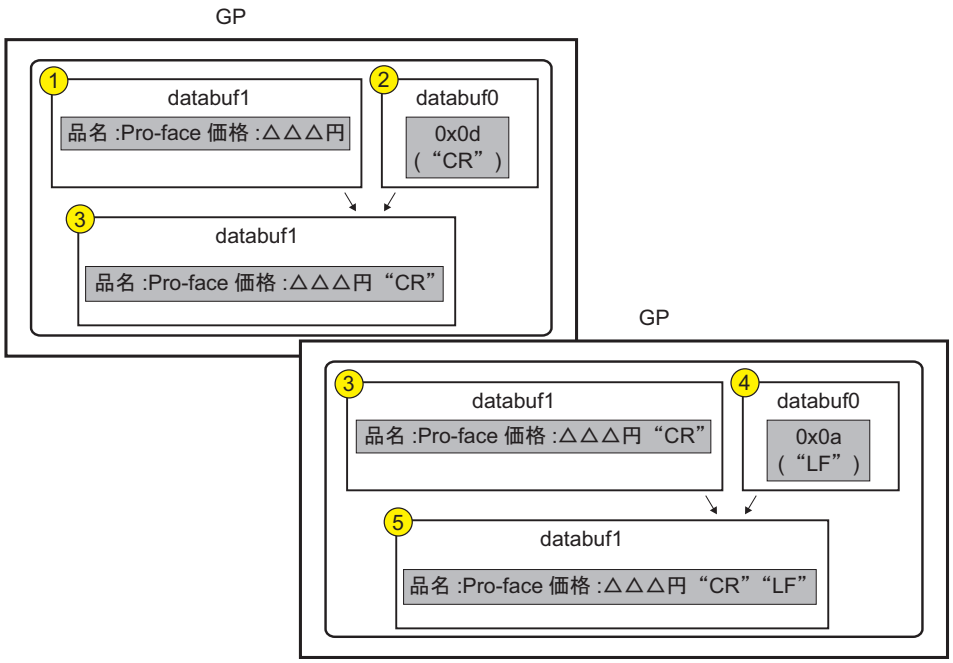
完成スクリプト

実行式　実行式を広く見せる　　アドレス入力

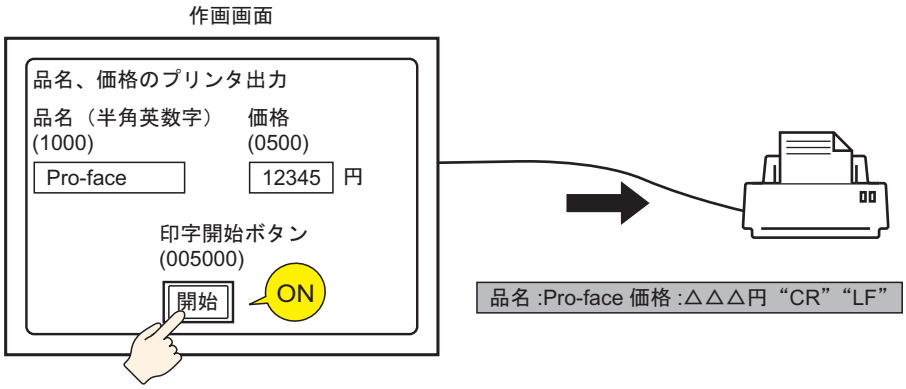
```
0001 Call Strset                          //印字ﾃﾞｰﾀ関数の呼出
0002 _strset(databuf0, "")                //ﾃﾞｰﾀﾊﾞｯﾌｧ1のｸﾘｱ
0003
0004 //印字とﾃﾞﾘﾐﾀ（改行復帰）
0005
0006 _strset(databuf0, 0x0d)              //印字して行の先頭へ戻る
0007 _strcat(databuf1, databuf0)          //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0008 _strset(databuf0, "")                //ﾃﾞｰﾀﾊﾞｯﾌｧ1のｸﾘｱ
0009 _strset(databuf0, 0x0a)              //改行して次の行へ
0010 _strcat(databuf1, databuf0)          //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0011
0012 _strlen([t:0000], databuf1)          //ﾃﾞｰﾀの長さを数値化してﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽへ格納
0013
0014 //ｼﾘｱﾙﾎﾟｰﾄよりﾃﾞｰﾀ送信
0015
0016 IO_WRITE_EX([p:EXT_SIO], databuf1, [t:0000])    //ﾊﾞｯﾌｧ0のﾃﾞｰﾀをﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽの値分送信
0017
```

動作概要

1 連続でプリンタ出力できるように“改行”処理を付加します。



2 印字データをプリンタへ送信します。

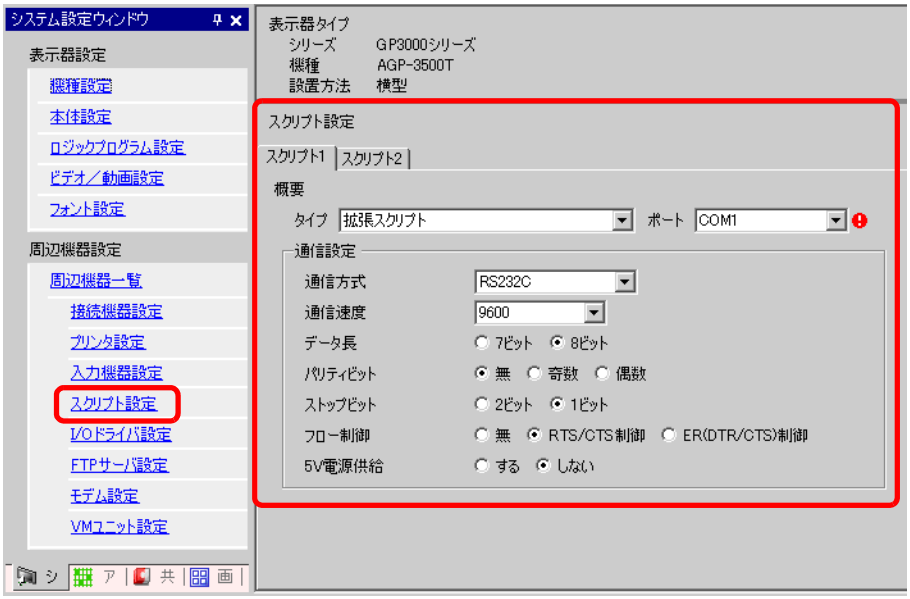


■ 使用する命令

命令	動作概要
If ()	if に続く（）内の条件式が成立時、if（）より後の処理を実行します。
ラベル設定 [r:EXT_SIO_RECV]	その時点の受信しているデータ数（バイト数）がわかります。また、受信データ数は、読み込みのみ有効です。
等しい（==）	N1 == N2（N1=N2）ならば真となります。
文字列設定（_strset）	固定文字列をデータバッファに格納します。
拡張受信 (IO_READ_EX)	指定バイト分のデータを外部機器から受信して、データバッファに格納します。
データバッファから内部デバイスへ (_dlcopy)	データバッファのオフセットから格納されている文字列データを 1 バイトずつ内部デバイスに文字列数分コピーします。
ラベル設定 [c:EXT_SIO_CTRL**]	送信バッファ、受信バッファ、エラーステータスのクリアを行うためのコントロール変数です。
文字列連結（_strcat）	文字列または文字コードを文字列バッファに連結します。
文字列長さ（_strlen）	格納されている文字列の長さを得ます。
拡張送信 (IO_WRITE_EX)	データバッファのデータを送信バイト数分、外部機器から送信します。
代入（=）	左辺に右辺の値を代入します。
加算（+）	ワードデバイスのデータと定数の加算を実行します。
数値 10 進文字列変換 (_bin2decasc)	整数値を 10 進文字列に変換する関数です。
内部デバイスからデータバッファへ (_ldcopy)	内部デバイスに格納されている文字列データを 1 バイトずつデータバッファに文字列数分コピーします。

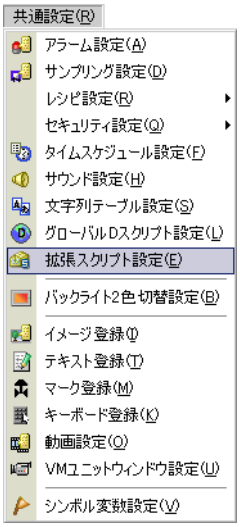
■ 作成手順

1 拡張スクリプトで通信するためにスクリプト設定します。[プロジェクト (F)] メニューから [システム設定 (C)] の [スクリプト設定] をクリックします。[タイプ] は [拡張スクリプト] を必ず指定してください。

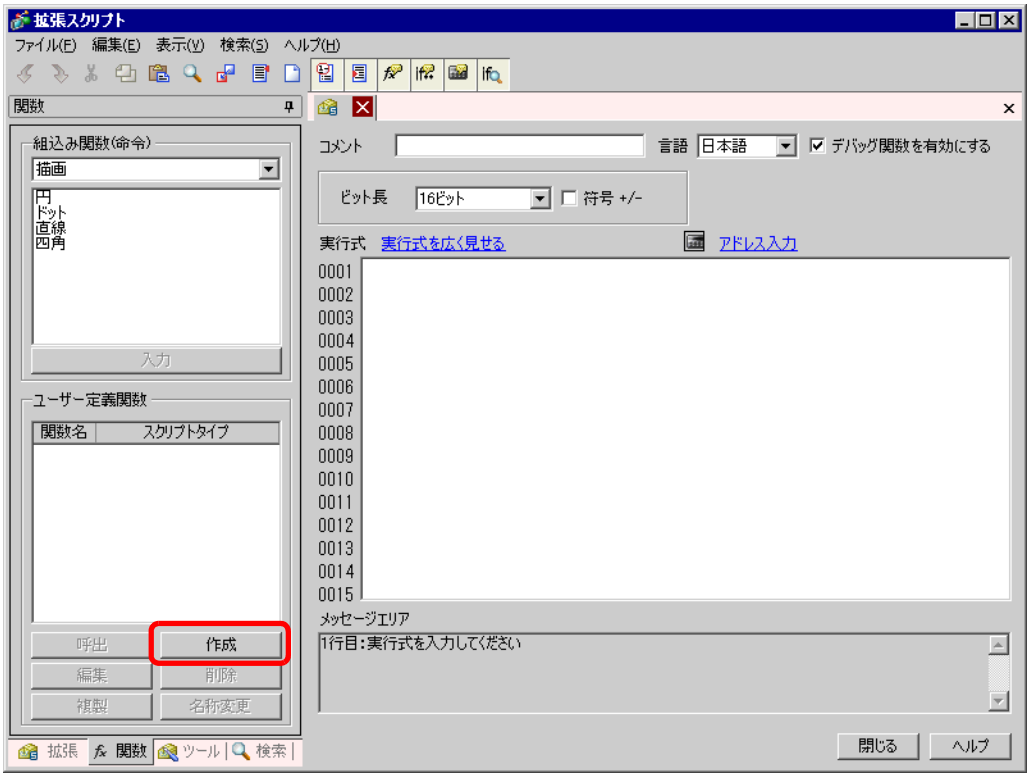


[スクリプト設定] には 2 つタブがあります。上記の例では [スクリプト 1] タブを使用しています。[ポート] は COM1 または COM2 を、[通信設定] の詳細は通信相手の外部機器に合わせて指定してください。

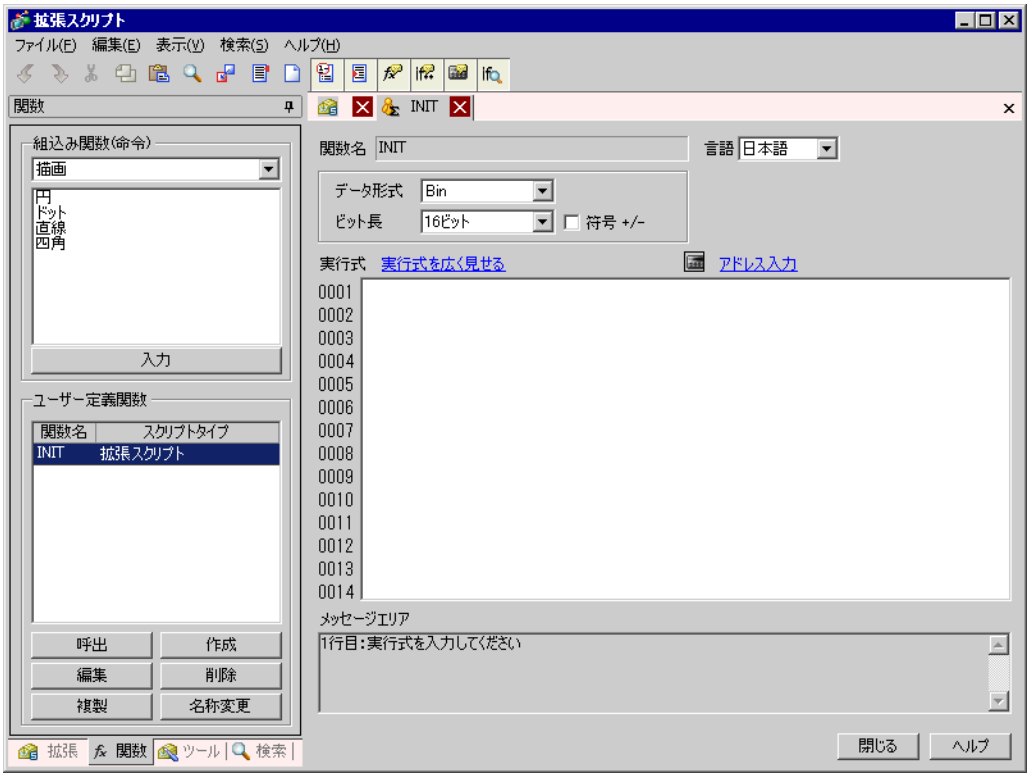
2 [共通設定 (R)] メニューから [拡張スクリプト設定 (E)] をクリックします。



3 “INIT” をユーザー定義関数として登録します。[関数] タブをクリックし、ユーザー定義関数の [作成] をクリックします。



4 関数名 [INIT] を入力し、[OK] をクリックすると以下の画面が表示されます。



5 コマンド、記述式、定数入力から実行式にスクリプトを作成します。

実行式　実行式を広く見せる　アドレス入力

```
0001 [c:EXT_SIO_CTRL00]=1    //送信ﾊﾞｯﾌｧｸﾘｱ
0002 [c:EXT_SIO_CTRL01]=1    //受信ﾊﾞｯﾌｧｸﾘｱ
0003 [c:EXT_SIO_CTRL02]=1    //ｴﾗｰｸﾘｱ
0004
```

6 同様に、“PINIT” をユーザー定義関数として登録します。関数名 [PINIT] を入力し、実行式に以下のスクリプトを作成します。

実行式　実行式を広く見せる　アドレス入力

```
0001 //ﾌﾟﾘﾝﾀ初期化 ( ESC/Pｺﾏﾝﾄﾞ 「ESC + @」 )
0002
0003 _strset(databuf0, "")            //ﾃﾞｰﾀﾊﾞｯﾌｧ0をｸﾘｱ
0004 _strset(databuf0, 0x1B)          //ASCIIｺｰﾄﾞ"ESC"をｾｯﾄ
0005 _strset(databuf1, "")            //ﾃﾞｰﾀﾊﾞｯﾌｧ1をｸﾘｱ
0006 _strset(databuf1, 0x40)          //ASCIIｺｰﾄﾞ"@"をｾｯﾄ
0007 _strcat(databuf0, databuf1)      //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0008 _strlen([t:0000], databuf0)      //ﾃﾞｰﾀの長さを数値化しﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽへ格納
0009
0010 //ｼﾘｱﾙﾎﾟｰﾄよりﾃﾞｰﾀ送信
0011
0012 IO_WRITE_EX([p:EXT_SIO], databuf0, [t:0000])    //ﾊﾞｯﾌｧ0のﾃﾞｰﾀをﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽの値分送信
0013
```

7 同様に、“Strset” をユーザー定義関数として登録します。関数名 [Strset] を入力し、実行式に以下のスクリプトを作成します。

実行式　実行式を広く見せる　アドレス入力

```
0001 //文字列”価格 ： ”と”円”を付加する
0002 _strset(databuf0, "")                          //ﾃﾞｰﾀﾊﾞｯﾌｧ0を初期化
0003 _strset(databuf0, "価格 ： ")                  //ﾃﾞｰﾀﾊﾞｯﾌｧ0へ文字列を格納
0004 _bin2decasc(databuf0, [w:[#MEMLINK]0500])      //数値を文字列に変換してﾃﾞｰﾀﾊﾞｯﾌｧ1へ格納
0005 _strcat(databuf0, databuf1)                    //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0006 _strset(databuf1, "")                          //ﾃﾞｰﾀﾊﾞｯﾌｧ1を初期化
0007 _strset(databuf1, "円")                        //ﾃﾞｰﾀﾊﾞｯﾌｧ1へ文字列を格納
0008 _strcat(databuf0, databuf1)                    //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0009
0010 //ﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽの初期化
0011 [t:0001]=0
0012 [t:0002]=0
0013
0014 //内部ﾒﾓﾘにﾜｰﾄﾞ単位で連続格納されている文字列をﾊﾞｲﾄ単位で格納し直す (30文字分)
0015 loop()
0016 {
0017     [w:[#MEMLINK]2000]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] >> 8        //上位ﾊﾞｲﾄを下位ﾊﾞｲﾄへ置き換えて格納
0018     [w:[#MEMLINK]2001]#[t:0002]=[w:[#MEMLINK]1000]#[t:0001] & 0xFF      //上位ﾊﾞｲﾄを消去して次ｱﾄﾞﾚｽへ格納
0019     [t:0001]=[t:0001]+1                                                 //ｱﾄﾞﾚｽｵﾌｾｯﾄ +1
0020     [t:0002]=[t:0002]+2                                                 //ｱﾄﾞﾚｽｵﾌｾｯﾄ +2
0021     if([t:0001]==15)                                                    //2ﾜｰﾄﾞへ2ﾊﾞｲﾄずつ格納することを15回繰り返したら抜ける
0022     {
0023         break
0024     }
0025     endif
0026 }
0027 endloop
0028 _ldcopy(databuf2, [w:[#MEMLINK]2000], 30)       //内部ﾒﾓﾘ2000～2030のﾃﾞｰﾀをﾃﾞｰﾀﾊﾞｯﾌｧへ文字列として格納
0029
0030 //文字列”品名 ： ”を付加する
0031 _strset(databuf1, "")                          //ﾃﾞｰﾀﾊﾞｯﾌｧ1を初期化
0032 _strset(databuf1, "品名 ： ")                  //ﾃﾞｰﾀﾊﾞｯﾌｧ1へ文字列を格納
0033 _strcat(databuf1, databuf2)                    //ﾃﾞｰﾀﾊﾞｯﾌｧ1の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ2を結合
0034
0035 //品名に価格の文字列をくっつける
0036 _strcat(databuf1, databuf0)                    //ﾃﾞｰﾀﾊﾞｯﾌｧ1へﾃﾞｰﾀﾊﾞｯﾌｧ0の値を結合
```

①

②

③

8 同様に、“Print” をユーザー定義関数として登録します。関数名 [Print] を入力し、実行式に以下のスクリプトを作成します。

実行式 実行式を広く見せる アドレス入力

```
0001 Call Strset                          //印字ﾃﾞｰﾀ関数の呼出
0002 _strset(databuf0, "")                //ﾃﾞｰﾀﾊﾞｯﾌｧ1のｸﾘｱ
0003
0004 //印字とﾃﾞﾘﾐﾀ（改行復帰）
0005
0006 _strset(databuf0, 0x0d)              //印字して行の先頭へ戻る
0007 _strcat(databuf1, databuf0)          //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0008 _strset(databuf0, "")                //ﾃﾞｰﾀﾊﾞｯﾌｧ1のｸﾘｱ
0009 _strset(databuf0, 0x0a)              //改行して次の行へ
0010 _strcat(databuf1, databuf0)          //ﾃﾞｰﾀﾊﾞｯﾌｧ0の後ろにﾃﾞｰﾀﾊﾞｯﾌｧ1を結合
0011
0012 _strlen([t:0000], databuf1)          //ﾃﾞｰﾀの長さを数値化してﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽへ格納
0013
0014 //ｼﾘｱﾙﾎﾟｰﾄよりﾃﾞｰﾀ送信
0015
0016 IO_WRITE_EX([p:EXT_SIO], databuf1, [t:0000])     //ﾊﾞｯﾌｧ0のﾃﾞｰﾀをﾃﾝﾎﾟﾗﾘｱﾄﾞﾚｽの値分送信
0017
```

9 メインのスクリプトを作成します。実行式に以下のスクリプトを作成して完成です。

実行式 実行式を広く見せる アドレス入力

```
0001 //ﾌﾟﾘﾝﾀより1ﾊﾞｲﾄの印刷許可用ﾃﾞｰﾀを受信する
0002 if([r:EXT_SIO_RECV]==1)                                    //受信格納したﾃﾞｰﾀ数が1のとき
0003 {
0004     _strset(databuf0, "")                                  //ﾃﾞｰﾀﾊﾞｯﾌｧ0を初期化
0005     IO_READ_EX([p:EXT_SIO], databuf0, 1)                   //ﾃﾞｰﾀﾊﾞｯﾌｧ0へﾃﾞｰﾀを読み出し
0006     _dlcopy([w:[#MEMLINK]0100], databuf0, 0, 1)            //内部ﾒﾓﾘにﾃﾞｰﾀﾊﾞｯﾌｧ0からの値を格納
0007 }
0008 endif
0009
0010 //印刷許可用ﾃﾞｰﾀから印刷を開始するかどうかを判断する
0011 if([b:[#MEMLINK]005000]==1 and [w:[#MEMLINK]0100]==0x31)    //ﾌﾟﾘﾝﾀ開始ｽｲｯﾁがONかつ関数「ACK」のﾃﾞｰﾀが
0012 {                                                          //1（ASCII）のとき
0013     Call INIT                                              //通信初期化関数の呼出
0014     Call PINIT                                             //ﾌﾟﾘﾝﾀ初期化関数の呼出
0015     Call Print                                             //印字ﾃﾞｰﾀ送信、ﾌﾟﾘﾝﾀ開始関数の呼出
0016     clear([b:[#MEMLINK]005000])                            //ﾌﾟﾘﾝﾀ開始ｽｲｯﾁOFF
0017 }
0018 endif
0019
0020 if([b:[#MEMLINK]005000]==1 and [w:[#MEMLINK]0100]==0x30)    //ﾌﾟﾘﾝﾀ開始ｽｲｯﾁがONかつ関数「ACK」のﾃﾞｰﾀが
0021 {                                                          //0（ASCII）のとき
0022     clear([b:[#MEMLINK]005000])                            //ﾌﾟﾘﾝﾀ開始ｽｲｯﾁOFF
0023 }
0024 endif
```

MEMO

- 手順 3 ～ 9 で作成したユーザー定義関数をメインのスクリプトへ配置させる場合、配置させる関数を選択した状態で [関数] タブの [呼出] をクリックします。「Call 関数名」で配置されます。
- 文字列選択時に［Ctrl］キー＋［Shift］キー＋［→］キー／［←］キーを押すと、テキストブロックの最後まで選択できます。
- ［Ctrl］キー＋［F4］キーを押すと、現在選択している画面を閉じます。
- ［Esc］キーを押すと、スクリプトを上書き保存／破棄して終了します。

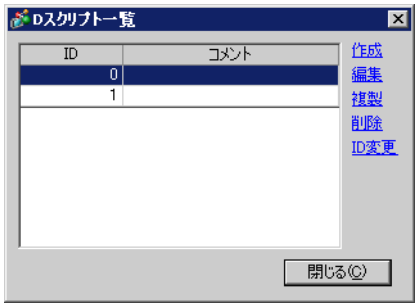
20.6 スクリプト作成の流れ

20.6.1 D スクリプト / グローバル D スクリプト作成の流れ

[部品 (P)] メニューから [D スクリプト (R)] を開きます。

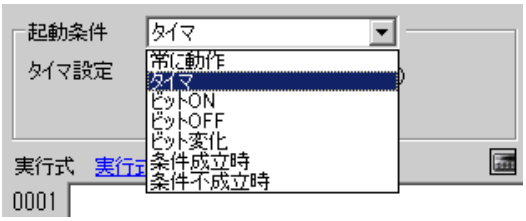
[共通設定 (R)] メニューから [グローバル D スクリプト設定 (L)] を開きます。

[作成] をクリックします。既にスクリプトを登録している場合は、ID 番号を指定して [編集] をクリックするか、ID 番号の行をダブルクリックします。

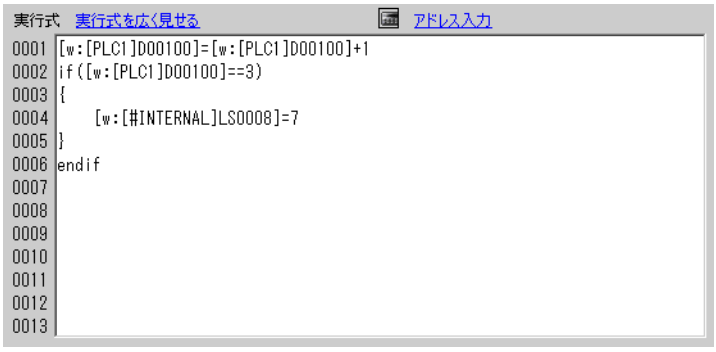


[作成] をクリックします。既にスクリプトを登録している場合は、ID 番号を指定して [編集] をクリックするか、ID 番号の行をダブルクリックします。

スクリプトを実行させる起動条件を設定します。動作の詳細については、「20.7 起動条件のしくみ」（20-42 ページ）を参照してください。

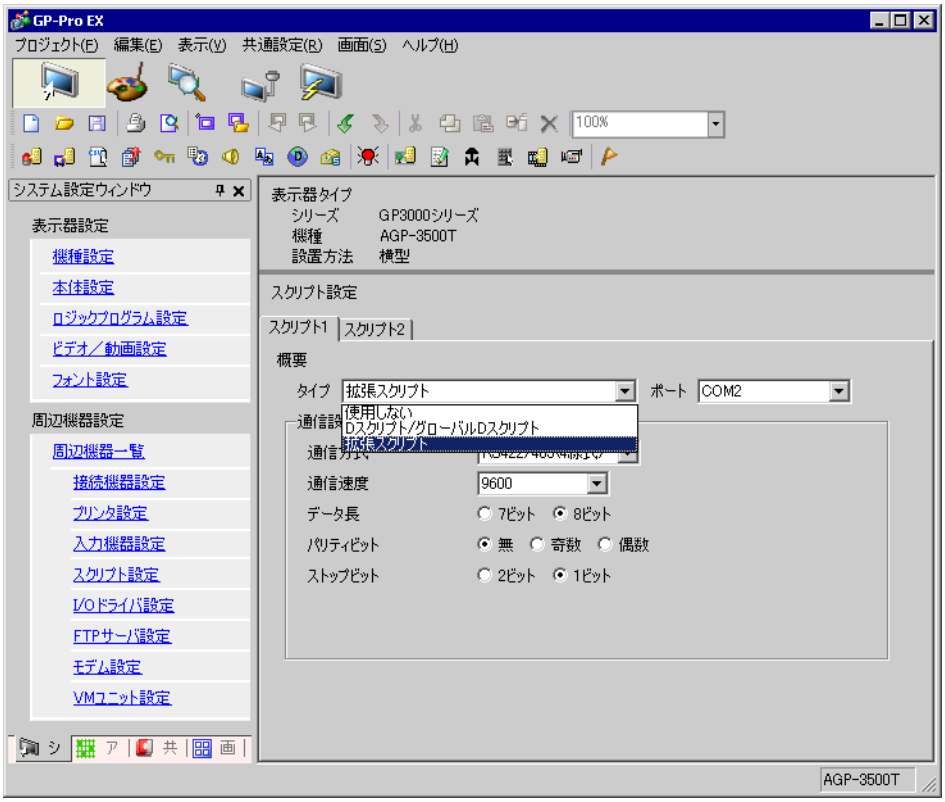


スクリプト（実行式）を作成します。命令や関数の詳細については、「21.13 命令一覧」（21-94 ページ）を参照してください。



20.6.2 拡張スクリプト作成の流れ

[プロジェクト (F)] メニューから [システム設定 (C)] を開きます。[スクリプト設定] をクリックすると、以下のダイアログボックスが表示されます。
拡張スクリプトを使用する場合、[タイプ] から [拡張スクリプト] を選択し、接続するポートを指定します。

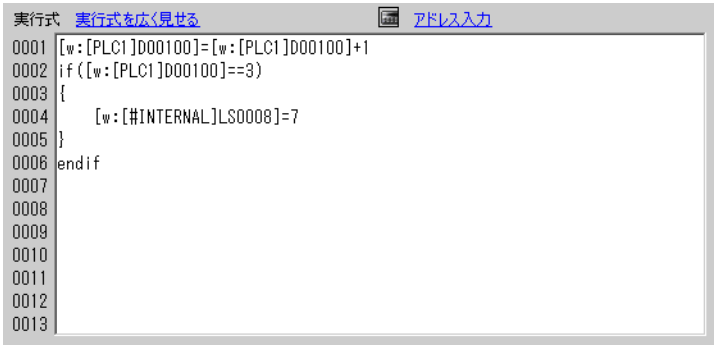


↓

[共通設定 (R)] メニューから [拡張スクリプト設定 (E)] を開きます。

↓

スクリプト（実行式）を作成します。命令や関数の詳細については、「21 プログラム命令、記述式一覧」（21-1 ページ）を参照してください。



20.6.3 ユーザー定義関数の設定の流れ

作成したスクリプトをユーザー定義関数として登録し、他のスクリプトで流用することができます。登録された関数は、D スクリプト、グローバル D スクリプト、拡張スクリプトで利用可能になります。

■ 設定の流れ

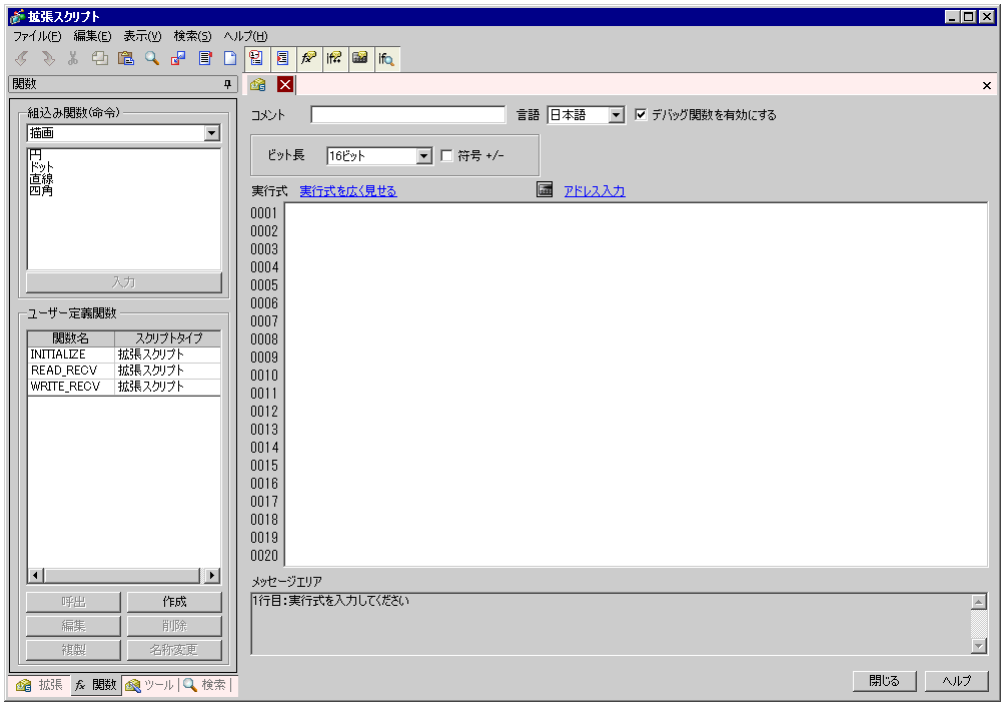
新規にユーザー定義関数を作成する場合

[作成] をクリックすると、ユーザー定義関数を作成するダイアログボックスが表示されます。

既に登録したユーザー定義関数を編集する場合

該当するユーザー定義関数を選択した状態で [編集] をクリックすると、登録したユーザー定義関数が表示されます。

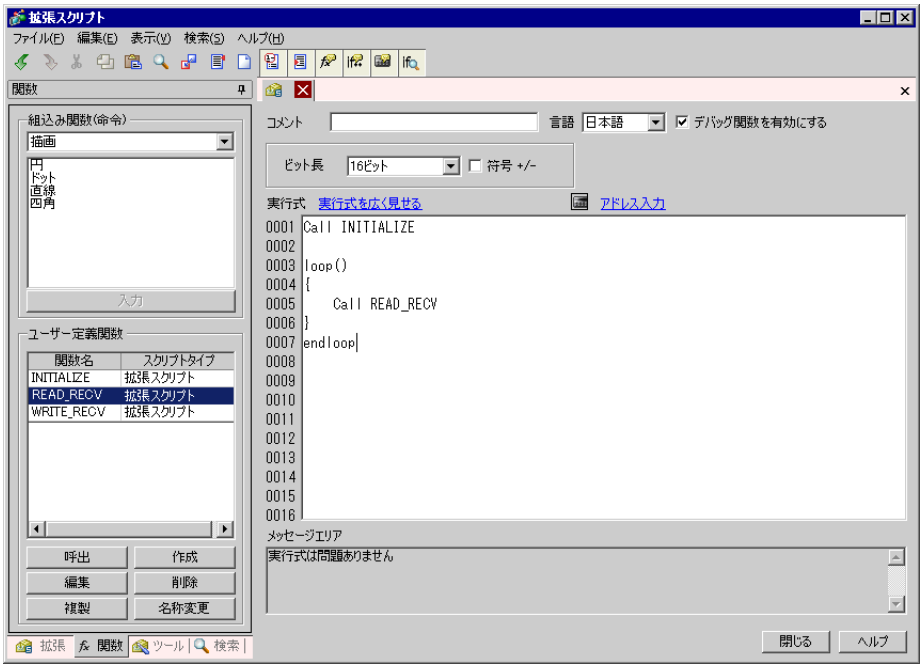
関数名を入力し、登録するスクリプトを実行フィールドに作成します。[OK] をクリックするとユーザー定義関数として登録されます。



MEMO

- 関数名に使用できる名称について制限があります。詳細については、「20.9.3 ユーザー定義関数の制限事項」(20-57 ページ) を参照してください。

呼び出すユーザー定義関数を選択し、[呼出] をクリックすると実行フィールドに「Call 関数名」で配置されます。



重　要

- **ユーザー定義関数を他のスクリプトで流用する場合、拡張スクリプトにて作成した関数に限り D スクリプト、グローバル D スクリプトで使用することはできません。**

20.7 起動条件のしくみ

作成したスクリプトの起動条件は、以下の 7 種類から選択できます。

設定項目		設定内容
常に動作		常にスクリプトを起動します。
タイマ		指定した時間の間隔でスクリプトを起動します。
ビット	**ビット** ON	指定したビットの立ち上がりを検出してスクリプトを起動します。
	ビット OFF	指定したビットの立ち下がりを検出してスクリプトを起動します。
	ビット変化	指定したビットの立ち上がり / 立ち下がりを検出してスクリプトを起動します。
条件式	**条件成立時**	指定した条件式の成立を検出してスクリプトを起動します。
	条件不成立時	指定した条件式の不成立を検出してスクリプトを起動します。

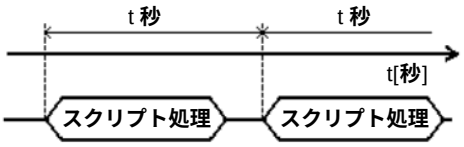
20.7.1 常に動作

表示スキャンタイム毎に処理を実行します。

20.7.2 タイマ

■ タイマ

設定した時間毎に処理を 1 回実行します。1 秒～ 32767 秒の範囲で指定します。



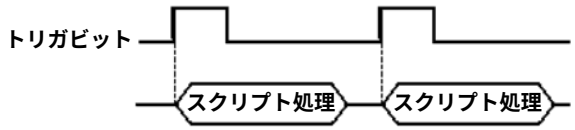
MEMO

- タイマの設定時間は、設定時間 + 表示スキャンタイムの誤差が発生します。また、描画時間やプリントアウトなどによって遅延することがあります。表示スキャンタイムの詳細については、「■ トリガビットの制限事項」（20-45 ページ）を参照してください。
- D スクリプトの場合、画面切替を行うと新たに 0 からカウントします。

20.7.3 ビット

■ ビット ON

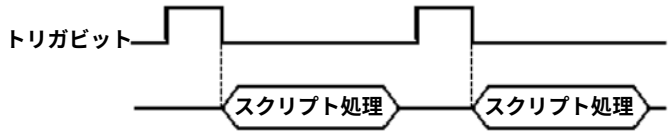
指定したビットアドレス（トリガビット）の立ち上がりを検出して処理を 1 回実行します。



MEMO
- トリガビットの ON/OFF は、通信サイクルタイムもしくは表示スキャンタイムのいずれか長い方の時間以上保持してください。動作の詳細については、「■ トリガビットの制限事項」（20-45 ページ）を参照してください。

■ ビット OFF

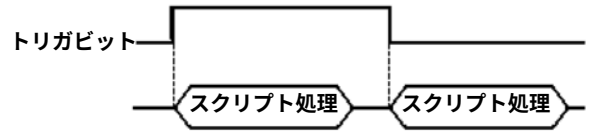
指定したビットアドレス（トリガビット）の立ち下がりを検出して処理を 1 回実行します。



MEMO
- トリガビットの ON/OFF は、通信サイクルタイムもしくは表示スキャンタイムのいずれか長い方の時間以上保持してください。動作の詳細については、「■ トリガビットの制限事項」（20-45 ページ）を参照してください。

■ ビット変化

指定したビットアドレス（トリガビット）の立ち上がり、もしくは立ち下がりを検出して処理を 1 回実行します。



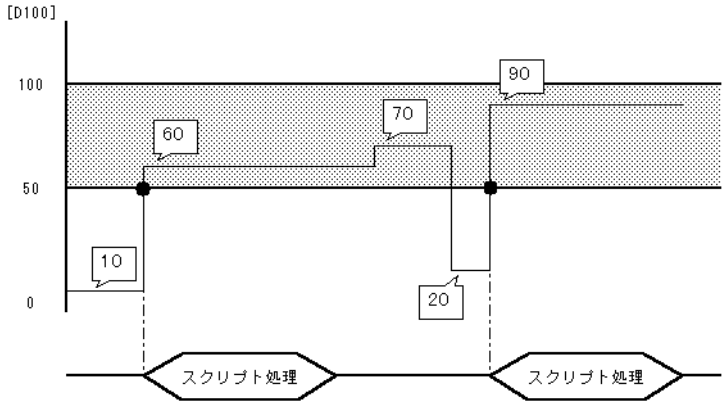
MEMO
- トリガビットの ON/OFF は、通信サイクルタイムもしくは表示スキャンタイムのいずれか長い方の時間以上保持してください。動作の詳細については、「■ トリガビットの制限事項」（20-45 ページ）を参照してください。

20.7.4 条件式

■ 条件成立時

起動条件式で指定した条件式の成立を検出して処理を 1 回実行します。

例　　条件式を 100>[D100]>50 とした場合、スクリプト処理は下図のタイミングで行われます。
[不成立]→[成立] を検出して処理を実行しますので、D100 に 70 が代入された
[成立]→[成立] のタイミングでは実行されません。



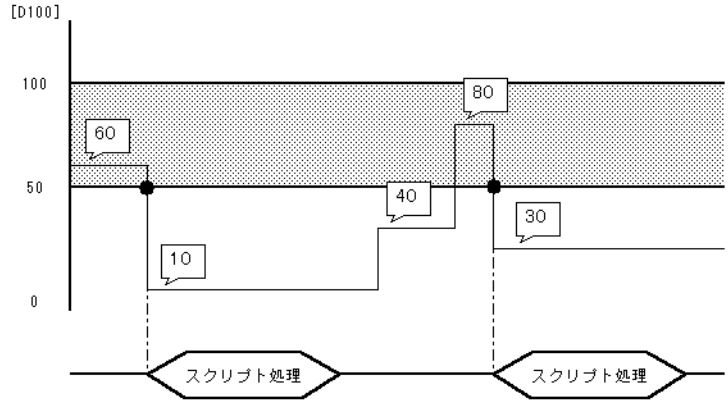
MEMO

- 起動条件は、通信サイクルタイムもしくは表示スキャンタイムのいずれか長い方の時間以上保持してください。動作の詳細については、「■ トリガビットの制限事項」（20-45 ページ）を参照してください。

■ 条件不成立時

起動条件式で指定した条件式の不成立を検出して処理を 1 回実行します。

例　　条件式を 100>[D100]>50 とした場合、スクリプト処理は下図のタイミングで行われます。
[成立]→[不成立] を検出して処理を実行しますので、D100 に 20 が代入された
[不成立]→[不成立] のタイミングでは実行されません。



MEMO

- 起動条件は、通信サイクルタイムもしくは表示スキャンタイムのいずれか長い方の時間以上保持してください。動作の詳細については、「■ トリガビットの制限事項」（20-45 ページ）を参照してください。

■ トリガビットの制限事項

- 接続機器のデバイスへ書き込む場合、起動条件は通信サイクルタイム以上の間隔をあけるようにしてください。GP 内部の特殊リレーの表示スキャンカウンタなどを書き込みのトリガにし、接続機器のデバイスへの書き込みを頻繁に行うと、通信エラーやシステムエラーになる場合があります。
- D スクリプトの起動条件のビットをタッチでセットし、D スクリプトの処理の中でそのビットを OFF する場合には、連続でタッチするとタイミングによってはビットの立ち上がりを検出できない場合があります。

  D スクリプトの起動条件式は、前回読み出した値と今回読み出した値とを比較して、起動条件が成立しているかを判断します。したがって、起動条件式の中で記憶するビットアドレスの値は、実行式の中で変更したとしても直後に反映されず、変更前の値のままとなっています。次のスキャンで変更後の値を読み出します。

  ※通信サイクルタイム：　通信サイクルタイムとは、GP から接続機器にデータを要求して取り込むまでの時間です。内部デバイスの LS2037 にバイナリデータで格納されます。単位は ms です。±10ms の誤差があります。

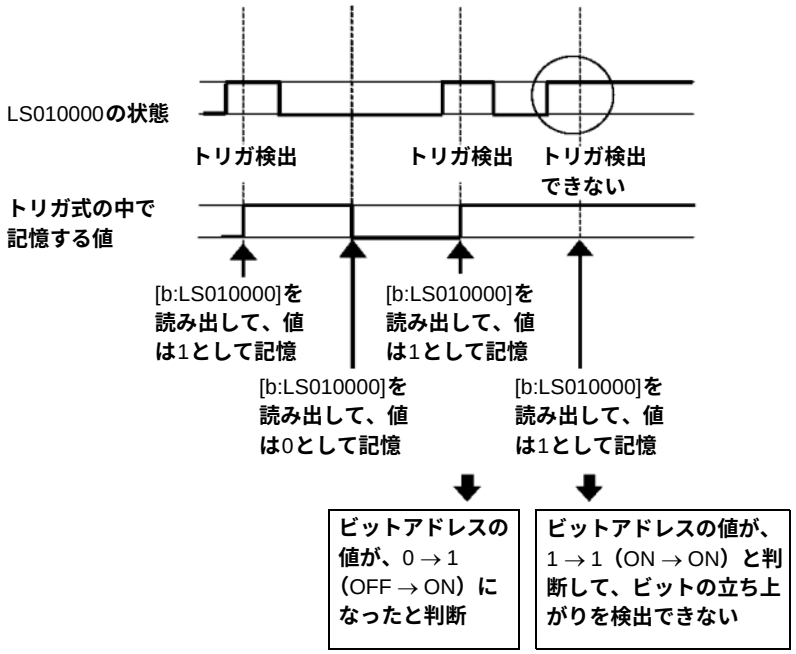
  表示スキャンタイム：　表示スキャンタイムとは、1 画面の表示・演算処理にかかる時間です。内部デバイスの LS2036 にバイナリデータで格納されます。単位は ms です。±10ms の誤差があります。

  例）タッチからトリガビット (LS010000) を ON し、D スクリプト内で OFF する場合

  起動条件：ビット ON[#INTERNAL] LS010000

  実行式：　clear([b:[#INTERNAL] LS010000])

◆D スクリプト処理　タイミングチャート



例のような D スクリプトをタッチのタイミングに依存せずに検出する方法として、次のように記述してください。

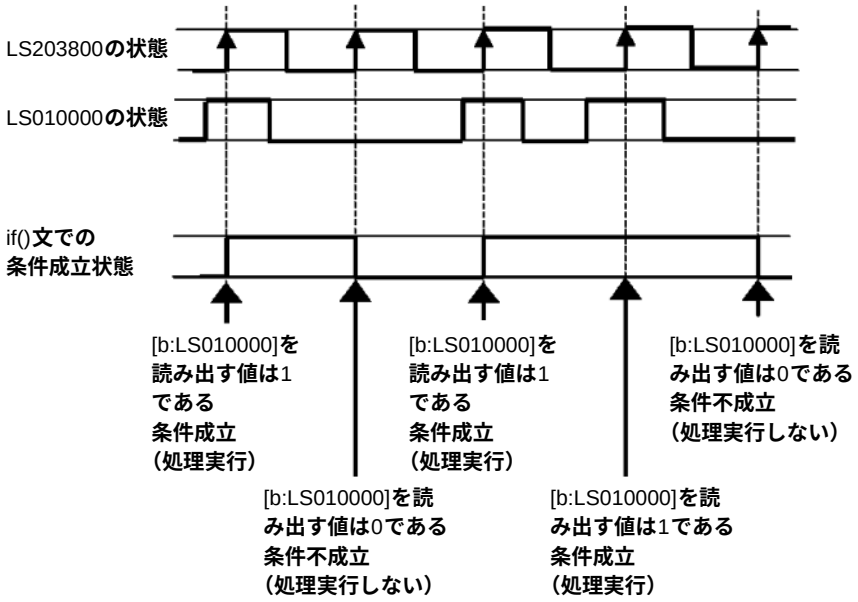
if() 文を用いて起動条件を検出
タッチでセットするビットを if 文で判断するようにします。if() 文では、実行するたびに値を読み出して比較チェックをしています。

起動条件：ビット ON[#INTERNAL]LS203800 ※1])
実行式：if([b:[#INTERNAL]LS010000]==1)
{
clear([b:[#INTERNAL]LS010000])
　　:
　　:

※1　GP 内部のカウンタです。表示画面に設定されている部品処理がひととおり完了するたびにカウントアップします。

上記のような D スクリプトの場合、連続してタッチ入力が行われても次項のタイミングチャートのように表示スキャン毎に値を読み出して条件が一致するかを判断するため、前回の値とは関係なく条件が一致すれば実行されます。

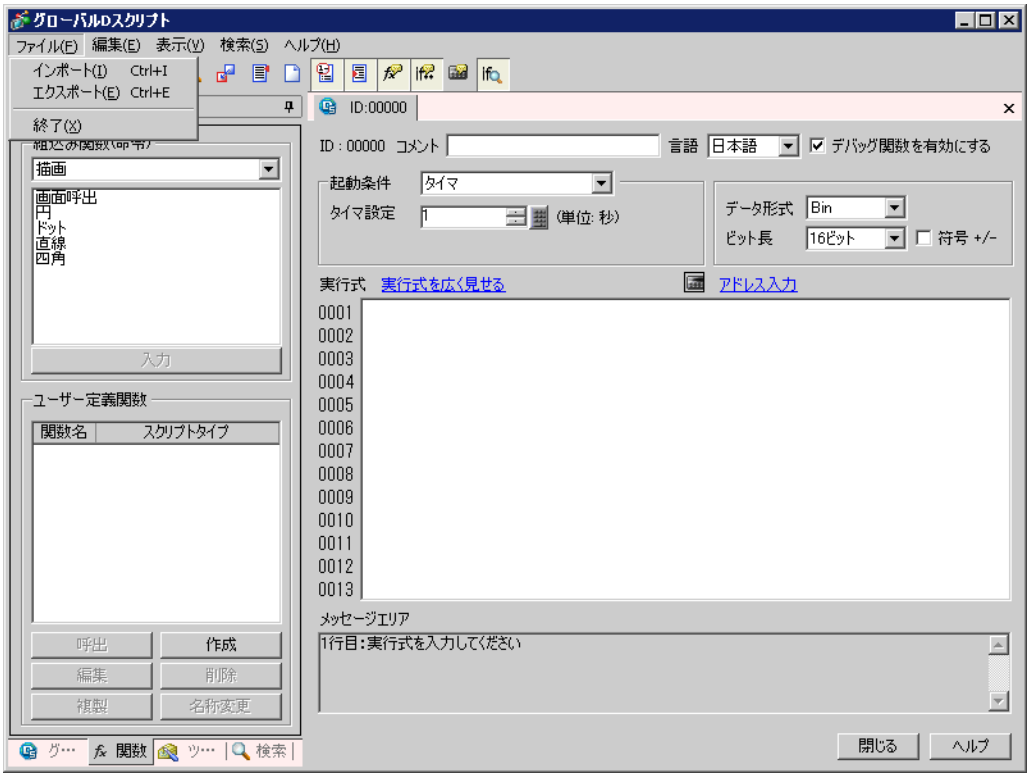
◆D スクリプト処理　タイミングチャート

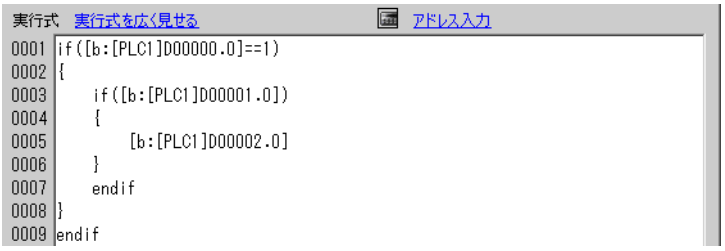


20.8 設定ガイド

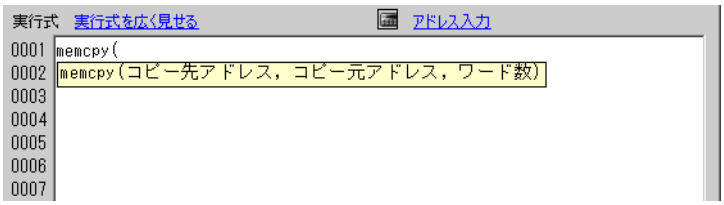
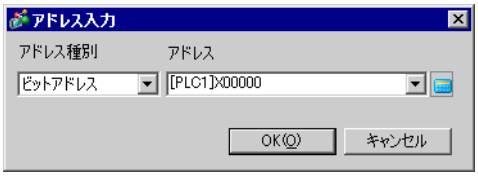
20.8.1 D スクリプト / 共通設定 [グローバル D スクリプト設定] の設定ガイド

以下は共通設定 [グローバル D スクリプト設定] のダイアログボックスです。D スクリプトの設定項目もこのダイアログボックスの内容と同様です。また共通設定 [拡張スクリプト設定] は、ID やトリガの設定項目がありませんが、その他の設定項目は以下同様です。

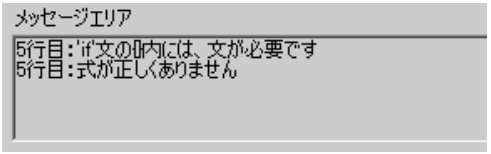


設定項目	設定内容
エクスポート	ファイルメニューから選択できます。エクスポートは、作成したスクリプトをテキストファイル（.txt）で書き出し、他のスクリプトへ流用（インポート）することができます。
インポート	ファイルメニューから選択できます。インポートは、書き出されたスクリプト（テキストファイル）を取り込むことができます。
行番号 	実行式右側の行番号を表示します。
自動字下げ調節 	下図のように自動的に字下げを調節します。 

次のページに続きます。

設定項目	設定内容
関数入力補助	下図のように関数 + (　を入力すると、その関数の書式が表示されます。 
自動構文補完	キーボードから“if”もしくは“loop”と入力した際、それに続く構文が自動で配置されます。
アドレス入力	スクリプト作成時、キーボードからアドレスの左側の括弧（[）を入力すると、自動で [アドレス入力] ダイアログボックスを表示します。  アドレス種別は [ビットアドレス]、[ワードアドレス]、[テンポラリアドレス] から選択します。 • **ビットアドレス** 接続機器アドレスや GP 内部デバイス、ビット変数が指定できます。 • **ワードアドレス** 接続機器アドレスや GP 内部デバイス、整数変数が指定できます。 • **テンポラリアドレス** スクリプトでのみ使用できるアドレスです。 内部デバイスの詳細については、以下を参照してください。 ☞**「付録 1.2 負荷をかけずに接続機器（PLC など）と通信したい（ダイレクトアクセス方式）」(A-3 ページ)** ☞**「付録 1.3 対応していない接続機器と通信したい（メモリリンク方式）」(A-5 ページ)** **重　要** • **スクリプトでは、暗証番号などの用途で 0 から始まる数値を設定しないでください。0 から始まる数値はすべて Oct（8 進数）データとして処理されます。** • **入力データ形式別データ記載方法** **（例）** DEC（10 進数）：**最初が 0 でない数値** （例）100 Hex（16 進数）：**最初が 0x で始まる数値** （例）0x100 Oct（8 進数）：**最初が 0 で始まる数値** （例）0100 • **演算子 AND を用いたデータ形式の異なる演算例（Hex と BCD の場合）** Hex **のみの場合** 0x270F & 0xFF00　**結果：**0x2700 BCD **と** Hex **の場合** 9999 & 0xFF00　**結果：**0x9900

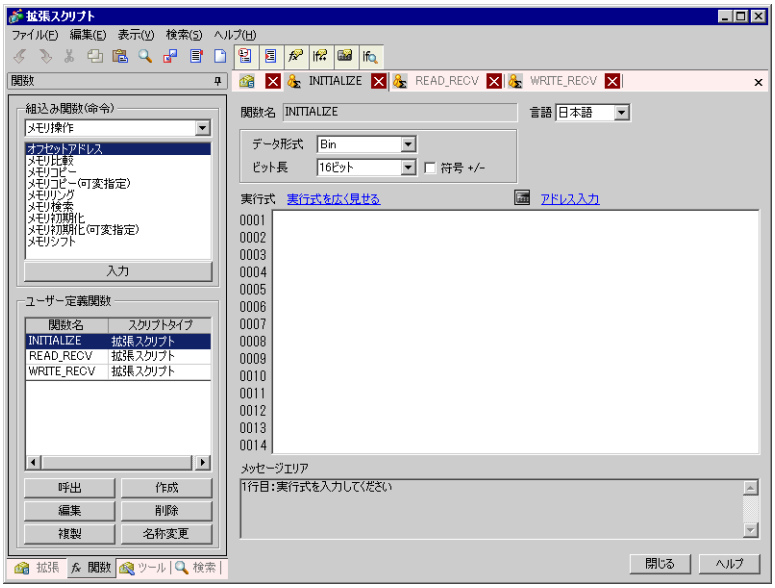
次のページに続きます。

設定項目	設定内容
自動構文解析 	スクリプト作成時に文法のチェックを行います。チェック結果は随時、ダイアログボックス下部のウィンドウに表示されます。 
ID	スクリプトは ID 番号で管理しています。 起動条件が異なる複数のスクリプトを作成する場合、0 ～ 65535 の範囲で指定します。
コメント	作成するスクリプトのコメントを入力します。
言語	日本語、欧米、中国語（繁体字）、中国語（簡体字）、韓国語から指定します。
デバッグ関数を有効にする	デバッグ関数を有効にするかどうかを指定します。_debug 関数がスクリプト本文中に存在する場合、_debug 関数を実行します。 動作の詳細については、「21.7.1 デバッグ関数」（21-61 ページ）を参照してください。
起動条件	スクリプトを実行させる起動条件を設定します。動作の詳細については、「20.7 起動条件のしくみ」（20-42 ページ）を参照してください。 拡張スクリプトでは、起動条件の設定項目はありません。
データ形式	スクリプトで扱うデータ形式を Bin か BCD で指定します。 拡張スクリプトでは、Bin 固定になります。
ビット長	スクリプトで扱うデータ長を 16 ビットか 32 ビットで指定します。
符号 +/–	マイナス数値を入れたい場合、選択します。 データ形式が Bin のときのみ設定できます。
実行式	スクリプトを記述します。
組み込み関数（命令）	スクリプトで使用できる命令や関数をアイコンなどから簡単に配置させることができるため、入力する手間を省くことができます。 使用できる命令や関数の詳細について ☞**「21.13 命令一覧」（21-94 ページ）** ＜組み込み関数＞  **上部のプルダウンメニューから分類を選択すると、下部に関連する関数が表示されます。 関数を選択した状態で[入力]をクリックすると、各種設定ダイアログボックスが表示されます**

次のページに続きます。

設定項目	設定内容
ユーザー定義関数	作成したスクリプトをユーザー定義関数として登録し、他のスクリプトで流用することができます。  **MEMO** • ユーザー定義関数についての詳細は「20.8.2 ユーザー定義関数の設定ガイド」(20-52 ページ）を参照してください。
ツールボックス	スクリプトで使用できる命令をクリックするだけで簡単に配置させることができるため、入力する手間を省くことができます。 また、スクリプトで使用している文字列の検索や置換なども行うことができます。 使用できる命令の詳細については、「21 章　プログラム命令、記述式一覧」(21-1 ページ）を参照してください。 

20.8.2 ユーザー定義関数の設定ガイド



設定項目	設定内容
呼出	作成した関数を呼び出します。呼び出す関数を選択し、[呼出] をクリックすると、実行フィールドに「Call 関数名」で配置されます。
作成	関数を新規に作成します。[作成] をクリックすると、関数名を作成するダイアログボックスが表示されます。
編集	作成した関数を編集します。編集する関数を選択し、[編集] をクリックすると、[D スクリプト関数] ダイアログボックスが表示されます。
削除	作成した関数を削除します。削除する関数を選択し、[削除] をクリックします。
複製	作成した関数をコピーします。コピーする関数を選択し [複製] をクリックすると、複製先の関数名を作成するダイアログボックスが表示されます。
名称変更	作成した関数の名称を変更します。[名称変更] をクリックすると、関数名を変更するダイアログボックスが表示されます。

20.9 制限事項

20.9.1 D スクリプト / グローバル D スクリプトの制限事項

- D スクリプトの設定数の目安として設定アドレス 3 個で部品 1 個と同じ容量となります。なお、1 つの D スクリプトに設定できる最大アドレス数は 255 となります。ただしデバイス数が多くなるとレスポンスが遅くなりますので最小限のデバイスで記述するようにしてください。
- プロジェクトマネージャ／ユーティリティのアドレス一括変換で D スクリプトが使用しているアドレスは変換できません。D スクリプトのプログラム上での変更が必要です。
- プロジェクトマネージャ／プロジェクトの名前を付けて保存で接続機器を変更した場合、D スクリプトで使用しているアドレスは変換できません。必ず D スクリプトの設定ダイアログボックス上での変更が必要です。
- D スクリプトでは浮動小数点（フロート変数・リアル変数）の演算はできません。
  また構造体変数指定もできません。ただし構造体の各メンバは指定できます。
- D スクリプトのプログラム量は表示スキャンタイムに影響します。特にアドレスを多数使用するとパフォーマンスが非常に低下しますのでご注意ください。
- 接続機器アドレスへの書き込みを行うスクリプトでは、起動条件で [常に動作] を指定しないでください。書き込み命令が大量に発生するため通信処理が間に合わず、エラーが発生します。[常に動作] を使用する場合は、GP 内部デバイスやテンポラリアドレスを使用してください。
- 関数から関数の呼び出しは最大 9 階層です。それ以上は設定しないでください。
- 関数の再帰呼び出しは最大 9 ネストまでです。
- 関数最大作成数は 254 個です。

◆トリガ条件に指定したデバイスによって、画面切り替え直後または電源投入直後の D スクリプトの実行詳細は次の様になります。

トリガ条件	接続機器デバイスまたは GP 内部デバイス (LS/USR)				メモリリンク			
現在値または現在の条件	ビット値「0」	ビット値「1」	条件不成立	条件成立	ビット値「0」	ビット値「1」	条件不成立	条件成立
ビット立ち上がり	×	○	−	−	×	×	−	−
ビット立ち下がり	○	×	−	−	×	×	−	−
ビット変化	○	○	−	−	×	×	−	−
タイマ設定	×	×	×	×	×	×	×	×
条件式成立時検出	−	−	×	○	−	−	×	○
条件式不成立時検出	−	−	○	×	−	−	○	×

○：画面切替直後または電源投入直後に処理を実行します。
×：画面切替直後または電源投入直後に処理を実行しません。

- タイマ設定時は画面切替時よりタイマのカウントを始めます。
- グローバル D スクリプトでは、電源投入時に上記表の動作を行います。画面切り替え時は上記表は適用されず、トリガ条件を継続して監視します。
- グローバル D スクリプトでのタイマ設定時は、電源投入時よりタイマのカウントを始めます。

MEMO
- タッチキー入力をトリガモードの起動やプログラムでの起動ビット操作に用いないで下さい。タッチ入力のタイミングによって取りこぼす場合があります。

◆D スクリプトの実行文の途中で画面切り替えのアドレスに値を代入する場合、その D スクリプトの処理がすべて終わった後に、画面切り替えの処理が行われます。

（例）

```
ID              00000
データ形式      Bin          データ長      16 ビット      符号 + / –      無し
トリガ          ビット立ち上がり（[b:M0000]）
[w:[PLC1]D0100]=0              // ①
[w:[#INTERNAL]LS0008]=30       // ② ベース画面 30 に切り替え
[w:[PLC1]D0101]=1              // ③
[w:[PLC1]D0102]=2              // ④
```

上記の D スクリプトが実行された場合、③、④が処理された後、画面切り替えの処理が行われます。

◆D スクリプトで使用するデータを GP からタッチにより設定する場合、全てのデータが書き込めたことを検出した上で、D スクリプトを動作させるようにしてください。

◆グローバル D スクリプト特有の制限事項

- 電源投入時に前項の表の動作を行います。画面切替時は前項の表は適用されず、トリガ条件を継続して監視します。
- グローバル D スクリプトは画面切替中などの場合は処理を中断しています。
- 電源投入後、初期画面においては１度全ての読み出しが終了するまで処理は実行されません。
  但し、画面切替を行うと読み出しが終了する前でも処理は実行されます。
- グローバル D スクリプト内の全てのデバイス合計は最大 255 デバイスです。デバイス数が 256 以上になった D スクリプトは動作しません。これらのデバイスは画面に関係なく常時読み出しを行いますので使用時は必要最小限の設定を行ってください。パフォーマンスを低下させる原因となります。
- グローバル D スクリプトの総数は最大 32 個までです。使用している関数も 1 個とカウントします。32 個を超えると超えた分は無視されます。

◆SIO ポート操作の制限事項

- 送受信関数では、アドレスを指定していますが、D スクリプトのアドレス数のカウントには加算されません。
- コントロール変数（書き込みのみ有効）、ステータス変数（読み込みのみ有効）、受信データ変数（読み込みのみ有効）となります。コントロール変数を読み出したり、ステータス変数に書き込みを行うと誤動作するためご注意下さい。
- 送信と受信については、それぞれ別の D スクリプト（あるいは関数）を作成し、実行するようにして下さい。
  転送のフローチャートについて
  ☞「■ フローチャート」（20-24 ページ）
- 送受信関数でデータを格納できる内部デバイスの有効範囲はユーザーエリア（LS20 ～ LS2031、LS2096 ～ LS8191）です。

- [システム設定] の [スクリプト設定] で [D スクリプト / グローバル D スクリプト] を設定していない場合に、D スクリプト / グローバル D スクリプトの [SIO ポート操作] のラベル設定（送信関数、受信関数、コントロール、ステータスの読み出し、受信データ数の読み出し）を実行すると、LS2032 の 13 ビット目が ON します。
  特殊リレーについて
  ☞「付録 1.4.3 特殊リレー」（A-15 ページ）
- 送信関数、受信関数を使用する場合は、D スクリプトのビット長を 16 ビットに設定してください。ビット長を 32 ビットに設定すると、誤作動するためご注意下さい。
- 送信バッファは 2048 バイト、受信バッファは 8192 バイトあります。受信バッファサイズの 80% 以上のデータを受信すると、ER 信号（出力）、RS 信号（出力）が OFF になります。

◆BCD 設定時の制限事項

演算中に BCD に変換できないデータ（Hex の A ～ F）がある場合は、実行を中止します。
A ～ F のデータを扱わないようにしてください。
本原因で実行を中止した場合 GP 内の共通リレー情報（LS2032）の 7 ビット目が ON します。本ビットは電源を OFF にするかオフラインになるまで保持します。
例）

[w:[PLC1]D0200]=([w:[PLC1]D0300]<<2)+80

D300 が 3 の場合、2 ビット左にシフトすると 0x000C となり BCD として扱うことができなくなり、プログラムの実行を中断します。

[w:[PLC1]D0200]=[w:[PLC1]D0300]<<2

D300 が 3 の場合、2 ビット左にシフトすると 0x000C となりますが、演算を終了した結果なので 0x000C を格納して中断されません。

◆ゼロ割算に関する制限事項

演算子の割算 / ・剰余算 % において 0（ゼロ）で割る場合は、実行を中止します。ゼロで割らないようにしてください。
本原因で実行を中止した場合 GP 内の共通リレー情報（LS2032）の 8 ビット目が ON します。本ビットは電源を OFF にするかオフラインになるまで保持します。

◆代入の遅延に関する注意事項

代入にデバイスアドレスを使用する場合、GP と接続機器は通信を行っているため、書き込みが遅延します。以下のような注意が必要です。
例）

[w:[PLC1]D0200]=[w:[PLC1]D0300]+1・・・①
[w:[PLC1]D0201]=[w:[PLC1]D0200]+1・・・②

①の命令文で D0200 に（D0300+1）を代入しますが通信を行っているため時間がかかり、②の命令文では D0200 には①の演算結果はまだ代入されていません。このような場合は①の演算結果を 1 度 LS エリアもしくはテンポラリワークアドレスに格納して実行するようプログラミングしてください。

[w:[#INTERNAL]LS0100]=[w:[PLC1]D0300]+1
[w:[PLC1]D0200]=[w:[#INTERNAL]LS0100]
[w:[PLC1]D0201]=[w:[#INTERNAL]LS0100]+1

20.9.2 拡張スクリプトの制限事項

- 使用できるデバイスアドレスは、LS エリアと USR エリア（拡張ユーザエリア）のみです。
- D スクリプト、グローバル D スクリプトのテンポラリアドレスと拡張スクリプトのテンポラリアドレスは別管理となります。このため、D スクリプトのテンポラリアドレスの内容を変更しても、拡張スクリプトのテンポラリアドレスには反映されません。
- D スクリプト / グローバル D スクリプトで作成したユーザー定義関数を Call することはできますが、関数中で内部デバイス以外のデバイスアドレスをアクセスした場合には、正常に動作しない場合があります。また、ユーザー定義関数は、転送時（GP 用のデータ生成時）に、D スクリプト / グローバル D スクリプト / 拡張スクリプトと別々に生成されます。
- 関数から関数の呼び出しは最大 9 階層です。
- 関数呼び出しは、254 個です。（Call で使用できる関数の数は 254 個です。）
- 拡張スクリプトは部品数のカウントには影響しません。
- 拡張スクリプトのみ対応の関数（文字列操作関数など）を D スクリプトおよびグローバル D スクリプトで呼び出しても動作しません。
- データ形式は、Bin のみです。BCD は設定できません。
- 送信バッファは 2048 バイト、受信バッファは 8192 バイトあります。受信バッファサイズの 80% 以上のデータを受信すると、ER 信号（出力）、RS 信号（出力）が OFF になります。
- 同時に汎用プロトコル、拡張スクリプトを選択することはできません。下表の組み合わせに注意してください。

拡張 SIO 設定	D スクリプトまたはグローバル D スクリプト用の拡張 SIO 関数	拡張スクリプト用の拡張 SIO 関数
汎用プロトコル	○動作可能です	×動作しません
拡張スクリプト	×動作しません	○動作可能です

- 文字列設定の表記について

_strset() 命令などで文字列を使用する場合、文字列をダブルクォーテーション (") で囲む表記となります。この文字列にダブルクォーテーション (") 自身を表したい場合は、\ 記号を付加して「\"」という表記にします。\ 記号単独を表記する方法はありませんので、文字コード形式の設定 (_strset (databuf0,92)) などを利用してください。

例）

"ABC\"DEF" → ABC"DEF
"ABC\DEF" → ABC\DEF
"ABC\\"DEF"→ ABC\"DEF
"ABC\\DEF" → ABC\\DEF

◆拡張 SIO の専用バッファ databuf0、databuf1、databuf2、databuf3 のサイズは以下のようになります。

バッファ	バッファ名称	サイズ
データバッファ 0	databuf0	1K バイト
データバッファ 1	databuf1	1K バイト
データバッファ 2	databuf2	1K バイト
データバッファ 3	databuf3	1K バイト

20.9.3 ユーザー定義関数の制限事項

- 各スクリプトで使用できる命令が一部異なります。流用する場合には、「21.13 命令一覧」（21-94 ページ）を確認してください。
- 関数名に使用できる文字は、半角英数字および ”_” です。（但し、関数名の先頭は半角アルファベットのみです。）
- 以下の関数名は使用しないでください。

and	b_call	Bcall	_bin2hexasc	break	Call
_CF_delete	_CF_dir	_CF_read	_CF_read_csv	_CF_rename	_CF_write
clear	databuf0	databuf1	databuf2	databuf3	_decasc2bin
_dlcopy	dsp_arc	dsp_circle	dsp_dot	dsp_line	dsp_rectangle
else	endif	fall	_hexasc2bin	if	IO_READ
IO_READ_EX	IO_READ_WAIT	IO_WRITE	IO_WRITE_EX	loop	_memcmp
memcpy	_memcpy_EX	memring	_memsearch	memset	_memset_EX
_memshift	not	or	return	rise	rise_expr
set	_strcat	_strlen	_strmid	_strset	timer
toggle	_wait				

20.9.4 演算結果の注意事項

■ 演算結果の桁あふれ例

演算結果が桁あふれをした場合はあふれた値は切り捨てられます。

16 ビット、符号なしの場合

- 65535 + 1 = 0（桁あふれあり）
- (65534 * 2) / 2 = 32766（桁あふれあり）
- (65534 / 2) * 2 = 65534（桁あふれなし）

■ 剰余算の演算結果の違い例

剰余算の場合右辺と左辺の符号により演算結果が違います。

- –9 % 5 = –4
- 9 % –5 = 4

■ 小数点切り捨て例

割算結果の小数点は切り捨てられます。

- 10 / 3 * 3 = 9
- 10 * 3 / 3 = 10

■ BCD 設定時の演算についての注意事項

BCD の演算時、演算結果がビット長をオーバーフローする場合は正しい結果が得られません。

20.9.5 エラーについて

スクリプトの設定の誤りによるエラーメッセージは以下のようになります。GP の画面下部にエラー表示されます。

同様に LS91XX 番台にもエラーコードを書き込みます。エラーコード領域に書き込む番号は、下表中の RAAA の後ろに付加されている数字です。(例えば RAAA130 のエラーが発生した場合、130 を書きこみます。)

スクリプトエラーコード一覧表

D **スクリプト** (**書込みアドレス** =LS9120)	**グローバル** D **スクリプト** (**書込みアドレス** =LS9110)	**拡張スクリプト** (**書込みアドレス** =LS9100)
–	RAAA130	RAAA140
未使用	最大数 32 個をオーバーしています (グローバル D スクリプト)	関数の最大数 255 個をオーバーしています (拡張スクリプト)
–	RAAA131	–
未使用	デバイス合計が最大数 255 個をオーバーしています (グローバル D スクリプト)	未使用
RAAA120	RAAA132	RAAA141
指定した関数が存在しない、または関数内にエラーがあります (D **スクリプト**)	指定した関数が存在しないか関数内にエラーがあります (グローバル D スクリプト)	指定した関数が存在しないか関数内にエラーがあります (拡張スクリプト)
RAAA121	RAAA133	RAAA142
関数のネストが 10 段階以上になっています (D スクリプト)	関数のネストが 10 段階以上になっています (グローバル D スクリプト)	関数のネストが 10 段階以上になっています (拡張スクリプト)
RAAA122	RAAA134	RAAA143
このバージョンのシステムでは実行できない未対応のスクリプトが記述されています (D スクリプト)	このバージョンのシステムでは実行できない未対応のスクリプトが記述されています (グローバル D スクリプト)	このバージョンのシステムでは実行できない未対応のスクリプトが記述されています (拡張スクリプト)
RAAA123	RAAA135	RAAA144
接続機器の設定が未設定の状態で SIO 操作関数が使用されています (D スクリプト)	接続機器の設定が未設定の状態で SIO 操作関数が使用されています (グローバル D スクリプト)	接続機器の設定が未設定の状態で SIO 操作関数が使用されています (拡張スクリプト)
RAAA124	RAAA136	RAAA145
D スクリプト内にエラーがあります	グローバル D スクリプト内にエラーがあります	拡張スクリプト内にエラーがあります