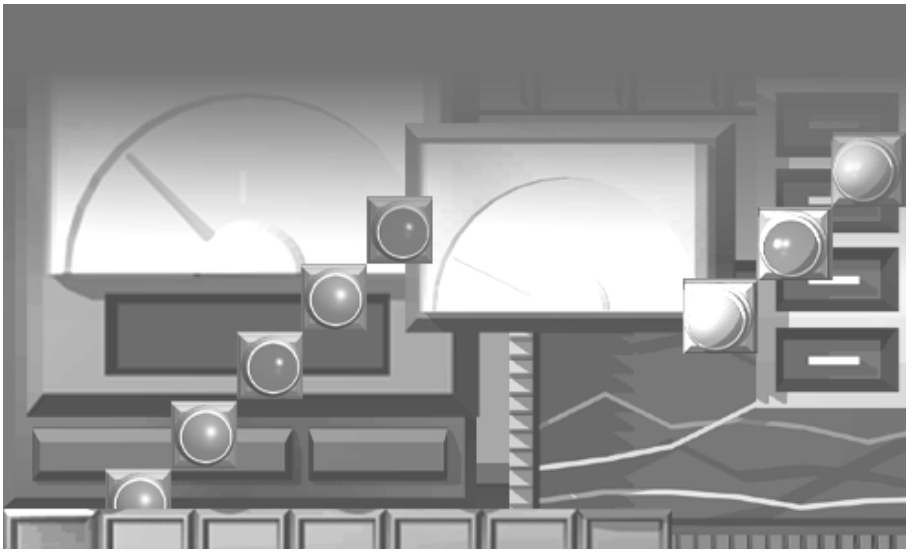
MITSUBISHI

GOT-A900シリーズ

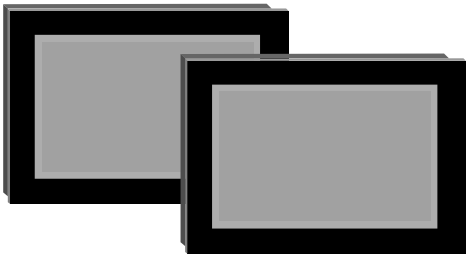
スクリプト機能事例集

MELSOFT



Graphic Operation Terminal
900
series

統合FAソフトウェア
MELSOFT



三菱グラフィックオペレーションターミナル

改 定 履 歴

※取扱説明書番号は，本説明書の裏表紙の左下に記載してあります。

印刷日付	※取扱説明書番号	改　定　内　容
2001年1月	BCN-P5464-A	初版印刷
2002年12月	BCN-P5464-B	GT Designer2のサンプルプログラムの格納先を追記

本書によって，工業所有権その他の権利の実施に対する保証，または実施権を許諾するものではありません。また本書の掲載内容の使用により起因する工業所有権上の諸問題については，当社は一切その責任を負うことができません。

© 2001 MITSUBISHI ELECTRIC CORPORATION

A - 1

は　じ　め　に

このたびは，三菱グラフィックオペレーションターミナルをお買い上げいただきまことにありがとうございました。

ご使用前に本書をよくお読みいただき，グラフィックオペレーションターミナルの機能・性能を十分ご理解のうえ，正しくご使用くださるようお願い致します。

目　　次

1　概　　要　　1- 1〜1- 2

1.1　本資料の読み方 ・・・

第1章 概　要

本資料はGT Designer Version5 A版以降または，GT Designer2でサポートしているスクリプト機能の使用例を記載したものです。

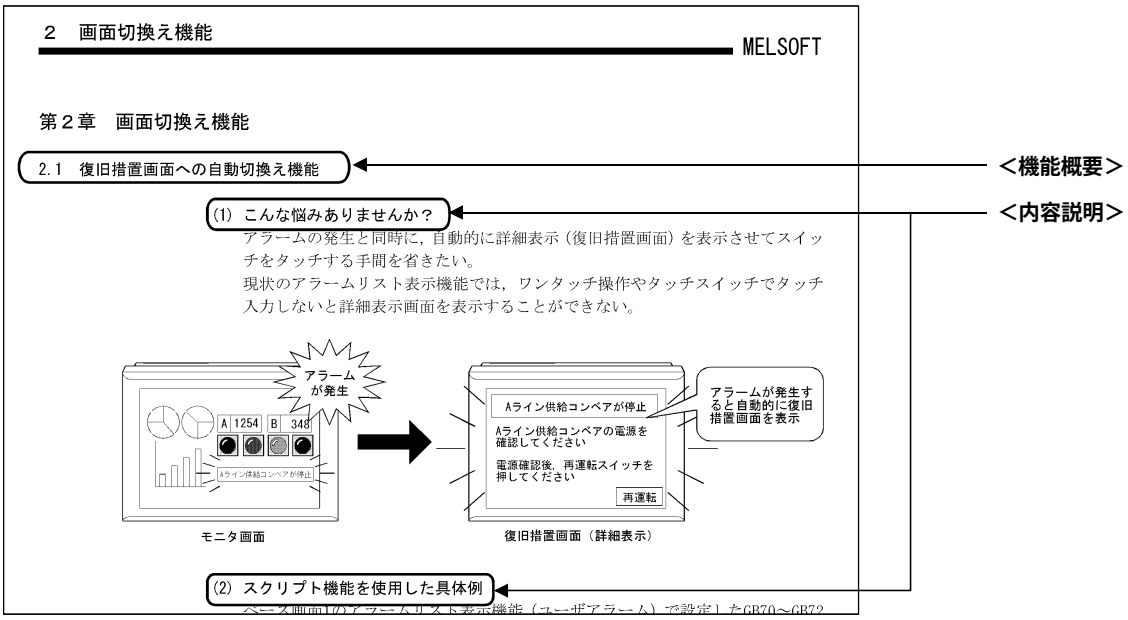
スクリプト機能の仕様，設定方法，制約などの詳細は，GT Works Version□/GT Designer Version□リファレンスマニュアルまたは，GT Designer2 Version□リファレンスマニュアルを参照してください。

ポイント

本設定例を使用または応用して，お客様のシステムでGOTを使用する場合，必ず最終検証を行った上で使用するようにしてください。

1.1 本資料の読み方

本資料の読み方を説明します。



＜機能概要＞

どのような機能の画面を作成するか記載しています。

＜内容説明＞

内容ごとに以下のように分類して説明しています。

(1) こんな悩みありませんか？

「現状のGOTの機能ではできないが，こんなことができたらいい」といった内容を記載しています。

(2) スクリプト機能を使用した具体例

機能概要の具体例を記載しています。
画面がどのように動作するかなどを説明しています。

1 - 1

(3) 設定例

画面作成時に設定するデバイス，オブジェクト機能などについて記載しています。
オブジェクト機能の説明では，機能を実行するための設定内容を記載しています。
設定内容は，そのオブジェクトのデフォルト値から変更する必要のある項目のみを特に記載しています。

(4) スクリプト設定

具体例で使用するスクリプトの設定内容やプログラムを記載しています。
また，参考としてスクリプト機能のデータサイズを記載しています。

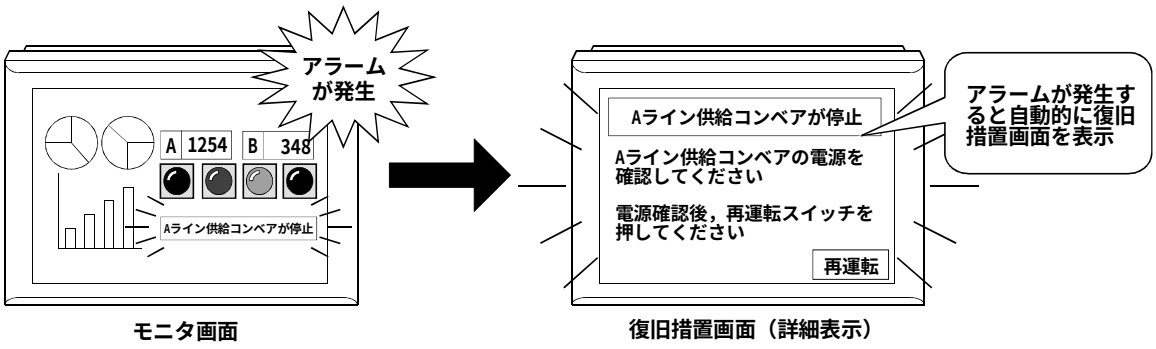
1 - 2

第２章　画面切換え機能

2.1　復旧措置画面への自動切換え機能

(1) こんな悩みありませんか？

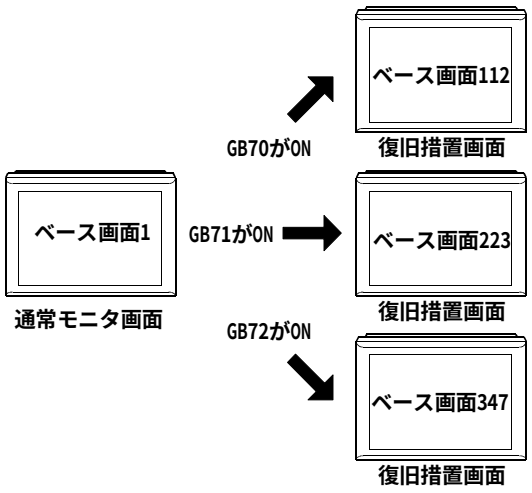
アラームの発生と同時に，自動的に詳細表示（復旧措置画面）を表示させてスイッチをタッチする手間を省きたい。
現状のアラームリスト表示機能では，ワンタッチ操作やタッチスイッチでタッチ入力しないと詳細表示画面を表示することができない。



(2) スクリプト機能を使用した具体例

ベース画面1のアラームリスト表示機能（ユーザアラーム）で設定したGB70～GB72の各デバイスがONした場合（エラーが発生した場合），各復旧措置説明画面に自動的に切換えます。

- GB70がONした場合：表示画面をベース画面112に切換えます。
- GB71がONした場合：表示画面をベース画面223に切換えます。
- GB72がONした場合：表示画面をベース画面347に切換えます。



本具体例はGT Designer, GT Designer2に格納されています。 GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\復旧措置画面への自動切換え機能 GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\復旧措置画面への自動切換え機能

2 - 1

ポイント

・ 本具体例では，アラームの発生と同時に画面切換えを行うため，ベース画面1で設定したアラーム表示機能（ユーザアラーム）の内容は確認できません。
・ 本具体例では，画面切換え後，対象となるビットデバイスは自動的にリセットします。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GD100	ベース画面切換えデバイス
GB70～GB72	ユーザアラームデバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
アラームリスト表示機能	アラーム表示用	＜基本設定タブ＞ アラーム種類：ユーザアラーム ＜表示設定タブ＞ デバイス点数：3，デバイス：GB70　連続（GB70～GB72が設定されます）

(4) スクリプト設定（約1kバイト）

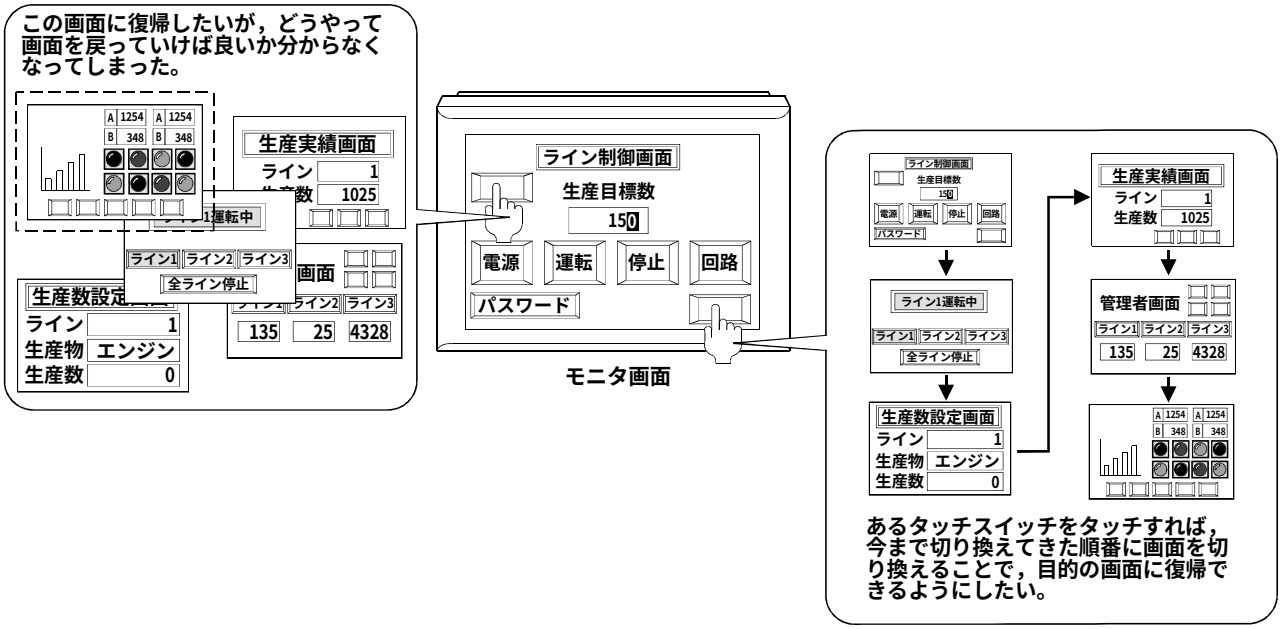
項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	常時
データ形式	16ビット／符号なしBIN

スクリプト記述	if ([b:GB70] == ON) {　　//アラーム1発生時
	　[w:GD100] = 112;　　//復旧措置説明画面（ベース画面112）へ切換え
	　[b:GB70] = OFF;　　//アラーム解除
	}
	if ([b:GB71] == ON) {　　//アラーム2発生時
	　[w:GD100] = 223;　　//復旧措置説明画面（ベース画面223）へ切換え
	　[b:GB71] = OFF;　　//アラーム解除
	}
	if ([b:GB72] == ON) {　　//アラーム3発生時
	　[w:GD100] = 347;　　//復旧措置説明画面（ベース画面347）へ切換え
	　[b:GB72] = OFF;　　//アラーム解除
	}

2.2　前画面スイッチ機能

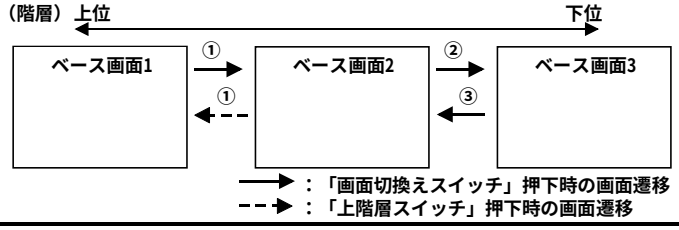
(1) こんな悩み，ありませんか？

画面数が多く，階層が深い複雑な画面構成になっているので，基の画面に戻る手順がわからなくなってしまうことが多い。
そこで，上階層スイッチではなく，切換え履歴通りに画面を戻すタッチスイッチを各画面に配置したいのだが，そのような機能は現状のGOTにはない。



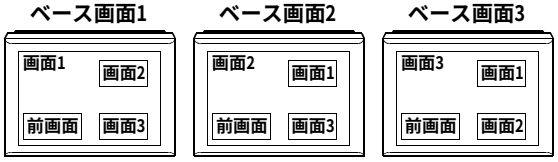
ポイント

・ 上階層スイッチ（スクリプト機能を使用しなくても可能）との違い
上階層スイッチは，スイッチをタッチするたびに1つ上の階層の画面に切り換えます。
そのため，下記のように画面を切り換えた場合，上階層スイッチをタッチするとベース画面3ではなく，ベース画面1に画面を切り換えるので切換え履歴通りに画面を切り換えることができない場合があります。



(2) スクリプト機能を使用した具体例

画面切換え用タッチスイッチで画面切換え後，前画面スイッチを押すことで切換え履歴通りにベース画面を切り換えます。



画面1 画面2 画面3 ･･･画面切換え用タッチスイッチ
前画面 ･･･････････前画面スイッチ

本具体例はGT Designer，GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\前画面スイッチ機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\前画面スイッチ機能

ポイント

・ 本具体例では，シーケンサからの画面切換え指示には対応しません。
・ スクリプトで設定したTMPデバイスの数だけ，履歴を記憶することができます。
　本具体例では3画面までの遷移が記憶可能です。

応用すればこんなこともできます

画面の切換え履歴をGOTの画面上に表示させたり，履歴ログをとることも可能です。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GD100	ベース画面切換えデバイス（共通設定）
GD200	ベース画面切換えデバイス（スクリプト）
GB100	前画面切換え用トリガデバイス
TMP0～TMP3	スタック用デバイス

(b) 関連オブジェクト設定

① ベース画面1

オブジェクト種別	用　途	設　定
タッチスイッチ	画面切換え用	＜動作設定タブ＞ [ワード]･･･デバイス：GD200，動作種別：固定値　2
タッチスイッチ	画面切換え用	＜動作設定タブ＞ [ワード]･･･デバイス：GD200，動作種別：固定値　3
タッチスイッチ	前画面切換え用	＜動作設定タブ＞ [ビット]･･･デバイス：GB100，動作種別：ビットSET

② ベース画面2

オブジェクト種別	用　途	設　定
タッチスイッチ	画面切換え用	＜動作設定タブ＞ [ワード]･･･デバイス：GD200，動作種別：固定値　1
タッチスイッチ	画面切換え用	＜動作設定タブ＞ [ワード]･･･デバイス：GD200，動作種別：固定値　3
タッチスイッチ	前画面切換え用	＜動作設定タブ＞ [ビット]･･･デバイス：GB100，動作種別：ビットSET

③ ベース画面3

オブジェクト種別	用　途	設　定
タッチスイッチ	画面切換え用	＜動作設定タブ＞ [ワード]･･･デバイス：GD200，動作種別：固定値　1
タッチスイッチ	画面切換え用	＜動作設定タブ＞ [ワード]･･･デバイス：GD200，動作種別：固定値　2
タッチスイッチ	前画面切換え用	＜動作設定タブ＞ [ビット]･･･デバイス：GB100，動作種別：ビットSET

(4) スクリプト設定（約9kバイト）

項　目	内　容
種別	プロジェクトスクリプト （ベース切換スイッチ押下時）
トリガ	常時
データ形式	16ビット／符号なしBIN

スクリプト記述	if ([w:GD200] != 0) {　　//[ベース切換SW押下時]
	[w:TMP3] = [w:TMP2];　　//スタックへPUSH①
	[w:TMP2] = [w:TMP1];　　//スタックへPUSH②
	[w:TMP1] = [w:TMP0];　　//スタックへPUSH③
	[w:TMP0] = [w:GD100];　　//
	[w:GD100] = [w:GD200];　　//ベース画面切換
	[w:GD200] = 0;　　//仮ベース画面切換デバイスクリア
	}

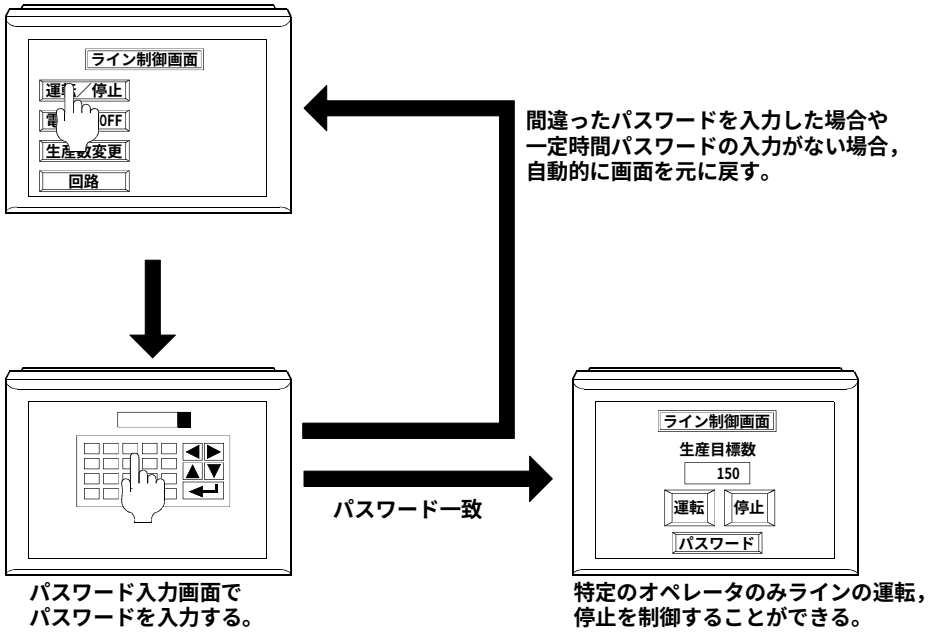
項　目	内　容
種別	プロジェクトスクリプト（前画面スイッチ押下時）
トリガ	立上り／GB100
データ形式	16ビット／符号なしBIN

スクリプト記述	[w:GD100] = [w:TMP0];　　//画面切換
	if ([w:TMP0] != 0) {　　//画面切換がまだある場合
	[w:TMP0]=[w:TMP1];　　//スタックからPOP①
	[w:TMP1]=[w:TMP2];　　//スタックからPOP②
	[w:TMP2]=[w:TMP3];　　//スタックからPOP③
	[w:TMP3]=0;
	}
	[b:GB100]　= 0;　　//トリガクリア

2.3　タイムアウト機能付きパスワード機能

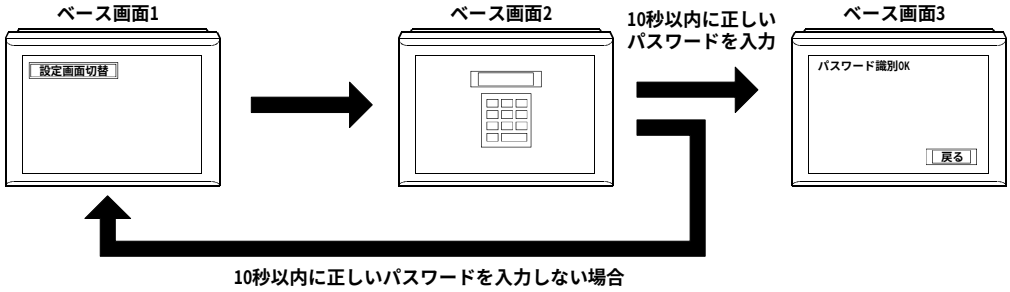
(1) こんな悩み，ありませんか？

特定のオペレータだけが参照できる画面を作成したい。
画面ごとに異なるパスワードを設け，不正パスワード入力時または一定時間入力のない場合，自動的に前画面に戻る画面を作成したいが，現状のパスワード機能では実現できない。



(2) スクリプト機能を使用した具体例

ベース画面1から，パスワード入力用のベース画面2に画面を切換えた後，ベース画面2上で10秒以内に正しいパスワード（本例ではパスワードを「3238」にしています。）の入力をした場合のみベース画面3に画面を切り換えます。
10秒以内に正しいパスワードの入力が無い場合，前画面（ベース画面1）に自動的に戻ります。



本具体例はGT Designer，GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\タイムアウト機能付きパスワード機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\タイムアウト機能付きパスワード機能

応用すればこんなこともできます

画面切換えだけでなく，タッチスイッチの動作などに応用することも可能です。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GB100	テンキー入力スクリプトトリガ用デバイス
GB101	パスワード確定スクリプトトリガ用デバイス
GD90	パスワード入力桁位置格納用デバイス
GD100	パスワード入力値格納用デバイス
GD150	テンキー入力値格納用デバイス
GD200～GD203	アスキー表示用データ格納用デバイス（パスワードマスク表示用）
GD300	ベース画面切換え用デバイス

(b) 関連オブジェクト設定

① 初期画面（ベース画面1）

オブジェクト種別	用　途	設　定
タッチスイッチ	設定画面切換えキー用	＜動作設定タブ＞ [ワード]･･･デバイス：GD90，動作種別：固定値　1 [ワード]･･･デバイス：GD100，動作種別：固定値　0 [ワード]･･･デバイス：GD200，動作種別：固定値　0 [ワード]･･･デバイス：GD201，動作種別：固定値　0 [ワード]･･･デバイス：GD202，動作種別：固定値　0 [ワード]･･･デバイス：GD203，動作種別：固定値　0 [ビット]･･･デバイス：GB100，動作種別：ビットRST [ビット]･･･デバイス：GB1011，動作種別：ビットRST [ベース切換え]･･･切換え先：固定値　2

② パスワード入力画面（ベース画面2）

オブジェクト種別	用　途	設　定
アスキー表示	マスク表示用	＜基本設定タブ＞ デバイス：GD200（4桁分でGD200～GD203を使用）
タッチスイッチ	テンキー(1)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150，動作種別：固定値　1 [ビット]･･･デバイス：GB100，動作種別：ビットSET
タッチスイッチ	テンキー(2)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150，動作種別：固定値　2 [ビット]･･･デバイス：GB100，動作種別：ビットSET
タッチスイッチ	テンキー(3)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150，動作種別：固定値　3 [ビット]･･･デバイス：GB100，動作種別：ビットSET
タッチスイッチ	テンキー(4)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150，動作種別：固定値　4 [ビット]･･･デバイス：GB100，動作種別：ビットSET
タッチスイッチ	テンキー(5)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150，動作種別：固定値　5 [ビット]･･･デバイス：GB100，動作種別：ビットSET

オブジェクト種別	用　途	設　定
タッチスイッチ	テンキー(6)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150, 動作種別：固定値　6 [ビット]･･･デバイス：GB100, 動作種別：ビットSET
タッチスイッチ	テンキー(7)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150, 動作種別：固定値　7 [ビット]･･･デバイス：GB100, 動作種別：ビットSET
タッチスイッチ	テンキー(8)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150, 動作種別：固定値　8 [ビット]･･･デバイス：GB100, 動作種別：ビットSET
タッチスイッチ	テンキー(9)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150, 動作種別：固定値　9 [ビット]･･･デバイス：GB100, 動作種別：ビットSET
タッチスイッチ	テンキー(0)用	＜動作設定タブ＞ [ワード]･･･デバイス：GD150, 動作種別：固定値　0 [ビット]･･･デバイス：GB100, 動作種別：ビットSET
タッチスイッチ	確定キー用	＜動作設定タブ＞ [ビット]･･･デバイス：GB101, 動作種別：ビットSET

③ パスワード識別OK画面（ベース画面3）

オブジェクト種別	用　途	設　定
タッチスイッチ	戻るキー用	＜動作設定タブ＞ [ベース切換え]･･･切換え先：固定値　1

(4) スクリプト設定（約3.3kバイト）

項　目	内　容
種別	画面スクリプト（ベース画面2：タイムアウト）
トリガ	周期（10秒）
データ形式	16ビット／符号付きBIN
スクリプト記述	[w:GD300]=1;　　//ベース画面1に切換

項　目	内　容
種別	画面スクリプト（ベース画面2：テンキー入力）
トリガ	立上り／GB100
データ形式	16ビット／符号付きBIN
スクリプト記述	switch([w:GD90]){　　//桁位置毎に処理を変える 　case 1:　　//1桁目 　　[w:GD200]=0x9681;　　//マスク表示“＊　　” 　　[w:GD201]=0x0000; 　　[w:GD202]=0x0000; 　　[w:GD203]=0x0000; 　　[w:GD100]=[w:GD100]+([w:GD150]*1000);　　//パスワード値作成 　　[w:GD90]=[w:GD90]+1;　　//桁位置+1 　　break; 　case 2:　　//2桁目 　　[w:GD200]=0x9681;　　//マスク表示“＊＊　　” 　　[w:GD201]=0x9681; 　　[w:GD202]=0x0000; 　　[w:GD203]=0x0000; 　　[w:GD100]=[w:GD100]+([w:GD150]*100);　　//パスワード値作成 　　[w:GD90]=[w:GD90]+1;　　//桁位置+1 　　break; 　case 3:　　//3桁目 　　[w:GD200]=0x9681;　　//マスク表示“＊＊＊　” 　　[w:GD201]=0x9681; 　　[w:GD202]=0x9681; 　　[w:GD203]=0x0000; 　　[w:GD100]=[w:GD100]+([w:GD150]*10);　　//パスワード値作成 　　[w:GD90]=[w:GD90]+1;　　//桁位置+1 　　break; 　case 4:　　//4桁目 　　[w:GD200]=0x9681;　　//マスク表示“＊＊＊＊” 　　[w:GD201]=0x9681; 　　[w:GD202]=0x9681; 　　[w:GD203]=0x9681; 　　[w:GD100]=[w:GD100]+[w:GD150];　　//パスワード値作成 　　[w:GD90]=[w:GD90]+1;　　//桁位置+1 　　break; 　default: 　　break; } rst([b:GB100]);　　//トリガデバイス初期化

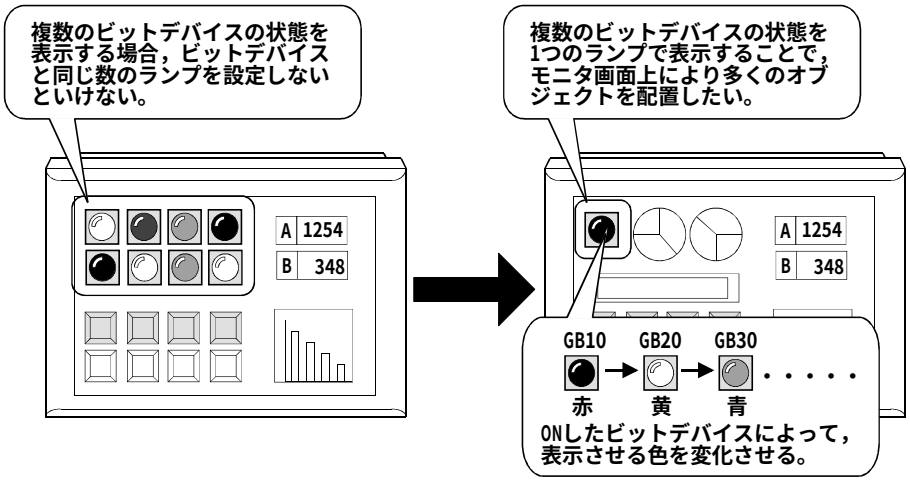
項　目	内　容
種別	画面スクリプト（ベース画面2：パスワード確定）
トリガ	立上り／GB101
データ形式	16ビット／符号付きBIN
スクリプト記述	**if([w:GD100]==3238){　　//パスワードチェック** **　[w:GD300]=3;　　//識別OK：ベース画面3に切換** **}** **else{** **　[w:GD300]=1;　　//識別NG：ベース画面1に切換** **}** **rst([b:GB101]);　　//トリガデバイス初期化**

第3章　ランプ表示機能

3.1　複数のビットデバイスの状態を表示するランプ表示機能

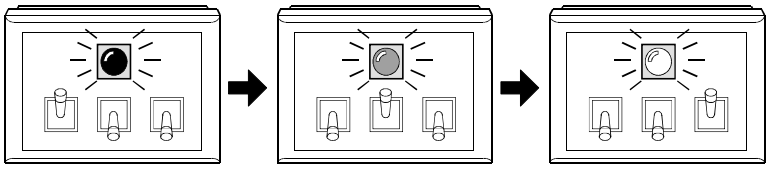
(1) こんな悩み，ありませんか？

複数のビットデバイスのON/OFF状態を1つのランプで表示させて，GOTに表示するランプの数を減らしたい。
現状のランプ表示機能では，1つのランプ表示機能に1つのビットデバイスのON/OFF状態しか表示できないので，複数のビットデバイスの状態をランプ表示機能で表示する場合，ビットデバイスと同じ数のランプ表示機能を設定しなければならない。



(2) スクリプト機能を使用した具体例

各スイッチをONすると対応したビットデバイス（GB10，GB20，GB30）がONします。
ONしたビットデバイスにより，ランプの表示色が変化します。



本具体例はGT Designer，GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\複数のビットデバイスの状態を表示するランプ表示機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\複数のビットデバイスの状態を表示するランプ表示機能

ポイント
本具体例ではビットデバイスの同時ON時のランプの表示は保証してません。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GD10	ランプ表示状態格納用デバイス
GB10	スイッチ1用トリガデバイス
GB20	スイッチ2用トリガデバイス
GB30	スイッチ3用トリガデバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
タッチスイッチ機能	スイッチ1用	＜動作設定タブ＞ [ビット]･･･デバイス：GB10，動作種別：ビットALT
タッチスイッチ機能	スイッチ2用	＜動作設定タブ＞ [ビット]･･･デバイス：GB20，動作種別：ビットALT
タッチスイッチ機能	スイッチ3用	＜動作設定タブ＞ [ビット]･･･デバイス：GB30，動作種別：ビットALT
ランプ表示機能	ランプ表示用	＜基本設定タプ＞ デバイス：GD10（ワードBIN16） ＜表示方法（ワード）タブ＞ 1･･･表示範囲：$V==0，枠色：182，ランプ色：255，背景色：255 2･･･表示範囲：$V==1，枠色：182，ランプ色：224，背景色：0 3･･･表示範囲：$V==2，枠色：182，ランプ色：31，背景色：0 4･･･表示範囲：$V==3，枠色：182，ランプ色：252，背景色：0

(4) スクリプト設定（約1.2kバイト）

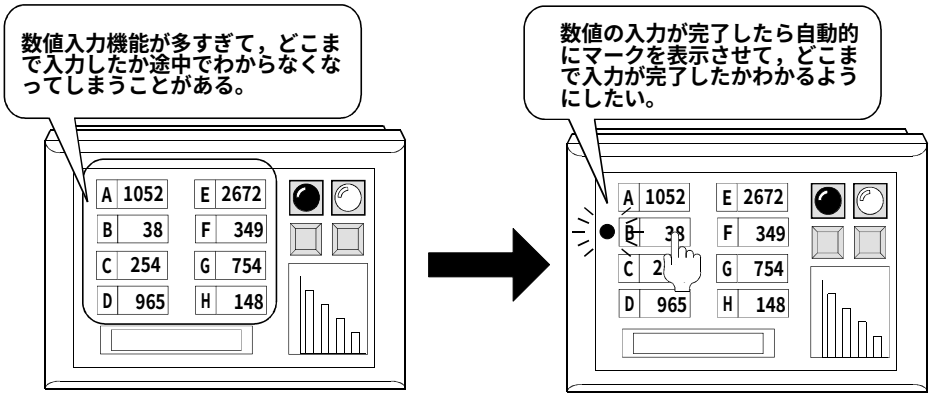
項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	常時
データ形式	16ビット／符号なしBIN

スクリプト記述	if ([b:GB10] == ON)
	　　[w:GD10] = 1;　　// ランプ表示状態：1
	　　return;
	}
	if ([b:GB20] == ON)
	　　[w:GD10] = 2;　　//ランプ表示状態：2
	　　return;
	}
	if ([b:GB30] == ON)
	　　[w:GD10] = 3;　　//ランプ表示状態：3
	　　return;
	}
	if (([b:GB10] == OFF)　　//全ビットデバイスの条件未成立時
	　　&& ([b:GB20] == OFF)
	　　&& ([b:GB30] == OFF)) {
	　　[w:GD10] = 0;　　//ランプ表示状態：4
	}

3.2　数値入力済みマーク表示機能

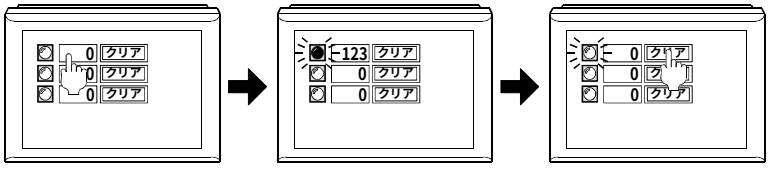
(1) こんな悩み，ありませんか？

画面上に設定した数値入力機能の数が多すぎて，どこまで入力がすんだのか途中でわからなくなることがある。
入力がすんだ数値入力については，入力が完了したマークを表示させたいが，そのような機能は現状のGOTにはない。



(2) スクリプト機能を使用した具体例

各数値入力枠の隣にランプを配置し，数値入力完了後に各ランプを点灯させます。
クリアスイッチを押すと，入力した数値はリセットされ，ランプが消灯します。



数値入力機能で，数値を入力します。

数値入力が完了すると，入力枠隣のランプが点灯します。

クリアスイッチを押すと数値をリセットして，ランプが消灯します。

本具体例はGT Designer，GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\数値入力済みマーク表示機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\数値入力済みマーク表示機能

ポイント

- 本具体例では，ランプの消灯には，それぞれのランプに対応したビットデバイスをリセットする必要があります。
- 本具体例では，共通設定のシステム情報設定を行う必要があります。

応用すればこんなこともできます

ランプの表示の代わりに，入力した順番を数値で表示させることも可能です。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GB100	入力枠1隣のランプ用デバイス
GB101	入力枠2隣のランプ用デバイス
GB102	入力枠3隣のランプ用デバイス
GD100	入力枠（数値入力）1用デバイス
GD101	入力枠（数値入力）2用デバイス
GD102	入力枠（数値入力）3用デバイス
GD200～GD214	システム情報設定の書込デバイス用
GD215	システム情報設定の読出デバイス用

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
ランプ表示機能	入力枠1用	＜基本設定タブ＞ デバイス：GB100
ランプ表示機能	入力枠2用	＜基本設定タブ＞ デバイス：GB101
ランプ表示機能	入力枠3用	＜基本設定タブ＞ デバイス：GB102
数値入力機能	入力枠1用	＜基本設定タブ＞ デバイス：GD100　16ビット　符号付きBIN ＜オプションタブ＞ ユーザID：1
数値入力機能	入力枠2用	＜基本設定タブ＞ デバイス：GD101　16ビット　符号付きBIN ＜オプションタブ＞ ユーザID：2
数値入力機能	入力枠3用	＜基本設定タブ＞ デバイス：GD102　16ビット　符号付きBIN ＜オプションタブ＞ ユーザID：3
タッチスイッチ機能	デバイスリセット用	＜動作設定タブ＞ [ワード]･･･デバイス：GD100，動作種別：固定値　0 [ビット]･･･デバイス：GB100，動作種別：ビットRST
タッチスイッチ機能	デバイスリセット用	＜動作設定タブ＞ [ワード]･･･デバイス：GD101，動作種別：固定値　0 [ビット]･･･デバイス：GB101，動作種別：ビットRST
タッチスイッチ機能	デバイスリセット用	＜動作設定タブ＞ [ワード]･･･デバイス：GD102，動作種別：固定値　0 [ビット]･･･デバイス：GB102，動作種別：ビットRST
システム情報機能	――	読出しデバイス：GD215，デバイス点数：1 書込みデバイス：GD200

(4) スクリプト設定（約1kバイト）

項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	常時
データ形式	16ビット／符号付きBIN

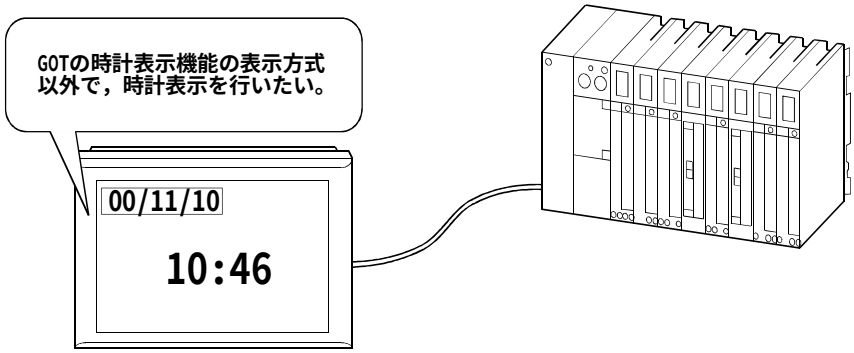
スクリプト記述	if([b:GD200.b4]==ON){ //数値入力信号がONの時
	switch([w:GD204]){ //数値入力番号格納エリアの値で入力された枠を識別
	case 1: //入力枠1
	[b:GB100]=ON; //ランプ1を点灯
	break;
	case 2: //入力枠2
	[b:GB101]=ON; //ランプ2を点灯
	break;
	default: //入力枠3
	[b:GB102]=ON; //ランプ3を点灯
	break;
	}
	[b:GD215.b4]=ON; //数値入力読出し完了信号をON
	}
	else{ //数値入力信号がOFFの時
	if([b:GD215.b4]==ON){ //数値入力読出し完了信号がONの時
	[b:GD215.b4]=OFF; //数値入力読出し完了信号をOFF
	}
	}

第４章　時計表示機能

4.1　自由フォーマット時計表示機能

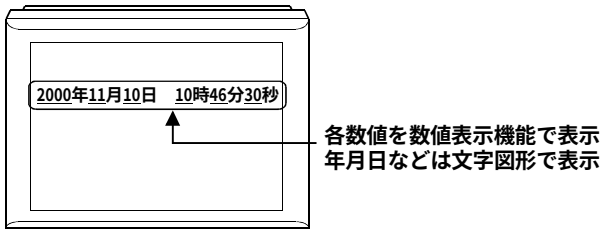
(1) こんな悩み，ありませんが？

GOTに表示させる時計表示の表示方式を変えたい。
現状の時計表示機能の表示方式は固定されているので実現できない。



(2) スクリプト機能を使用した具体例

QnACPUの時計データを加工して，GOTで時計表示を行います。
GOTで表示できないフォーマットで表示することが可能になります。



本具体例はGT Designer，GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\自由フォーマット時計表示機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\自由フォーマット時計表示機能

ポイント

・ 本具体例では，BCDで格納された下記データの年～分までを，マスク処理を行ってGD90～GD95に読出します。

SD210	年	月
SD211	日	時
SD212	分	秒

・ 本具体例では，接続先CPUをQnACPUにしています。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
SD210～SD212	時計データ（特殊レジスタ）
GD90～GD95	時計表示用デバイス
TMP0～TMP2	時計データ加工デバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
数値表示	時計表示（年）用	＜基本設定タブ＞ デバイス：GD90　16ビット　BCD ＜表示設定タブ＞ 表示形式：符号無し10進数
数値表示	時計表示（月）用	＜基本設定タブ＞ デバイス：GD91　16ビット　BCD ＜表示設定タブ＞ 表示形式：符号無し10進数
数値表示	時計表示（日）用	＜基本設定タブ＞ デバイス：GD92　16ビット　BCD ＜表示設定タブ＞ 表示形式：符号無し10進数
数値表示	時計表示（時）用	＜基本設定タブ＞ デバイス：GD93　16ビット　BCD ＜表示設定タブ＞ 表示形式：符号無し10進数
数値表示	時計表示（分）用	＜基本設定タブ＞ デバイス：GD94　16ビット　BCD ＜表示設定タブ＞ 表示形式：符号無し10進数
数値表示	時計表示（秒）用	＜基本設定タブ＞ デバイス：GD95　16ビット　BCD ＜表示設定タブ＞ 表示形式：符号無し10進数

(4) スクリプト設定（約1kバイト）

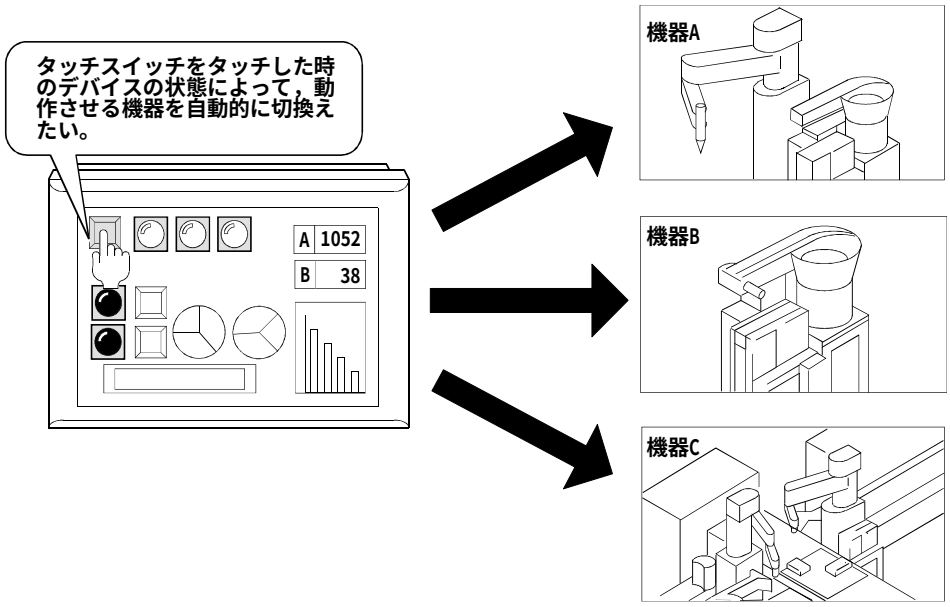
項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	立上り／GS10.b1
データ形式	16ビット／符号なしBIN
スクリプト記述	[w:TMP0] = [w:SD210] & 0xff00;　　//年データ設定 [w:GD90] = [w:TMP0] >> 8;　　//右8ビットシフトしてGD90に格納 [w:GD91] = [w:SD210] & 0xff;　　//月データ設定 [w:TMP1] = [w:SD211] & 0xff00;　　//日データ設定 [w:GD92] = [w:TMP1] >> 8;　　//右8ビットシフトしてGD92に格納 [w:GD93] = [w:SD211] & 0xff;　　//時データ設定 [w:TMP2] = [w:SD212] & 0xff00;　　//分データ設定 [w:GD94] = [w:TMP2] >> 8;　　//右8ビットシフトしてGD94に格納 [w:GD95] = [w:SD212] & 0xff;　　//秒データ設定

第5章　タッチスイッチ機能

5.1　状態自動判別タッチスイッチ機能

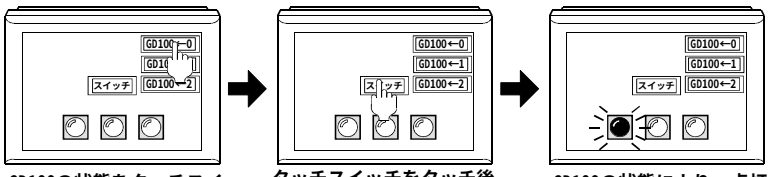
(1) こんな悩み，ありませんか？

始動スイッチをタッチした時のデバイスの状態によって，機器を変えて始動させたいが，誤動作などを防ぐために，オペレータにはその条件を意識させたくない。そこで，現在の状態を自動判別し，適切な機器を始動させるようなタッチスイッチを実現したいが，そのような機能は現状のGOTにはない。



(2) スクリプト機能を使用した具体例

タッチスイッチ押下時，ワードデバイスGD100の状態により，点灯させるランプ（セットするビットデバイス）を変化させます。



GD100の状態をタッチスイッチで変化させます。

タッチスイッチをタッチ後，ランプを表示させるタッチスイッチをタッチします。

GD100の状態により，点灯するランプが異なります。

本具体例はGT Designer, GT Designer2に格納されています。

GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\状態自動判別タッチスイッチ機能

GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\状態自動判別タッチスイッチ機能

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GB100	ランプ1用デバイス
GB101	ランプ2用デバイス
GB102	ランプ3用デバイス
GB200	スイッチ用デバイス
GD100	状態格納用デバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
ランプ表示機能	ランプ1用	＜基本設定タブ＞ デバイス：GB100
ランプ表示機能	ランプ2用	＜基本設定タブ＞ デバイス：GB101
ランプ表示機能	ランプ3用	＜基本設定タブ＞ デバイス：GB102
タッチスイッチ機能	タッチスイッチ用	＜動作設定タブ＞ [ビット]･･･デバイス：GB200，動作種別：ビットモーメンタリ

(4) スクリプト設定（約850バイト）

項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	立上り／GB200
データ形式	16ビット／符号なしBIN

スクリプト記述	if([w:GD100]==0){　　//GD100が0の時
	　[b:GB100]=ON;　　//ランプ1点灯
	}
	else{　　//GD100が0以外の時
	　[b:GB100]=OFF;　　//ランプ1消灯
	}
	if([w:GD100]==1){　　//GD100が1の時
	　[b:GB101]=ON;　　//ランプ2点灯
	}
	else{　　//GD100が1以外の時
	　[b:GB101]=OFF;　　//ランプ2消灯
	}
	if([w:GD100]==2){　　//GD100が2の時
	　[b:GB102]=ON;　　//ランプ3点灯
	}
	else{　　//GD100が2以外の時
	　[b:GB102]=OFF;　　//ランプ3消灯
	}

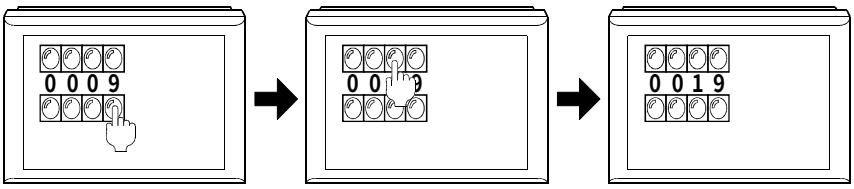
5.2 デジスイッチ機能

(1) こんな悩み，ありませんか？

デジスイッチを実現させたいが，現状のGOTにはそのような機能はない。

(2) スクリプト機能を使用した具体例

4桁の10進数デジスイッチで，タッチスイッチを押すごとに各桁の数値がインクリメントまたはデクリメントします。
インクリメントまたはデクリメントするスイッチによる入力は自動的にGD200へ反映されます。



上段のタッチスイッチをタッチするたびに数値を1ずつ増やします。（“9”の次は“0”を表示します。）
下段のタッチスイッチをタッチするたびに数値を1ずつ減らします。（“0”の次は“9”を表示します。）

本具体例はGT Designer，GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\デジスイッチ機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\デジスイッチ機能

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GD200	デジスイッチ変更値格納デバイス
GD100	4桁目数値一時格納デバイス
GD101	3桁目数値一時格納デバイス
GD102	2桁目数値一時格納デバイス
GD103	1桁目数値一時格納デバイス
GB100	4桁目インクリメント用トリガデバイス
GB101	4桁目デクリメント用トリガデバイス
GB200	3桁目インクリメント用トリガデバイス
GB201	3桁目デクリメント用トリガデバイス
GB300	2桁目インクリメント用トリガデバイス
GB301	2桁目デクリメント用トリガデバイス
GB400	1桁目インクリメント用トリガデバイス
GB401	1桁目デクリメント用トリガデバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
タッチスイッチ機能	4桁目インクリメント用 トリガ値操作	＜動作設定タブ＞ [ビット]･･･デバイス：GB100，動作種別：ビットモーメンタリ
タッチスイッチ機能	4桁目デクリメント用 トリガ値操作	＜動作設定タブ＞ [ビット]･･･デバイス：GB101，動作種別：ビットモーメンタリ
タッチスイッチ機能	3桁目インクリメント用 トリガ値操作	＜動作設定タブ＞ [ビット]･･･デバイス：GB200，動作種別：ビットモーメンタリ
タッチスイッチ機能	3桁目デクリメント用 トリガ値操作	＜動作設定タブ＞ [ビット]･･･デバイス：GB201，動作種別：ビットモーメンタリ
タッチスイッチ機能	2桁目インクリメント用 トリガ値操作	＜動作設定タブ＞ [ビット]･･･デバイス：GB300，動作種別：ビットモーメンタリ
タッチスイッチ機能	2桁目デクリメント用 トリガ値操作	＜動作設定タブ＞ [ビット]･･･デバイス：GB301，動作種別：ビットモーメンタリ
タッチスイッチ機能	1桁目インクリメント用 トリガ値操作	＜動作設定タブ＞ [ビット]･･･デバイス：GB400，動作種別：ビットモーメンタリ
タッチスイッチ機能	1桁目デクリメント用 トリガ値操作	＜動作設定タブ＞ [ビット]･･･デバイス：GB401，動作種別：ビットモーメンタリ
数値表示機能	4桁目値表示用	＜基本設定タブ＞ デバイス：GD100，16ビット符号付きBIN ＜表示設定タブ＞ 表示桁数：1
数値表示機能	3桁目値表示用	＜基本設定タブ＞ デバイス：GD101，16ビット符号付きBIN ＜表示設定タブ＞ 表示桁数：1
数値表示機能	2桁目値表示用	＜基本設定タブ＞ デバイス：GD102，16ビット符号付きBIN ＜表示設定タブ＞ 表示桁数：1
数値表示機能	1桁目値表示用	＜基本設定タブ＞ デバイス：GD103，16ビット符号付きBIN ＜表示設定タブ＞ 表示桁数：1
数値表示機能	デジスイッチ 操作デバイス値表示用	＜基本設定タブ＞ デバイス：GD200，16ビット符号付きBIN ＜表示設定タブ＞ 表示桁数：4

(4) スクリプト設定（約4kバイト）

項　目	内　容
種別	画面スクリプト（No.1）
トリガ	立上り／GB100
データ形式	16ビット／符号なしBIN

スクリプト記述	//4桁目のインクリメントスイッチ押下時の制御
	if([w:GD100] >= 9){　　//4桁目"9"ならば、"0"に戻す
	　[w:GD100] = 0;
	}else{　　//そうでなければインクリメント
	　[w:GD100] = [w:GD100]+1;
	}

項　目	内　容
種別	画面スクリプト（No.2）
トリガ	立上り／GB101
データ形式	16ビット／符号なしBIN

スクリプト記述	// 4桁目のデクリメントスイッチ押下時の制御
	if([w:GD100] <= 0){　　//4桁目"0"ならば、"9"に戻す
	　[w:GD100] = 9;
	}else{　　//そうでなければデクリメント
	　[w:GD100] = [w:GD100]-1;
	}

項　目	内　容
種別	画面スクリプト（No.3）
トリガ	立上り／GB200
データ形式	16ビット／符号なしBIN

スクリプト記述	//3桁目のインクリメントスイッチ押下時の制御
	if([w:GD101] >= 9){　　//3桁目"9"ならば、"0"に戻す
	　[w:GD101] = 0;
	}else{　　//そうでなければインクリメント
	　[w:GD101] = [w:GD101]+1;
	}

項　目	内　容
種別	画面スクリプト（No.4）
トリガ	立上り／GB201
データ形式	16ビット／符号なしBIN

スクリプト記述	//3桁目のデクリメントスイッチ押下時の制御
	if([w:GD101] <= 0){　　//3桁目"0"ならば、"9"に戻す
	　[w:GD101] = 9;
	}else{　　//そうでなければデクリメント
	　[w:GD101] = [w:GD101]-1;
	}

項　目	内　容
種別	画面スクリプト（No.5）
トリガ	立上り／GB300
データ形式	16ビット／符号なしBIN

スクリプト記述	//2桁目のインクリメントスイッチ押下時の制御
	if([w:GD102] >= 9){　　//2桁目"9"ならば、"0"に戻す
	　[w:GD102] = 0;
	}else{　　//そうでなければインクリメント
	　[w:GD102] = [w:GD102]+1;
	}

項　目	内　容
種別	画面スクリプト（No.6）
トリガ	立上り／GB301
データ形式	16ビット／符号なしBIN

スクリプト記述	//2桁目のデクリメントスイッチ押下時の制御
	if([w:GD102] <= 0){　　//2桁目"0"ならば、"9"に戻す
	　[w:GD102] = 9;
	}else{　　//そうでなければデクリメント
	　[w:GD102] = [w:GD102]-1;
	}

項　目	内　容
種別	画面スクリプト（No.7）
トリガ	立上り／GB400
データ形式	16ビット／符号なしBIN

スクリプト記述	//1桁目のインクリメントスイッチ押下時の制御
	if([w:GD103] >= 9){　　//1桁目"9"ならば、"0"に戻す
	　[w:GD103] = 0;
	}else{　　//そうでなければインクリメント
	　[w:GD103] = [w:GD103]+1;
	}

項　目	内　容
種別	画面スクリプト（No.8）
トリガ	立上り／GB401
データ形式	16ビット／符号なしBIN

スクリプト記述	//1桁目のデクリメントスイッチ押下時の制御
	if([w:GD103] <= 0){　　//1桁目"0"ならば、"9"に戻す
	　[w:GD103] = 9;
	}else{　　//そうでなければデクリメント
	　[w:GD103] = [w:GD103]-1;
	}

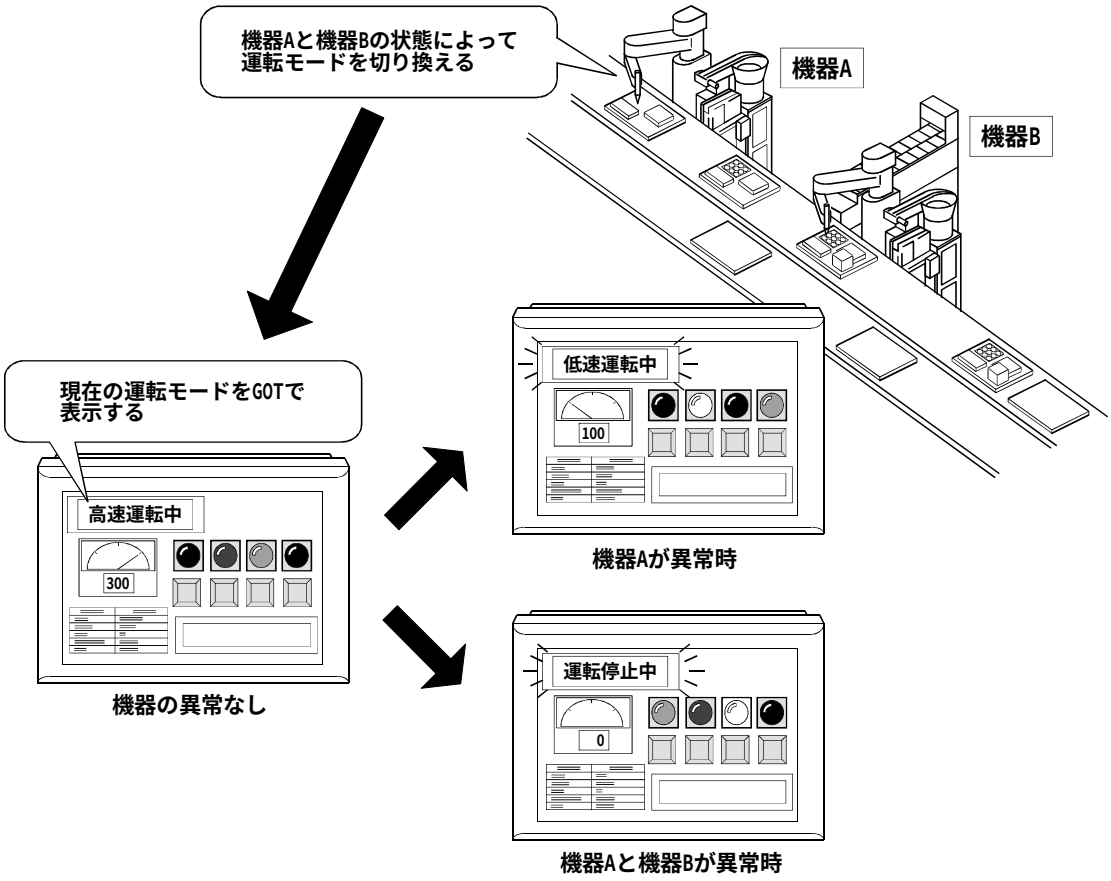
項　目	内　容
種別	画面スクリプト（No.9）
トリガ	常時
データ形式	16ビット／符号なしBIN
スクリプト記述	//4桁分の値をマージし，GD200へセットする [w:GD200] = [w:GD100]*1000 + [w:GD101]*100 　　+ [w:GD102]*10 + [w:GD103];

第6章　部品表示機能

6.1　動作モード判別機能

(1) こんな悩み，ありませんか？

動作中の複数の機器で異常が発生した場合，運転モード（低速／高速など）を切り換えて，現在の運転モードをGOTに表示させたい。
異常センサと運転モードの関係を表現する場合，複雑な論理演算式を設定する必要があり，GOTの状態監視機能では実現できない。

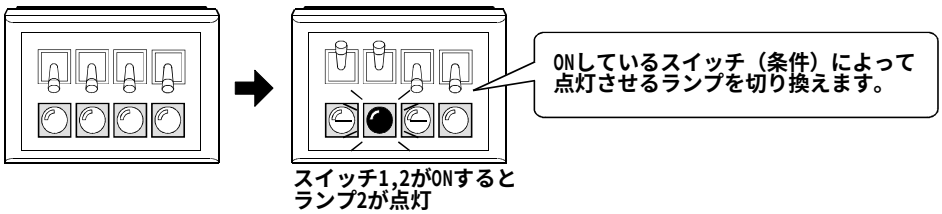


6 - 1

(2) スクリプト機能を使用した具体例

4個のビットスイッチの状態により，点灯するランプを変化させます。
各ランプが点灯する条件を下記に示します。

項　目	状態（条件）			
	スイッチ1	スイッチ2	スイッチ3	スイッチ4
ランプ1が点灯	スイッチ1～4がすべてON			
ランプ2が点灯	スイッチ1,2の両方がON		スイッチ3,4の片方または両方がOFF	
ランプ3が点灯	スイッチ1,2の片方または両方がOFF		スイッチ3,4の両方がON	
ランプ4が点灯	スイッチ1,2の片方または両方がOFF		スイッチ3,4の片方または両方がOFF	



本具体例はGT Designer, GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\動作モード判別機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\動作モード判別機能

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GB100	スイッチ1用デバイス
GB101	スイッチ2用デバイス
GB102	スイッチ3用デバイス
GB103	スイッチ4用デバイス
GD200	ランプ1用デバイス
GD200	ランプ2用デバイス
GD200	ランプ3用デバイス
GD200	ランプ4用デバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
タッチスイッチ機能	スイッチ1用	＜基本設定タブ＞ 表示切換え：ビット，デバイス：GB100 ＜動作設定タブ＞ 「ビット」･･･デバイス：GB100，動作種別：ビットALT
タッチスイッチ機能	スイッチ2用	＜基本設定タブ＞ 表示切換え：ビット，デバイス：GB101 ＜動作設定タブ＞ 「ビット」･･･デバイス：GB101，動作種別：ビットALT
タッチスイッチ機能	スイッチ3用	＜基本設定タブ＞ 表示切換え：ビット，デバイス：GB102 ＜動作設定タブ＞ 「ビット」･･･デバイス：GB102，動作種別：ビットALT
タッチスイッチ機能	スイッチ4用	＜基本設定タブ＞ 表示切換え：ビット，デバイス：GB103 ＜動作設定タブ＞ 「ビット」･･･デバイス：GB103，動作種別：ビットALT
ランプ表示機能	ランプ1用	＜基本設定タブ＞ デバイス：GD200.b0
ランプ表示機能	ランプ2用	＜基本設定タブ＞ デバイス：GD200.b1
ランプ表示機能	ランプ3用	＜基本設定タブ＞ デバイス：GD200.b2
ランプ表示機能	ランプ4用	＜基本設定タブ＞ デバイス：GD200.b3

(4) スクリプト設定（約1.2kバイト）

項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	常時
データ形式	16ビット／符号なしBIN

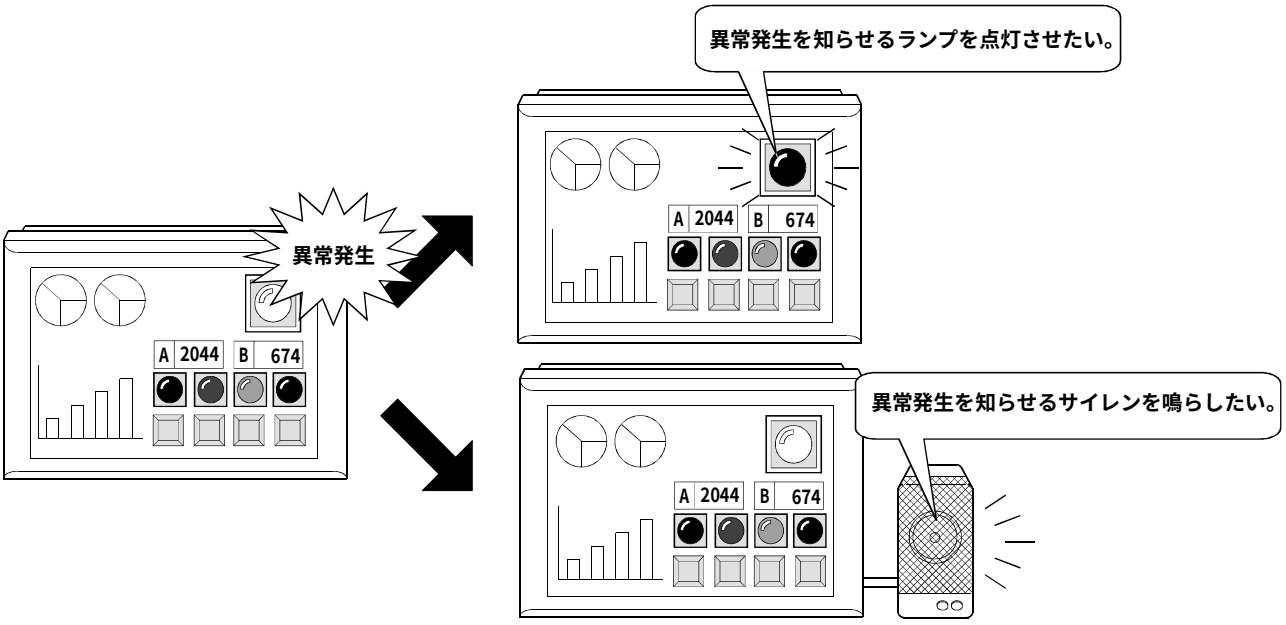
スクリプト記述	if(([b:GB100]==ON)　　//ランプ1点灯条件成立時
	&&([b:GB101]==ON)
	&&([b:GB102]==ON)
	&&([b:GB103]==ON)){
	　[w:GD200] = 1;　　//ランプ1のみ点灯
	　return;
	}
	if(([b:GB100]==ON)　　//ランプ2点灯条件成立時
	&&([b:GB101]==ON)
	&&(!([b:GB102]==ON)
	&&([b:GB103]==ON))){
	　[w:GD200] = 2;　　//ランプ2のみ点灯
	　return;
	}
	if(([b:GB102]==ON)　　//ランプ3点灯条件成立時
	&&([b:GB103]==ON)
	&&(!([b:GB100]==ON)
	&&([b:GB101]==ON))){
	　[w:GD200] = 4;　　//ランプ3のみ点灯
	}
	else{　　//ランプ4点灯条件成立時（上記条件で成立しなかった時）
	　[w:GD200] = 8;　　//ランプ4のみ点灯
	}

6.2 異常発生警報機能

(1) こんな悩み，ありませんか？

具体的内容については別途アラームリスト表示機能で確認するとして，要因を問わずに，とにかく異常が発生したらオペレータがまずその異常に気付くような出力をするシンプルな画面（1個のランプ出力もしくは，サイレンの音声出力）を作りたい。

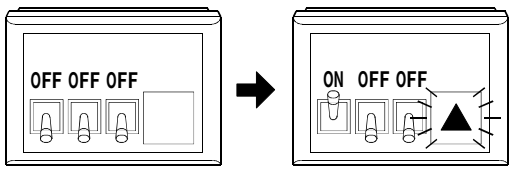
しかし要因となるビットをすべて監視するロジックが必要であり，そのような機能は現状のGOTにはない。



(2) スクリプト機能を使用した具体例

3個のビットデバイス（GB100，GB200，GB300）のうち，1つ以上がONの場合，部品を表示します。

ビットデバイスがすべてOFFの場合，部品は表示しません。



本具体例はGT Designer，GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\異常発生警報機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\異常発生警報機能

応用すればこんなこともできます

GOTに警告灯などを接続して，異常発生時に点灯させることも可能です。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GB100	スイッチ1用デバイス
GB200	スイッチ2用デバイス
GB300	スイッチ3用デバイス
GB500	部品表示デバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
タッチスイッチ機能	スイッチ1用	＜基本設定タブ＞ 表示切換え：ビット，デバイス：GB100 ＜動作設定タブ＞ 「ビット」･･･デバイス：GB100，動作種別：ビットALT
タッチスイッチ機能	スイッチ2用	＜基本設定タブ＞ 表示切換え：ビット，デバイス：GB200 ＜動作設定タブ＞ 「ビット」･･･デバイス：GB200，動作種別：ビットALT
タッチスイッチ機能	スイッチ3用	＜基本設定タブ＞ 表示切換え：ビット，デバイス：GB300 ＜動作設定タブ＞ 「ビット」･･･デバイス：GB300，動作種別：ビットALT
部品表示機能	部品用	＜基本設定タブ＞ デバイス：GB500 ＜表示方法（ビット）タブ＞ ON･･･部品No.1，OFF･･･部品No.0

(4) スクリプト設定（約700バイト）

項　目	内　容
種別	画面スクリプト
トリガ	常時
データ形式	16ビット／符号なしBIN

スクリプト記述	if (([b: GB100] == ON)　　//いずれかのトリガON時
	　|| ([b: GB200] == ON)
	　 || ([b: GB300] == ON)){
	　 [b:GB500] = ON;　　//部品を表示
	}else{　　//全トリガOFF時
	　[b:GB500] = OFF;　　//部品を非表示
	}

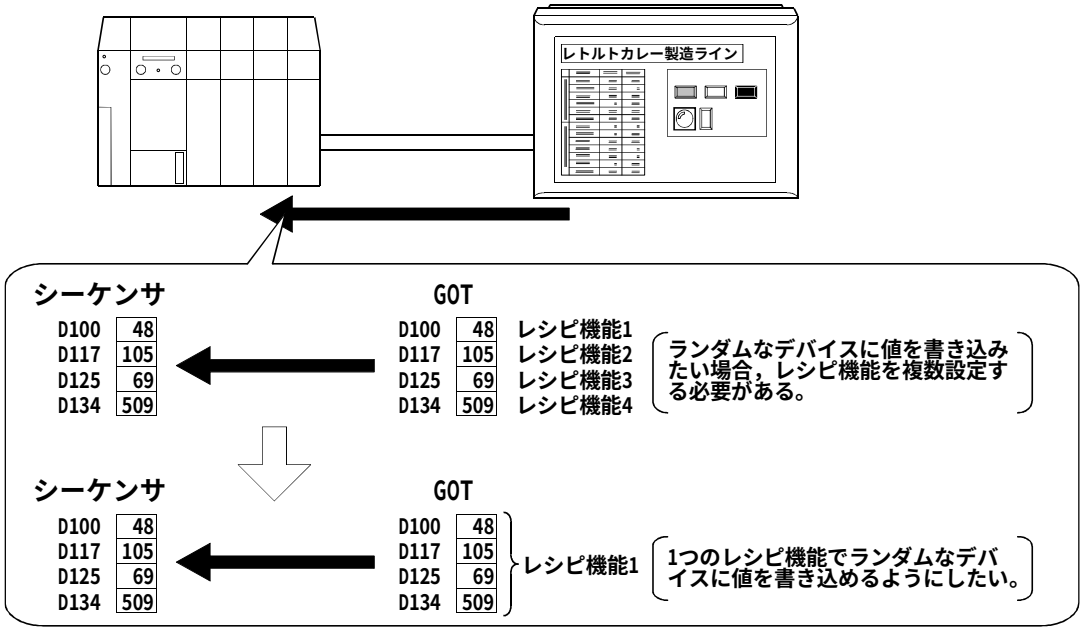
第7章　レシピ機能

7.1　ランダムデバイス書込みレシピ機能

(1) こんな悩み，ありませんか？

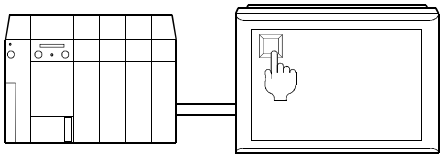
書込みトリガは同一で，書込み先のデバイスがランダムな書込みレシピを使いたい。

現状のレシピ機能ではランダムデバイスに対応していないため，複数のレシピ機能に分けて設定する必要があり，作画設定が非常に面倒である上，設定数が足りなくなる場合がある。



(2) スクリプト機能を使用した具体例

通常のレシピ機能ではランダムなデバイスにデータを書き込むことができませんが，本例では書込レシピ実行スイッチをタッチすると，ランダムなデバイス（9点）に初期値を書込みます。



書き込みレシピ実行スイッチをタッチするとシーケンサCPUの指定デバイスに，デバイス値が書き込まれます。

本具体例はGT Designer，GT Designer2に格納されています。
GT Designer：　\Melsec\Gotr\Examples\A975got\09 スクリプト\ランダムデバイス書込みレシピ機能
GT Designer2：　\Melsec\GTD2\Example\A975got\09 スクリプト\ランダムデバイス書込みレシピ機能

ポイント

本具体例は，Q/QnA/ACPUとの接続を想定して作成しています。
他社製シーケンサを使用する場合，デバイス値を変更する必要があります。

(3) 設定例

(a) 使用デバイス

デバイス種別		用　途
X0		書込みレシピ実行トリガ
D100		データリセット値設定デバイス
D200～D299	D200	書込みレシピ対象デバイス（ランダム）
	D203	
	D205	
	D210	
	D220	
	D230	
	D240	
	D250	
	D260	
	上記以外	数値データ表示用

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
タッチスイッチ機能	書込みレシピ実行用	＜動作設定タブ＞ [ビット]･･･デバイス：X0, 動作種別：ビットモーメンタリ
タッチスイッチ機能	デバイス値クリア用	＜動作設定タブ＞ [ビット]･･･デバイス：X1, 動作種別：ビットモーメンタリ
数値入力機能	データ表示用（100点）	＜基本設定タブ＞ デバイス：D200～D299
数値入力機能	リセット値入力用	＜基本設定タブ＞ デバイス：D100

(4) スクリプト設定（約1kバイト）

項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	立上り／X0
データ形式	16ビット／符号なしBIN
スクリプト記述	[w:D200] = 100; [w:D203] = 76; [w:D205] = 1; [w:D210] = 2; [w:D220] = 1; [w:D230] = 1; [w:D240] = 30; [w:D250] = 4300; [w:D260] = 200;

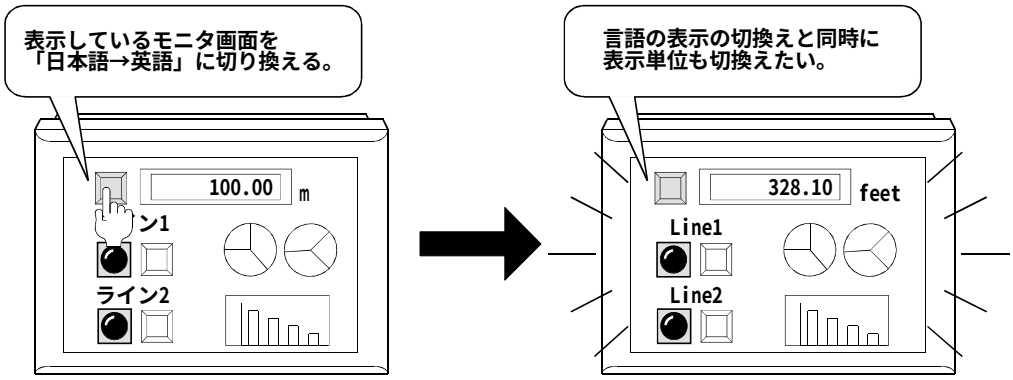
項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	立上り／X1
データ形式	16ビット／符号なしBIN
スクリプト記述	fmov([w:D100],[w:D200],100);

第８章　その他の使用例

8.1　日英単位自動変換機能

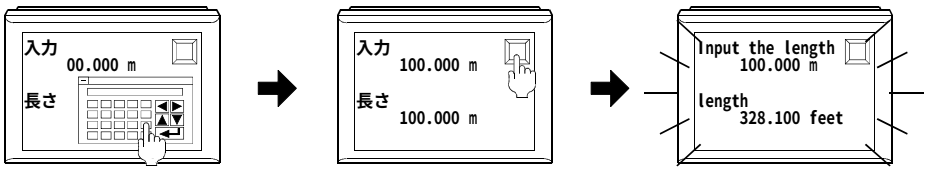
(1) こんな悩み，ありませんか？

画面の表示を日本語から英語に切り換えた場合，表示する単位の種類も同時に切り換えたい。
例えば日本語モードでは“ｍ”，英語モードでは“ feet ”に単位を切り換えてGOTに表示したいが，そのような機能は現状のGOTにはない。



(2) スクリプト機能を使用した具体例

表示したいデータをm単位で入力し，条件（GD100の値）に応じて対応する単位（m/feet）で入力値を表示します。
表示枠の右下には表示しているデータの単位を表示します。



数値を入力　　言語切換えスイッチをタッチ　　言語と一緒に数値の単位も切り換わります

8

本具体例はGT Designer, GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\日英単位自動変換機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\日英単位自動変換機能

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GB100	単位切換え（ケース切換え）用デバイス
GD110	データ出力用デバイス
GD150	データ入力用デバイス
GB200	単位出力用デバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
数値入力機能	データ入力用	＜基本設定タブ＞ デバイス：GD150，実数，32ビット ＜表示設定タブ＞ 表示形式：実数，表示桁数：9，少数桁数：3
数値表示機能	データ出力用	＜基本設定タブ＞ デバイス：GD110，実数，32ビット ＜表示設定タブ＞ 表示形式：実数，表示桁数：9，小数桁数：3
部品表示機能＊	文字（入力）表示用	＜基本設定タブ＞ デバイス；GB100 ＜表示方法（ビット）タブ＞ ON：部品No.4，OFF：部品No3
部品表示機能＊	文字（長さ）表示用	＜基本設定タブ＞ デバイス；GB100 ＜表示方法（ビット）タブ＞ ON：部品No.6，OFF：部品No5
部品表示機能＊	単位出力用	＜基本設定タブ＞ デバイス；GB200，ビット ＜表示方法（ビット）タブ＞ ON：部品No.2，OFF：部品No1
タッチスイッチ機能	言語切換えスイッチ用	＜基本設定タブ＞ デバイス；GB100，ビット ＜表示方法（ビット）タブ＞ 文字･･･直接：ON文字“上,English”，OFF文字“上，日本語” ＜動作設定タブ＞ [ビット]･･･デバイス：GB100 動作種別：ビットALT

＊ 部品表示機能で使用する単位は下記の部品No.であらかじめ登録しておきます。
1.m，2.feet，3.入力，4.Input the length，5.長さ，6.length

(4) スクリプト設定（約1.4kバイト）

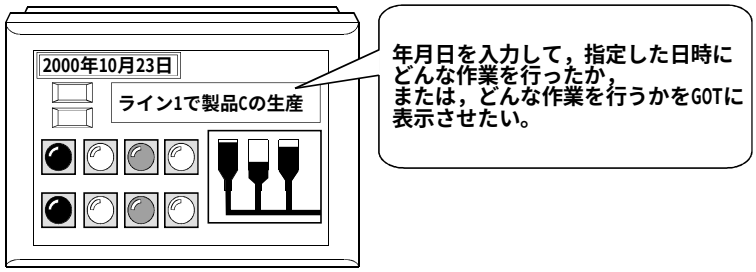
項　目	内　容
種別	画面スクリプト（ベース画面1）
トリガ	常時
データ形式	実数

スクリプト記述	if([b:GB100] == OFF){
	　[w:GD110] = [w:GD150];　// GD150値(m)をそのまま格納
	　[b:GB200] = OFF;　// 単位 "m" を表示
	}else{
	　[w:GD110] = [w:GD150] * 3.281;　// GD150値(m)を ft 値に再計算後格納
	　[b:GB200] = ON;　// 単位 "ft" を表示
	}

8.2　指定年月日の曜日算出機能

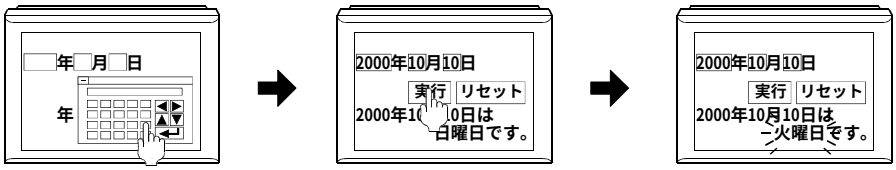
(1) こんな悩み，ありませんか？

曜日によって実施する工程が異なるので，年月日を入力して曜日を求めて，その日にどのような工程を実施したか，またどのような工程を実施するかを表示させたいが，そのような機能は現状のGOTにはない。



(2) スクリプト機能を使用した具体例

年月日を入力後，実行ボタンを押すとその曜日を表示します。
1度曜日を求めた後に再度別の曜日を求める場合は，リセットボタンを押し，入力した年月日をクリアしてから年月日を再入力してください。
連続して曜日は求めれません。
また，1899年以前の曜日は表示できません。



年月日を入力　　実行スイッチをタッチ　　入力した年月日の曜日を表示します。

本具体例はGT Designer，GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\指定年月日の曜日算出機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\指定年月日の曜日算出機能

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
TMP5	1900年を起点とした場合、最初の閏年
GD200	曜日出力用デバイス
GD100	「年」入力用デバイス
GD101	「月」入力用デバイス
GD102	「日」入力用デバイス
TMP0	カウンタ
TMP1	指定年までに存在する閏年数
TMP2	yyyy/1/1までの通算日
TMP3	yyyy/1/1からyyyy/mm/1までの通算日
TMP4	yyyy/mm/1からyyyy/mm/ddまでの通算日
TMP10	1900/1/1からの通算日
GB100	スクリプト実行トリガデバイス

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
数値入力機能	「年」入力用	＜基本設定タブ＞ デバイス：GD100 ＜表示設定タブ＞ 表示桁数：4 ＜範囲設定タブ＞ 表示範囲：1900<=$V
数値入力機能	「月」入力用	＜基本設定タブ＞ デバイス：GD101 ＜表示設定タブ＞ 表示桁数：2 ＜範囲設定タブ＞ 表示範囲：1<=$V<=12
数値入力機能	「日」入力用	＜基本設定タブ＞ デバイス：GD102 ＜表示設定タブ＞ 表示桁数：2 ＜範囲設定タブ＞ 表示範囲：1<=$V<=31
タッチスイッチ機能	「実行」スイッチ用	＜動作設定タブ＞ [ビット]･･･デバイス：GB100，動作種別：ビットSET
タッチスイッチ機能	「リセット」スイッチ用	＜動作設定タブ＞ [ワード]･･･デバイス：GD100，動作種別：固定値　0 [ワード]･･･デバイス：GD101，動作種別：固定値　0 [ワード]･･･デバイス：GD102，動作種別：固定値　0 [ワード]･･･デバイス：GD200，動作種別：固定値　7 [ビット]･･･デバイス：GD100，動作種別：ビットRST
数値表示機能	「年」表示用	＜基本設定タブ＞ デバイス：GD100 ＜表示設定タブ＞ 表示桁数：2
数値表示機能	「月」表示用	＜基本設定タブ＞ デバイス：GD101
数値表示機能	「日」表示用	＜基本設定タブ＞ デバイス：GD102 ＜表示設定タブ＞ 表示桁数：2
コメント表示機能＊	「曜日」表示用	＜基本設定タブ＞ デバイス：GD200 ワード(BIN16) ＜表示方法（ワード）＞ 1 表示範囲：$V==0，コメントNo.1（日曜日） 2 表示範囲：$V==1，コメントNo.2（月曜日） 3 表示範囲：$V==2，コメントNo.3（火曜日） 4 表示範囲：$V==3，コメントNo.4（水曜日） 5 表示範囲：$V==4，コメントNo.5（木曜日） 6 表示範囲：$V==5，コメントNo.6（金曜日） 7 表示範囲：$V==6，コメントNo.7（土曜日）

＊ コメント表示で使用するコメントは，コメント作成であらかじめ作成しておきます。

(4) スクリプト設定（約2.5kバイト）

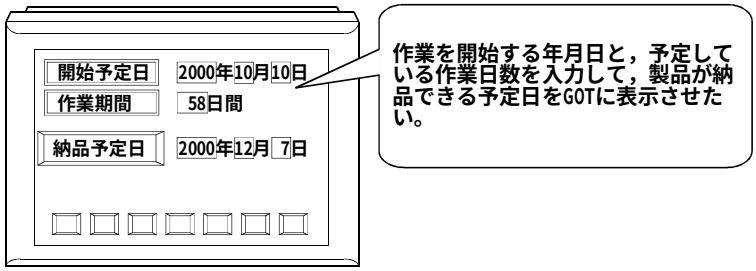
項　目	内　容
種別	画面スクリプト
トリガ	立上り／GB100
データ形式	16ビット／符号なしBIN

スクリプト記述	[w:TMP5]=1900;
	[w:TMP0]=1900;
	[w:TMP1]=0;
	while([w:TMP0]<[w:GD100]){
	　　if(!(([w:TMP0]-1900)%4)){
	　　[w:TMP1]=[w:TMP1]+1;
	　　}
	　　[w:TMP0]=[w:TMP0]+1;
	}
	[w:TMP2]=([w:GD100]-1900)*365+[w:TMP1];
	[w:TMP0]=1;
	[w:TMP3]=0;
	while([w:TMP0]<[w:GD101]){
	　　switch([w:TMP0]){
	　　case 1:
	　　case 3:
	　　case 5:
	　　case 7:
	　　case 8:
	　　case 10:
	　　case 12:
	　　[w:TMP3]=[w:TMP3]+31;
	　　break;
	　　case 2:
	　　if(!(([w:GD100]-[w:TMP5])%4)){
	　　[w:TMP3]=[w:TMP3]+29;
	　　}
	　　else{
	　　[w:TMP3]=[w:TMP3]+28;
	　　}
	　　break;
	　　default:
	　　[w:TMP3]=[w:TMP3]+30;
	　　break;
	　　}
	　　[w:TMP0]=[w:TMP0]+1;
	}
	[w:TMP4]=[w:GD102];
	[w:TMP10]=[w:TMP2]+[w:TMP3]+[w:TMP4]-1;
	[w:GD200] = [w:TMP10]%7;

8.3　指定期間後の年月日算出機能

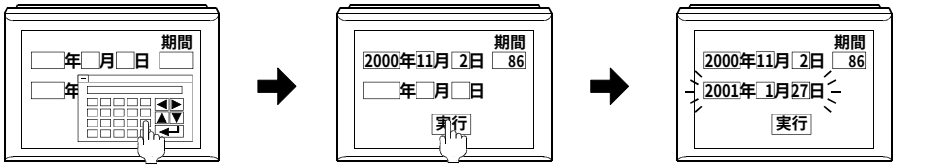
(1) こんな悩み，ありませんか？

予定工数（期間）から，納期を計算したいが，そのような機能は現状のGOTにはない。



(2) スクリプト機能を使用した具体例

年月日と期間（日数）を入力後，実行キーを押すとその期間後の年月日を表示します。



年月日と期間を入力　　実行スイッチをタッチ　　入力した年月日と期間から，期間後の年月日を表示します。

本具体例はGT Designer，GT Designer2に格納されています。
GT Designer：　\Melsec\Gotr\Examples\A975got\09 スクリプト\指定期間後の年月日算出機能
GT Designer2：　\Melsec\GTD2\Example\A975got\09 スクリプト\指定期間後の年月日算出機能

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GB100	スクリプト実行トリガデバイス
GD100	「年」入力用デバイス
GD101	「月」入力用デバイス
GD102	「日」入力用デバイス
GD103	「期間」入力用デバイス
GD200	「年」出力用デバイス
GD201	「月」出力用デバイス
GD202	「日」出力用デバイス
TMP0	カウンタ
TMP1	指定年までに存在する閏年数
TMP2	yyyy/1/1までの通算日
TMP3	yyyy/1/1からyyyy/mm/1までの通算日
TMP4	yyyy/mm/1からyyyy/mm/ddまでの通算日
TMP5	1900年を起点とした場合、最初の閏年

デバイス種別	用　途
TMP10	1900/1/1からの通算日
TMP20	年カウンタ
TMP21	月カウンタ
TMP22	日カウンタ
TMP23	通算日カウンタ
TMP33	終了フラグ

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
数値入力機能	「年」入力用	＜基本設定タブ＞ デバイス：GD100 ＜表示設定タブ＞ 表示桁数：4 ＜範囲設定タブ＞ 表示範囲：1900<=$V
数値入力機能	「月」入力用	＜基本設定タブ＞ デバイス：GD101 ＜表示設定タブ＞ 表示桁数：2 ＜範囲設定タブ＞ 表示範囲：1<=$V<=12
数値入力機能	「日」入力用	＜基本設定タブ＞ デバイス：GD102 ＜表示設定タブ＞ 表示桁数：2 ＜範囲設定タブ＞ 表示範囲：1<=$V<=31
数値入力機能	「期間」入力用	＜基本設定タブ＞ デバイス：GD103 ＜表示設定タブ＞ 表示桁数：5 ＜範囲設定タブ＞ 表示範囲：0<=$V
タッチスイッチ機能	「実行」スイッチ用	＜動作設定タブ＞ [ビット]･･･デバイス：GB100，動作種別：ビットモーメンタリ
数値表示機能	「年」表示用	＜基本設定タブ＞ デバイス：GD200 ＜表示設定タブ＞ 表示桁数：4
数値表示機能	「月」表示用	＜基本設定タブ＞ デバイス：GD201 ＜表示設定タブ＞ 表示桁数：2
数値表示機能	「日」表示用	＜基本設定タブ＞ デバイス：GD202 ＜表示設定タブ＞ 表示桁数：2

(4) スクリプト設定（約7kバイト）

項　目	内　容
種別	画面スクリプト
トリガ	立上り／GB100
データ形式	16ビット／符号なしBIN

スクリプト記述	[w:TMP0005]=1900; //1900年を起点とした場合，最初の閏年をセット
	[w:TMP0000]=1900; //下記ループ処理の初期値
	[w:TMP0001]=0; //指定年までに存在する閏年数を求める
	while([w:TMP0000]<[w:GD100]){ //現年(TMP0000)が閏年ならば
	  if(!(([w:TMP0000]-1900)%4)){
	  [w:TMP0001]=[w:TMP0001]+1; //閏年カウンタをインクリメント
	  }
	  [w:TMP0000]=[w:TMP0000]+1;
	}
	[w:TMP0002]=([w:GD100]-1900)*365+[w:TMP0001]; //①指定年1/1までの通算日を算出し，tmp0002に格納
	[w:TMP0000]=1;
	[w:TMP0003]=0;
	while([w:TMP0000]<[w:GD101]){ //②yyyy/1/1 から yyyy/mm/1までの通算日を算出し，tmp0003に格納
	  switch([w:TMP0000]){
	  case 1:
	  case 3:
	  case 5:
	  case 7:
	  case 8:
	  case 10:
	  case 12:
	  [w:TMP0003]=[w:TMP0003]+31;
	  break;
	  case 2:
	  if(!(([w:GD100]-[w:TMP0005])%4)){ //指定年が閏年の場合
	  [w:TMP0003]=[w:TMP0003]+29;
	  }
	  else{
	  [w:TMP0003]=[w:TMP0003]+28;
	  }
	  break;
	  default:
	  [w:TMP0003]=[w:TMP0003]+30;
	  break;
	  }
	  [w:TMP0000]=[w:TMP0000]+1;
	}
	[w:TMP0004]=[w:GD102]; //③yyyy/mm/1からyyyy/mm/ddまでの通算日を算出し，tmp0004に格納
	[w:TMP0010]=[w:TMP0002]+[w:TMP0003]+[w:TMP0004]-1; //④1900/1/1からyyyy/mm/ddまでの通算日を算出し，tmp0010に格納
	[w:TMP0010]=[w:TMP0010]+[w:GD103]; //⑤指定日後の通算日を算出
	if([w:TMP0010]<0){
	  [w:TMP0010]=0;
	}
	[w:TMP0020]=1900; //期間を加算した通算日より年月日を算出
	[w:TMP0023]=0;
	while(1){ //通算日と通算日カウンタの差が１年未満になるまで年単位で処理
	  if(([w:TMP0020]-1900)%4){ //閏年以外の場合
	  if(([w:TMP0023]+365)<=[w:TMP0010]){
	  [w:TMP0023]=[w:TMP0023]+365;
	  }
	  else{
	  break;
	  }
	  }

＊ 次ページへ続きます。

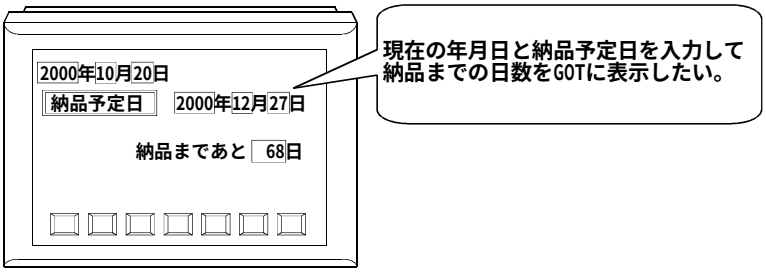
項　目	内　容

スクリプト記述	else{ //閏年の場合
	if(([w:TMP0023]+366)<=[w:TMP0010]){
	[w:TMP0023]=[w:TMP0023]+366;
	}
	else{
	break;
	}
	}
	[w:TMP0020]=[w:TMP0020]+1; //年カウンタをインクリメント
	}
	[w:TMP0033]=0;
	while(1){
	[w:TMP0021]=1;
	while([w:TMP0021]<=12){
	[w:TMP0022]=1;
	while(1){
	if([w:TMP0023]>=[w:TMP0010]){
	[w:GD200]=[w:TMP0020]; //年の出力
	[w:GD201]=[w:TMP0021]; //月の出力
	[w:GD202]=[w:TMP0022]; //日の出力
	[w:TMP0033]=1; //終了フラグON
	break;
	}
	if([w:TMP0022]==28){
	if(([w:TMP0021]==2)&&(([w:TMP0020]-1900)%4)){ //(現:2月)&&(閏年でない)&&(現:28日)
	[w:TMP0023]=[w:TMP0023]+1; //通算日カウンタ調整
	break; //日カウント終了
	}
	}
	if([w:TMP0022]==29){
	if(([w:TMP0021]==2)&&(!(([w:TMP0020]-1900)%4))){ //(現:2月)&&(閏年)&&(現:29日)
	[w:TMP0023]=[w:TMP0023]+1; //通算日カウンタ調整
	break; //日カウント終了
	}
	}
	if([w:TMP0022]==30){
	if(([w:TMP0021]==4)||([w:TMP0021]==6)||([w:TMP0021]==9)||([w:TMP0021]==11)){ //(現:4月or6月or9月or11月)
	[w:TMP0023]=[w:TMP0023]+1; //通算日カウンタ調整
	break; //日カウント終了
	}
	}
	if([w:TMP0022]==31){
	if(([w:TMP0021]==1)||([w:TMP0021]==3)||([w:TMP0021]==5)||([w:TMP0021]==7)||([w:TMP0021]==8)||([w:TMP0021]==10)
	||([w:TMP0021]==12)){ //(現:1月or3月or5月or7月or8月or10月or12月)
	[w:TMP0023]=[w:TMP0023]+1; //通算日カウンタ調整
	break; //日カウント終了
	}
	}
	[w:TMP0022]=[w:TMP0022]+1; //日カウンタインクリメント
	[w:TMP0023]=[w:TMP0023]+1; //通算日カウンタインクリメント
	}
	if([w:TMP0033]==1){
	break;
	}
	[w:TMP0021]=[w:TMP0021]+1; //月カウンタインクリメント
	}
	if([w:TMP0033]==1){
	break;
	}
	[w:TMP0020]=[w:TMP0020]+1; //年カウンタインクリメント
	}

8.4　期間算出機能

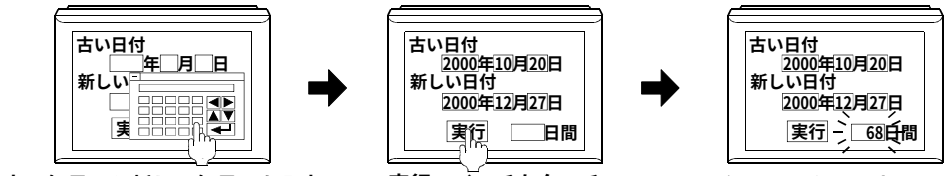
(1) こんな悩み，ありませんか？

2つの年月日を入力し，納品までの残りの日数を算出し，進捗を把握したいが，そのような機能は現状のGOTにはない。



(2) スクリプト機能を使用した具体例

今日の日付（古い日付）と納品の日付（新しい日付）を入力後，実行キーを押すとその期間（日数）を表示します。



古い年月日と新しい年月日を入力　　実行スイッチをタッチ　　2つの年月日の期間を表示します。

本具体例はGT Designer, GT Designer2に格納されています。
GT Designer ： \Melsec\Gotr\Examples\A975got\09 スクリプト\期間算出機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\期間算出機能

応用すればこんなこともできます

今日の日付をシーケンサから自動入力させることにより，あらかじめ設定しておいた納期までの残日数を自動更新表示させることも可能です。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GB100	スクリプト実行トリガデバイス
GD100	古い日付の「年」入力用デバイス
GD101	古い日付の「月」入力用デバイス
GD102	古い日付の「日」入力用デバイス
GD103	新しい日付の「年」入力用デバイス
GD104	新しい日付の「月」入力用デバイス
GD105	新しい日付の「日」入力用デバイス
GD200	期間出力用デバイス

デバイス種別	用　途
TMP0	カウンタ
TMP1	指定年までに存在する閏年数
TMP2	yyyy/1/1までの通算日
TMP3	yyyy/1/1からyyyy/mm/1までの通算日
TMP4	yyyy/mm/1からyyyy/mm/ddまでの通算日
TMP5	1900年を起点とした場合、最初の閏年
TMP10	1900/1/1からの通算日1
TMP11	1900/1/1からの通算日2

(b) 関連オブジェクト設定

オブジェクト種別	用　途	設　定
数値入力機能	古い日付の「年」入力用	＜基本設定タブ＞ デバイス：GD100　16ビット　符号なしBIN ＜表示設定タブ＞ 表示形式：符号なし10進数，表示桁数：4 ＜範囲設定タブ＞ 表示範囲：1900<=$V
数値入力機能	古い日付の「月」入力用	＜基本設定タブ＞ デバイス：GD101　16ビット　符号なしBIN ＜表示設定タブ＞ 表示形式：符号なし10進数，表示桁数：2 ＜範囲設定タブ＞ 表示範囲：1<=$V<=12
数値入力機能	古い日付の「日」入力用	＜基本設定タブ＞ デバイス：GD102　16ビット　符号なしBIN ＜表示設定タブ＞ 表示形式：符号なし10進数，表示桁数：2 ＜範囲設定タブ＞ 表示範囲：1<=$V<=31
数値入力機能	新しい日付の「年」入力用	＜基本設定タブ＞ デバイス：GD103　16ビット　符号なしBIN ＜表示設定タブ＞ 表示形式：符号なし10進数，表示桁数：4 ＜範囲設定タブ＞ 表示範囲：1900<=$W
数値入力機能	新しい日付の「月」入力用	＜基本設定タブ＞ デバイス：GD104　16ビット　符号なしBIN ＜表示設定タブ＞ 表示形式：符号なし10進数，表示桁数：2 ＜範囲設定タブ＞ 表示範囲：1<=$W<=12
数値入力機能	新しい日付の「日」入力用	＜基本設定タブ＞ デバイス：GD105　16ビット　符号なしBIN ＜表示設定タブ＞ 表示形式：符号なし10進数，表示桁数：2 ＜範囲設定タブ＞ 表示範囲：1<=$W<=31
タッチスイッチ機能	「実行」スイッチ用	＜動作設定タブ＞ [ビット]･･･デバイス：GB100，動作種別：ビットモーメンタリ
数値表示機能	期間出力用	＜基本設定タブ＞ デバイス：GD200　16ビット　符号なしBIN ＜表示設定タブ＞ 表示形式：符号なし10進数，表示桁数：5

(4) スクリプト設定（約4.8kバイト）

項　目	内　容
種別	画面スクリプト
トリガ	立上り／GB100
データ形式	16ビット／符号なしBIN

スクリプト記述	[w:TMP0005]=1900; //1900年を起点とした場合，最初の閏年をセット
	[w:TMP0000]=1900; //下記ループ処理の初期値
	[w:TMP0001]=0; //指定年までに存在する閏年数を求める
	while([w:TMP0000]<[w:GD100]){ //現年(TMP0000)が閏年ならば
	  if(!(([w:TMP0000]-1900)%4)){
	  [w:TMP0001]=[w:TMP0001]+1; //閏年カウンタをインクリメント
	  }
	  [w:TMP0000]=[w:TMP0000]+1;
	}
	[w:TMP0002]=([w:GD100]-1900)*365+[w:TMP0001]; //①指定年1/1までの通算日を算出し，tmp0002に格納
	[w:TMP0000]=1;
	[w:TMP0003]=0;
	while([w:TMP0000]<[w:GD101]){ //②yyyy/1/1からyyyy/mm/1までの通算日を算出し，tmp0003に格納
	  switch([w:TMP0000]){
	  case 1:
	  case 3:
	  case 5:
	  case 7:
	  case 8:
	  case 10:
	  case 12:
	  [w:TMP0003]=[w:TMP0003]+31;
	  break;
	  case 2:
	  if(!(([w:GD100]-[w:TMP0005])%4)){ //指定年が閏年の場合
	  [w:TMP0003]=[w:TMP0003]+29;
	  }
	  else{
	  [w:TMP0003]=[w:TMP0003]+28;
	  }
	  break;
	  default:
	  [w:TMP0003]=[w:TMP0003]+30;
	  break;
	  }
	  [w:TMP0000]=[w:TMP0000]+1;
	}
	[w:TMP0004]=[w:GD102]; //③yyyy/mm/1からyyyy/mm/ddまでの通算日を算出し，tmp0004に格納
	[w:TMP0010]=[w:TMP0002]+[w:TMP0003]+[w:TMP0004]-1;
	//④1900/1/1からyyyy/mm/ddまでの通算日を算出し，tmp0010に格納

＊ 次ページへ続きます。

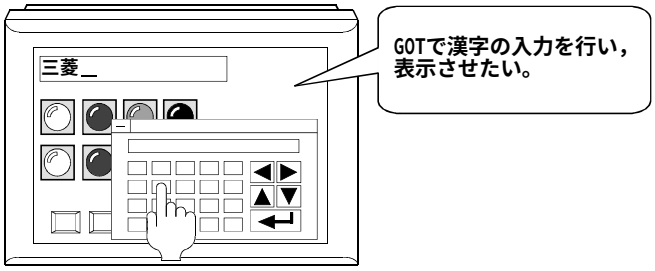
項　目	内　容

スクリプト記述	[w:TMP0000]=1900; //下記ループ処理の初期値
	[w:TMP0001]=0; //指定年までに存在する閏年数を求める
	while([w:TMP0000]<[w:GD103]){ //現年(TMP0000)が閏年ならば
	  if(!(([w:TMP0000]-1900)%4)){
	  [w:TMP0001]=[w:TMP0001]+1; //閏年カウンタをインクリメント
	  }
	  [w:TMP0000]=[w:TMP0000]+1;
	}
	[w:TMP0002]=([w:GD103]-1900)*365+[w:TMP0001]; //①指定年1/1までの通算日を算出し、tmp0002に格納
	[w:TMP0000]=1;
	[w:TMP0003]=0;
	while([w:TMP0000]<[w:GD104]){ //②yyyy/1/1からyyyy/mm/1までの通算日を算出し,tmp0003に格納
	  switch([w:TMP0000]){
	  case 1:
	  case 3:
	  case 5:
	  case 7:
	  case 8:
	  case 10:
	  case 12:
	  [w:TMP0003]=[w:TMP0003]+31;
	  break;
	  case 2:
	  if(!(([w:GD100]-[w:TMP0005])%4)){ //指定年が閏年の場合
	  [w:TMP0003]=[w:TMP0003]+29;
	  }
	  else{
	  [w:TMP0003]=[w:TMP0003]+28;
	  }
	  break;
	  default:
	  [w:TMP0003]=[w:TMP0003]+30;
	  break;
	  }
	  [w:TMP0000]=[w:TMP0000]+1;
	}
	[w:TMP0004]=[w:GD105]; //③yyyy/mm/1からyyyy/mm/ddまでの通算日を算出し，tmp0004に格納
	[w:TMP0011]=[w:TMP0002]+[w:TMP0003]+[w:TMP0004]-1; //④1900/1/1からyyyy/mm/ddまでの通算日を算出し，tmp0011に格納
	[w:GD200]=[w:TMP0011]-[w:TMP0010];

8.5　漢字，英数文字入力機能

(1) こんな悩み，ありませんか

GOT上で漢字入力を行いたいが，そのような機能は現状のGOTにはない。



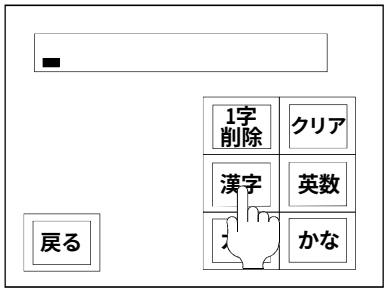
(2) スクリプト機能を使用した具体例

(a) 「漢字」キーを押すと，頭文字選択ウインドウが表示されますので，入力したい漢字の最初の文字を選択します。（音読みした場合の文字です。）
頭文字選択後，候補文字表示ウインドウが表示されますので，ウィンドウ上から入力したい漢字を選択します。

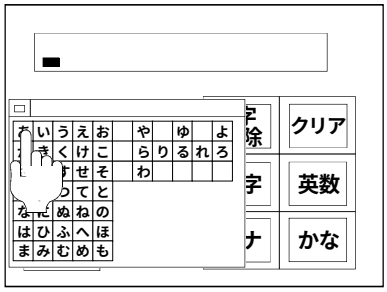
(b) 「かな」／「カナ」／「英数」キーを押すと候補文字表示ウインドウが表示されますので，それぞれのウィンドウ上から入力したい文字を選択します。（頭文字選択ウィンドウは表示しません。）

文字は下記の2通りの方法で選択できます。
・「カーソル移動」キーと「確定」キーを使用する。
・表示文字上を押下して直接入力する。

＜漢字入力時の操作＞



↓



↓

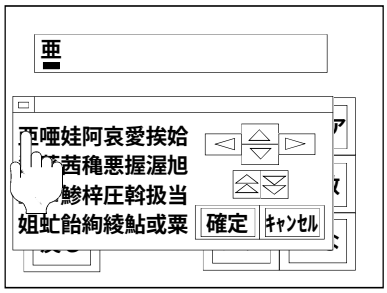
① 漢字入力スイッチをタッチします。

② 頭文字選択ウィンドウが表示されますので，表示したい漢字の頭文字を選択します。

（次ページへ）

（前ページから）

↓



③ 頭文字選択後，漢字入力ウィンドウが表示されますので，入力したい漢字を選択します。

本具体例はGT Designer，GT Designer2に格納されています。
GT Designer2： \Melsec\Gotr\Examples\A975got\09 スクリプト\漢字英数文字入力機能
GT Designer2： \Melsec\GTD2\Example\A975got\09 スクリプト\漢字英数文字入力機能

ポイント

・ 本具体例では，最大15文字まで入力することができます。
・ 本具体例では，シフトJISコードのことを文字コードと記載しています。

(3) 設定例

(a) 使用デバイス

デバイス種別	用　途
GS1.b1	初期化スクリプトトリガ用デバイス
GB100	候補文字表示スクリプトトリガ用デバイス
GB101	カーソル移動操作（↑）スクリプトトリガ用デバイス
GB102	カーソル移動操作（↓）スクリプトトリガ用デバイス
GB103	カーソル移動操作（←）スクリプトトリガ用デバイス
GB104	カーソル移動操作（→）スクリプトトリガ用デバイス
GB106	キャンセルスクリプトトリガ用デバイス
GB108	ページ切替操作（前）スクリプトトリガ用デバイス
GB109	ページ切替操作（次）スクリプトトリガ用デバイス
GB110	文字直接確定スクリプトトリガ用デバイス
GB111	１字削除スクリプトトリガ用デバイス
GB112	クリアスクリプトトリガ用デバイス
GB120	候補文字表示共通スクリプトトリガ用デバイス
GD90	入力エリアのカーソル位置格納用デバイス
GD96	選択文字の基準文字コードとの差分値格納用デバイス
GD99	基準文字コード格納用デバイス
GD100～GD114	入力文字コード格納用デバイス
GD150～GD181	候補文字コード格納用デバイス
GD200	候補文字選択カーソル位置格納用デバイス
TMP2	文字コード変換（上位下位反転）用デバイス
TMP3	初期化（クリア）データ格納用デバイス

(b) 関連オブジェクト設定

① 入力画面

オブジェクト種別	用　途	設　定
アスキー表示機能	入力エリア1～15桁目用	＜基本設定タブ＞ デバイス：GD100～GD114 ＜表示設定タブ＞ 表示桁数：2
部品表示機能	入力カーソル用	＜基本設定タブ＞ デバイス：GD90　ワード(BIN16)，描画モード：XOR ＜表示方法（ワード）タブ＞ 表示範囲：$V==0～$V==14 ＜オプション設定タブ＞ プレビュー番号：1 ＜表示条件タブ＞ トリガ種別：周期　1（秒）
タッチスイッチ機能	「漢字」キー用	＜動作設定タブ＞ [ウィンドウ切換え]･･･切換え先：固定値　106
タッチスイッチ機能	「英数」キー用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99，データ形式：符号なしBIN，動作種別：固定値　33359 [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	「カナ」キー用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99，データ形式：符号なしBIN，動作種別：固定値　33600 [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	「かな」キー用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99，データ形式：符号なしBIN，動作種別：固定値　33439 [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	「1字削除」キー用	＜動作設定タブ＞ [ビット]･･･デバイス：GB111，動作種別：ビットSET
タッチスイッチ機能	「クリア」キー用	＜動作設定タブ＞ [ビット]･･･デバイス：GB112，動作種別：ビットSET

② 頭文字選択ウィンドウ

オブジェクト種別	用　途	設　定
タッチスイッチ機能	頭文字候補キー（あ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　34975 （固定値　34975（0x889F）基準文字コード“亜”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（い）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　35016 （固定値　35016（0x88C8）は基準文字コード“以”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（う）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　35141 （固定値　35141（0x8945）は基準文字コード“右”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（え）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　35168 （固定値　35168（0x8960）は基準文字コード“荏”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11

② 頭文字選択ウィンドウ

オブジェクト種別	用　途	設　定
タッチスイッチ機能	頭文字候補キー（お）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　35223 （固定値　35223（0x8997）は基準文字コード“於”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（か）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　35258 （固定値　35258（0x89BA）は基準文字コード“下”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（き）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　35561 （固定値　35561（0x8AE9）は基準文字コード“企”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（く）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　35811 （固定値　35811（0x8BE3）は基準文字コード“九”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（け）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　35811 （固定値　35924（0x8C54）は基準文字コード“卦”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（こ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　36033 （固定値　36033（0x8CC1）は基準文字コード“乎”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（さ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　36273 （固定値　36273（0x8DB1）基準文字コード“些”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（し）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　36452 （固定値　36452（0x8E64）基準文字コード“仕”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（す）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　36985 （固定値　36985（0x9079）基準文字コード“筒”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（せ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37026 （固定値　37026（0x90A2）基準文字コード“世”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（そ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37219 （固定値　37219（0x9163）基準文字コード“祖”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11

オブジェクト種別	用　途	設　定
タッチスイッチ機能	頭文字候補キー（た）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37308 （固定値　37308（0x91BC）基準文字コード“他”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ち）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37484 （固定値　37484（0x926C）基準文字コード“値”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（つ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37571 （固定値　37571（0x92C3）基準文字コード“津”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（て）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37600 （固定値　37600（0x92E0）基準文字コード“亭”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（と）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37733 （固定値　37733（0x9365）基準文字コード“兎”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（な）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37854 （固定値　37854（0x93DE）基準文字コード“奈”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（に）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37873 （固定値　37873（0x93F1）基準文字コード“二”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ぬ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37959 （固定値　37959（0x9447）基準文字コード“濡”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ね）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37960 （固定値　37960（0x9448）基準文字コード“禰”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（の）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37972 （固定値　37972（0x9454）基準文字コード“乃”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（は）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　37986 （固定値　37986（0x9462）基準文字コード“巴”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11

オブジェクト種別	用　途	設　定
タッチスイッチ機能	頭文字候補キー（ひ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38105 （固定値　38105（0x94D9）基準文字コード“匪”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ふ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38259 （固定値　38259（0x9573）基準文字コード“不”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（へ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38328 （固定値　38328（0x95B8）基準文字コード“丙”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ほ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38363 （固定値　38363（0x95DB）基準文字コード“保”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ま）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38528 （固定値　38528（0x9680）基準文字コード“摩”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（み）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38561 （固定値　38561（0x96A1）基準文字コード“味”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（む）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38577 （固定値　38577（0x96B1）基準文字コード“務”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（め）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38587 （固定値　38587（0x96BB）基準文字コード“冥”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（も）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38604 （固定値　38604（0x96CC）基準文字コード“摸”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（や）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38631 （固定値　38631（0x96E7）基準文字コード“也”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ゆ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38649 （固定値　38649（0x96F9）基準文字コード“愉”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11

オブジェクト種別	用　途	設　定
タッチスイッチ機能	頭文字候補キー（よ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38748 （固定値　38748（0x975C）基準文字コード“予”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ら）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38789 （固定値　38789（0x9785）基準文字コード“羅”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（り）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38808 （固定値　38808（0x9798）基準文字コード“利”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（る）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38874 （固定値　38874（0x97DA）基準文字コード“瑠”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（れ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38879 （固定値　38879（0x97DF）基準文字コード“令”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（ろ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　38979 （固定値　38979（0x9843）基準文字コード“呂”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11
タッチスイッチ機能	頭文字候補キー（わ）用	＜動作設定タブ＞ [ワード]･･･デバイス：GD99　符号なしBIN，動作種別：固定値　39008 （固定値　39008（0x9860）基準文字コード“倭”を示します） [ビット]･･･デバイス：GB100，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　11

③ 候補文字表示ウィンドウ

オブジェクト種別	用　途	設　定
アスキー表示機能	候補文字(1)～(32)表示用	＜基本設定タブ＞ デバイス：GD150～GD181 ＜表示設定タブ＞ 表示桁数：2 ＜オプションタブ＞ オフセット：デバイス　D149
部品表示機能	入力カーソル用	＜基本設定タブ＞ デバイス：GD200　ワード(BIN16)，描画モード：XOR ＜表示方法（ワード）タブ＞ 表示範囲：$V==1～$V==32 ＜オプションタブ＞ プレビュー番号：1 ＜表示条件タブ＞ トリガ種別：周期　1（秒）
タッチスイッチ機能	直接入力用	＜動作設定タブ＞ [ワード]･･･デバイス：GD96，動作種別：固定値　0，256～7936 [ビット]･･･デバイス：GB110，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　0
タッチスイッチ機能	カーソル移動キー用（↑）	＜動作設定タブ＞ [ビット]･･･デバイス：GB101，動作種別：ビットSET
タッチスイッチ機能	カーソル移動キー用（↓）	＜動作設定タブ＞ [ビット]･･･デバイス：GB102，動作種別：ビットSET
タッチスイッチ機能	カーソル移動キー用（←）	＜動作設定タブ＞ [ビット]･･･デバイス：GB103，動作種別：ビットSET
タッチスイッチ機能	カーソル移動キー用（→）	＜動作設定タブ＞ [ビット]･･･デバイス：GB104，動作種別：ビットSET
タッチスイッチ機能	ページ切換えキー用（前ページ）	＜動作設定タブ＞ [ワード]･･･デバイス：GD200，動作種別：固定値　1 [ビット]･･･デバイス：GB108，動作種別：ビットSET
タッチスイッチ機能	ページ切換えキー用（次ページ）	＜動作設定タブ＞ [ワード]･･･デバイス：GD200，動作種別：固定値　1 [ビット]･･･デバイス：GB109，動作種別：ビットSET
タッチスイッチ機能	確定キー用	＜動作設定タブ＞ [ワード]･･･デバイス：GD90，動作種別：固定値　1，間接デバイス：GD90 [ビット]･･･デバイス：GB106，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　0
タッチスイッチ機能	キャンセルキー用	＜動作設定タブ＞ [ビット]･･･デバイス：GB106，動作種別：ビットSET [ウィンドウ切換え]･･･切換え先：固定値　0

(4) スクリプト設定（約40kバイト）

項　目	内　容
種別	プロジェクトスクリプト（初期化）
トリガ	立上り／GS1.b1
データ形式	16ビット／符号なしBIN
スクリプト記述	[w:GD90]=0;　　//入力エリアカーソル位置初期化 set([b:GB112]);　//入力エリアクリアスクリプト起動

項　目	内　容
種別	プロジェクトスクリプト（候補文字表示）
トリガ	立上り／GB100
データ形式	16ビット／符号なしBIN

スクリプト記述	[w:TMP0002] = (([w:GD99] >> 8) & 0xff) + (([w:GD99] << 8) & 0xff00);
	　　//上位下位反転
	switch([w:GD90]){　　//入力エリアのカーソル位置に候補表示
	　case 0:
	　　[w:GD100] = [w:TMP0002];
	　　break;
	　case 1:
	　　[w:GD101] = [w:TMP0002];
	　　break;
	　case 2:
	　　[w:GD102] = [w:TMP0002];
	　　break;
	　case 3:
	　　[w:GD103] = [w:TMP0002];
	　　break;
	　case 4:
	　　[w:GD104] = [w:TMP0002];
	　　break;
	　case 5:
	　　[w:GD105] = [w:TMP0002];
	　　break;
	　case 6:
	　　[w:GD106] = [w:TMP0002];
	　　break;
	　case 7:
	　　[w:GD107] = [w:TMP0002];
	　　break;
	　case 8:
	　　[w:GD108] = [w:TMP0002];
	　　break;
	　case 9:
	　　[w:GD109] = [w:TMP0002];
	　　break;
	　case 10:
	　　[w:GD110] = [w:TMP0002];
	　　break;
	　case 11:
	　　[w:GD111] = [w:TMP0002];
	　　break;
	　case 12:
	　　[w:GD112] = [w:TMP0002];
	　　break;
	　case 13:
	　　[w:GD113] = [w:TMP0002];
	　　break;
	　default:
	　　[w:GD114] = [w:TMP0002];
	　　break;
	}
	[w:GD200]=1;　　//候補文字表示ウインドウカーソル位置初期化
	set([b:GB120]);　　//候補文字表示ー共通を起動

項　目	内　容
種別	プロジェクトスクリプト（カーソル移動キー操作－↑）
トリガ	立上り／GB101
データ形式	16ビット／符号なしBIN

スクリプト記述	if([w:GD200] > 8){　　//カーソル位置が2行目以降のとき
	　switch([w:GD90]){　　//入力エリアのカーソル位置に候補表示
	　　case 0:
	　　[w:GD100] = [w:GD100] - 0x800;
	　　break;
	　　case 1:
	　　[w:GD101] = [w:GD101] - 0x800;
	　　break;
	　　case 2:
	　　[w:GD102] = [w:GD102] - 0x800;
	　　break;
	　　case 3:
	　　[w:GD103] = [w:GD103] - 0x800;
	　　break;
	　　case 4:
	　　[w:GD104] = [w:GD104] - 0x800;
	　　break;
	　　case 5:
	　　[w:GD105] = [w:GD105] - 0x800;
	　　break;
	　　case 6:
	　　[w:GD106] = [w:GD106] - 0x800;
	　　break;
	　　case 7:
	　　[w:GD107] = [w:GD107] - 0x800;
	　　break;
	　　case 8:
	　　[w:GD108] = [w:GD108] - 0x800;
	　　break;
	　　case 9:
	　　[w:GD109] = [w:GD109] - 0x800;
	　　break;
	　　case 10:
	　　[w:GD110] = [w:GD110] - 0x800;
	　　break;
	　　case 11:
	　　[w:GD111] = [w:GD111] - 0x800;
	　　break;
	　　case 12:
	　　[w:GD112] = [w:GD112] - 0x800;
	　　break;
	　　case 13:
	　　[w:GD113] = [w:GD113] - 0x800;
	　　break;
	　　default:
	　　[w:GD114] = [w:GD114] - 0x800;
	　　break;
	　}
	　[w:GD200] = [w:GD200] - 8;　　//カーソル位置を上に移動
	}
	rst[w:GB101];　　//トリガデバイス初期化

項　目	内　容
種別	プロジェクトスクリプト（カーソル移動キー操作－↓）
トリガ	立上り／GB102
データ形式	16ビット／符号なしBIN

スクリプト記述	if([w:GD200] < 24){　　//カーソル位置が最下行以外のとき
	　switch([w:GD90]){　　//入力エリアのカーソル位置に候補表示
	　　case 0:
	　　[w:GD100] = [w:GD100] + 0x800;
	　　break;
	　　case 1:
	　　[w:GD101] = [w:GD101] + 0x800;
	　　break;
	　　case 2:
	　　[w:GD102] = [w:GD102] + 0x800;
	　　break;
	　　case 3:
	　　[w:GD103] = [w:GD103] + 0x800;
	　　break;
	　　case 4:
	　　[w:GD104] = [w:GD104] + 0x800;
	　　break;
	　　case 5:
	　　[w:GD105] = [w:GD105] + 0x800;
	　　break;
	　　case 6:
	　　[w:GD106] = [w:GD106] + 0x800;
	　　break;
	　　case 7:
	　　[w:GD107] = [w:GD107] + 0x800;
	　　break;
	　　case 8:
	　　[w:GD108] = [w:GD108] + 0x800;
	　　break;
	　　case 9:
	　　[w:GD109] = [w:GD109] + 0x800;
	　　break;
	　　case 10:
	　　[w:GD110] = [w:GD110] + 0x800;
	　　break;
	　　case 11:
	　　[w:GD111] = [w:GD111] + 0x800;
	　　break;
	　　case 12:
	　　[w:GD112] = [w:GD112] + 0x800;
	　　break;
	　　case 13:
	　　[w:GD113] = [w:GD113] + 0x800;
	　　break;
	　　default:
	　　[w:GD114] = [w:GD114] + 0x800;
	　　break;
	　}
	　[w:GD200] = [w:GD200] + 8;　　//カーソル位置を下に移動
	}
	rst[w:GB102];　　//トリガデバイス初期化

項　目	内　容
種別	プロジェクトスクリプト（カーソル移動キー操作ー←）
トリガ	立上り／GB103
データ形式	16ビット／符号なしBIN

スクリプト記述	if([w:GD200] > 1){　　//カーソル位置が2以上のとき
	  switch([w:GD90]){　　//入力エリアのカーソル位置に候補表示
	  case 0:
	  [w:GD100] = [w:GD100] - 0x100;
	  break;
	  case 1:
	  [w:GD101] = [w:GD101] - 0x100;
	  break;
	  case 2:
	  [w:GD102] = [w:GD102] - 0x100;
	  break;
	  case 3:
	  [w:GD103] = [w:GD103] - 0x100;
	  break;
	  case 4:
	  [w:GD104] = [w:GD104] - 0x100;
	  break;
	  case 5:
	  [w:GD105] = [w:GD105] - 0x100;
	  break;
	  case 6:
	  [w:GD106] = [w:GD106] - 0x100;
	  break;
	  case 7:
	  [w:GD107] = [w:GD107] - 0x100;
	  break;
	  case 8:
	  [w:GD108] = [w:GD108] - 0x100;
	  break;
	  case 9:
	  [w:GD109] = [w:GD109] - 0x100;
	  break;
	  case 10:
	  [w:GD110] = [w:GD110] - 0x100;
	  break;
	  case 11:
	  [w:GD111] = [w:GD111] - 0x100;
	  break;
	  case 12:
	  [w:GD112] = [w:GD112] - 0x100;
	  break;
	  case 13:
	  [w:GD113] = [w:GD113] - 0x100;
	  break;
	  default:
	  [w:GD114] = [w:GD114] - 0x100;
	  break;
	  }
	  [w:GD200] = [w:GD200] - 1;　　//カーソル位置を前に移動
	}
	rst[w:GB103];　　//トリガデバイス初期化

項　目	内　容
種別	プロジェクトスクリプト（カーソル移動キー操作→）
トリガ	立上り／GB104
データ形式	16ビット／符号なしBIN
スクリプト記述	

```
if([w:GD200] < 32){                          //カーソル位置が32未満のとき
  switch([w:GD90]){                          //入力エリアのカーソル位置に候補表示
    case 0:
      [w:GD100] = [w:GD100] + 0x100;
      break;
    case 1:
      [w:GD101] = [w:GD101] + 0x100;
      break;
    case 2:
      [w:GD102] = [w:GD102] + 0x100;
      break;
    case 3:
      [w:GD103] = [w:GD103] + 0x100;
      break;
    case 4:
      [w:GD104] = [w:GD104] + 0x100;
      break;
    case 5:
      [w:GD105] = [w:GD105] + 0x100;
      break;
    case 6:
      [w:GD106] = [w:GD106] + 0x100;
      break;
    case 7:
      [w:GD107] = [w:GD107] + 0x100;
      break;
    case 8:
      [w:GD108] = [w:GD108] + 0x100;
      break;
    case 9:
      [w:GD109] = [w:GD109] + 0x100;
      break;
    case 10:
      [w:GD110] = [w:GD110] + 0x100;
      break;
    case 11:
      [w:GD111] = [w:GD111] + 0x100;
      break;
    case 12:
      [w:GD112] = [w:GD112] + 0x100;
      break;
    case 13:
      [w:GD113] = [w:GD113] + 0x100;
      break;
    default:
      [w:GD114] = [w:GD114] + 0x100;
      break;
  }
  [w:GD200] = [w:GD200] + 1;                 //カーソル位置を次に移動
}
rst[w:GB104];                                //トリガデバイス初期化
```

項　目	内　容
種別	プロジェクトスクリプト（ページ切換えキー操作ー前）
トリガ	立上り／GB108
データ形式	16ビット／符号なしBIN

スクリプト記述	[w:GD99] = [w:GD99] - 0x20;　　//基準文字コードを変更
	[w:TMP0002] = ((([w:GD99] - 0x20) >> 8) & 0xff) + ((([w:GD99] - 0x20) << 8) & 0xff00);
	　　//上位下位反転
	switch([w:GD90]){　　//入力エリアのカーソル位置に候補表示
	　case 0:
	　　[w:GD100] = [w:TMP0002];
	　　break;
	　case 1:
	　　[w:GD101] = [w:TMP0002];
	　　break;
	　case 2:
	　　[w:GD102] = [w:TMP0002];
	　　break;
	　case 3:
	　　[w:GD103] = [w:TMP0002];
	　　break;
	　case 4:
	　　[w:GD104] = [w:TMP0002];
	　　break;
	　case 5:
	　　[w:GD105] = [w:TMP0002];
	　　break;
	　case 6:
	　　[w:GD106] = [w:TMP0002];
	　　break;
	　case 7:
	　　[w:GD107] = [w:TMP0002];
	　　break;
	　case 8:
	　　[w:GD108] = [w:TMP0002];
	　　break;
	　case 9:
	　　[w:GD109] = [w:TMP0002];
	　　break;
	　case 10:
	　　[w:GD110] = [w:TMP0002];
	　　break;
	　case 11:
	　　[w:GD111] = [w:TMP0002];
	　　break;
	　case 12:
	　　[w:GD112] = [w:TMP0002];
	　　break;
	　case 13:
	　　[w:GD113] = [w:TMP0002];
	　　break;
	　default:
	　　[w:GD114] = [w:TMP0002];
	　　break;
	}
	set([b:GB120]);　　//候補文字表示ー共通を起動

項　目	内　容
種別	プロジェクトスクリプト（ページ切換えキー操作一次）
トリガ	立上り／GB109
データ形式	16ビット／符号なしBIN

スクリプト記述	[w:GD99] = [w:GD99] + 0x20;　　//基準文字コードを変更
	[w:TMP0002] = ((([w:GD99] + 0x20) >> 8) & 0xff) + ((([w:GD99] + 0x20) << 8) & 0xff00);
	　　//上位下位反転
	switch([w:GD90]){　　//入力エリアのカーソル位置に候補表示
	　case 0:
	　　[w:GD100] = [w:TMP0002];
	　　break;
	　case 1:
	　　[w:GD101] = [w:TMP0002];
	　　break;
	　case 2:
	　　[w:GD102] = [w:TMP0002];
	　　break;
	　case 3:
	　　[w:GD103] = [w:TMP0002];
	　　break;
	　case 4:
	　　[w:GD104] = [w:TMP0002];
	　　break;
	　case 5:
	　　[w:GD105] = [w:TMP0002];
	　　break;
	　case 6:
	　　[w:GD106] = [w:TMP0002];
	　　break;
	　case 7:
	　　[w:GD107] = [w:TMP0002];
	　　break;
	　case 8:
	　　[w:GD108] = [w:TMP0002];
	　　break;
	　case 9:
	　　[w:GD109] = [w:TMP0002];
	　　break;
	　case 10:
	　　[w:GD110] = [w:TMP0002];
	　　break;
	　case 11:
	　　[w:GD111] = [w:TMP0002];
	　　break;
	　case 12:
	　　[w:GD112] = [w:TMP0002];
	　　break;
	　case 13:
	　　[w:GD113] = [w:TMP0002];
	　　break;
	　default:
	　　[w:GD114] = [w:TMP0002];
	　　break;
	}
	set([b:GB120]);　　//候補文字表示一共通を起動

項　目	内　容
種別	プロジェクトスクリプト（文字直接確定）
トリガ	立上り／GB110
データ形式	16ビット／符号なしBIN

スクリプト記述	switch([w:GD90]){　　//入力エリアのカーソル位置に候補表示
	  case 0:
	  [w:GD100] = [w:GD150] + [w:GD96];
	  break;
	  case 1:
	  [w:GD101] = [w:GD150] + [w:GD96];
	  break;
	  case 2:
	  [w:GD102] = [w:GD150] + [w:GD96];
	  break;
	  case 3:
	  [w:GD103] = [w:GD150] + [w:GD96];
	  break;
	  case 4:
	  [w:GD104] = [w:GD150] + [w:GD96];
	  break;
	  case 5:
	  [w:GD105] = [w:GD150] + [w:GD96];
	  break;
	  case 6:
	  [w:GD106] = [w:GD150] + [w:GD96];
	  break;
	  case 7:
	  [w:GD107] = [w:GD150] + [w:GD96];
	  break;
	  case 8:
	  [w:GD108] = [w:GD150] + [w:GD96];
	  break;
	  case 9:
	  [w:GD109] = [w:GD150] + [w:GD96];
	  break;
	  case 10:
	  [w:GD110] = [w:GD150] + [w:GD96];
	  break;
	  case 11:
	  [w:GD111] = [w:GD150] + [w:GD96];
	  break;
	  case 12:
	  [w:GD112] = [w:GD150] + [w:GD96];
	  break;
	  case 13:
	  [w:GD113] = [w:GD150] + [w:GD96];
	  break;
	  default:
	  [w:GD114] = [w:GD150] + [w:GD96];
	  break;
	}
	[w:GD90] = [w:GD90] + 1;　　//入力エリアのカーソル位置を右へ移動
	rst[w:GB110];　　//トリガデバイス初期化

項　目	内　容
種別	プロジェクトスクリプト（1文字削除）
トリガ	立上り／GB111
データ形式	16ビット／符号なしBIN

スクリプト記述	if([w:GD0] > 0){　　//入力エリアのカーソル位置が2桁目以降のとき
	[w:GD90] = [w:GD90] - 1;　　//入力エリアのカーソル位置を左へ移動
	switch([w:GD90] - 1){　　//入力エリアのカーソル位置をクリア
	case 0:
	[w:GD100] = 0;
	break;
	case 1:
	[w:GD101] = 0;
	break;
	case 2:
	[w:GD102] = 0;
	break;
	case 3:
	[w:GD103] = 0;
	break;
	case 4:
	[w:GD104] = 0;
	break;
	case 5:
	[w:GD105] = 0;
	break;
	case 6:
	[w:GD106] = 0;
	break;
	case 7:
	[w:GD107] = 0;
	break;
	case 8:
	[w:GD108] = 0;
	break;
	case 9:
	[w:GD109] = 0;
	break;
	case 10:
	[w:GD110] = 0;
	break;
	case 11:
	[w:GD111] = 0;
	break;
	case 12:
	[w:GD112] = 0;
	break;
	case 13:
	[w:GD113] = 0;
	break;
	default:
	[w:GD114] = 0;
	break;
	}
	}
	rst[w:GB111];　　//トリガデバイス初期化

項　目	内　容
種別	プロジェクトスクリプト（クリア）
トリガ	立上り／GB112
データ形式	16ビット／符号なしBIN

スクリプト記述	[w:GD90] = 0; //入力エリアのカーソル位置初期化
	[w:TMP0003] = 0; //入力文字格納デバイスクリア
	fmov([w:TMP0003],[w:GD100],15);
	rst[w:GB112]; //トリガデバイス初期化

項　目	内　容
種別	プロジェクトスクリプト（キャンセル）
トリガ	立上り／GB106
データ形式	16ビット／符号なしBIN

スクリプト記述	switch([w:GD90]){　　//入力エリアのカーソル位置をクリア
	  case 0:
	  [w:GD100] = 0;
	  break;
	  case 1:
	  [w:GD101] = 0;
	  break;
	  case 2:
	  [w:GD102] = 0;
	  break;
	  case 3:
	  [w:GD103] = 0;
	  break;
	  case 4:
	  [w:GD104] = 0;
	  break;
	  case 5:
	  [w:GD105] = 0;
	  break;
	  case 6:
	  [w:GD106] = 0;
	  break;
	  case 7:
	  [w:GD107] = 0;
	  break;
	  case 8:
	  [w:GD108] = 0;
	  break;
	  case 9:
	  [w:GD109] = 0;
	  break;
	  case 10:
	  [w:GD110] = 0;
	  break;
	  case 11:
	  [w:GD111] = 0;
	  break;
	  case 12:
	  [w:GD112] = 0;
	  break;
	  case 13:
	  [w:GD113] = 0;
	  break;
	  default:
	  [w:GD114] = 0;
	  break;
	}
	rst[w:GB106];　　//トリガデバイス初期化

項　目	内　容
種別	プロジェクトスクリプト（候補文字表示）
トリガ	立上り／GB120
データ形式	16ビット／符号なしBIN

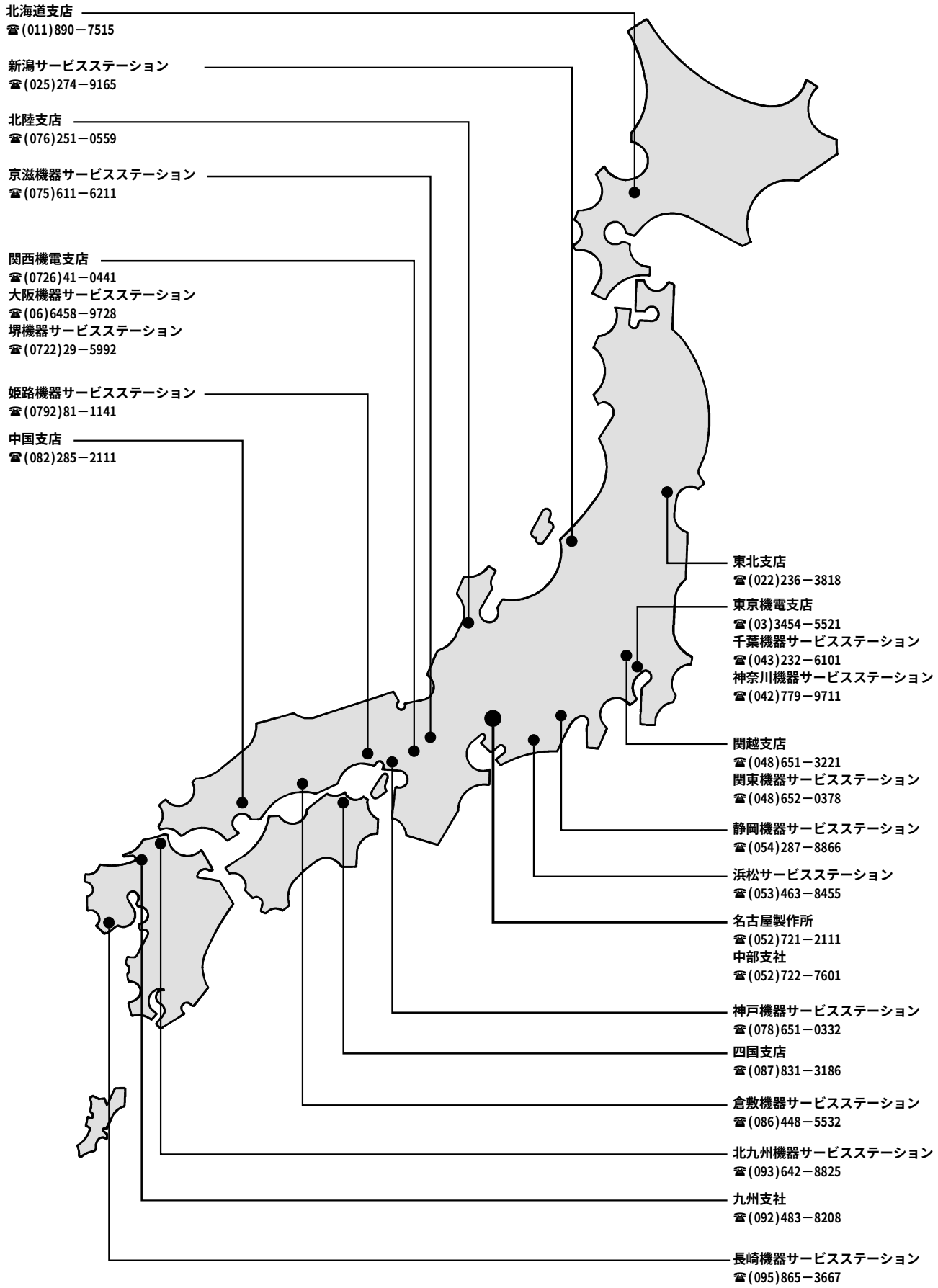
スクリプト記述	[w:GD150] = (([w:GD99] >> 8) & 0xff) + (([w:GD99] << 8) & 0xff00);
	[w:GD151] = ((([w:GD99] + 1) >> 8) & 0xff) + ((([w:GD99] + 1) << 8) & 0xff00);
	[w:GD152] = ((([w:GD99] + 2) >> 8) & 0xff) + ((([w:GD99] + 2) << 8) & 0xff00);
	[w:GD153] = ((([w:GD99] + 3) >> 8) & 0xff) + ((([w:GD99] + 3) << 8) & 0xff00);
	[w:GD154] = ((([w:GD99] + 4) >> 8) & 0xff) + ((([w:GD99] + 4) << 8) & 0xff00);
	[w:GD155] = ((([w:GD99] + 5) >> 8) & 0xff) + ((([w:GD99] + 5) << 8) & 0xff00);
	[w:GD156] = ((([w:GD99] + 6) >> 8) & 0xff) + ((([w:GD99] + 6) << 8) & 0xff00);
	[w:GD157] = ((([w:GD99] + 7) >> 8) & 0xff) + ((([w:GD99] + 7) << 8) & 0xff00);
	[w:GD158] = ((([w:GD99] + 8) >> 8) & 0xff) + ((([w:GD99] + 8) << 8) & 0xff00);
	[w:GD159] = ((([w:GD99] + 9) >> 8) & 0xff) + ((([w:GD99] + 9) << 8) & 0xff00);
	[w:GD160] = ((([w:GD99] + 10) >> 8) & 0xff) + ((([w:GD99] + 10) << 8) & 0xff00);
	[w:GD161] = ((([w:GD99] + 11) >> 8) & 0xff) + ((([w:GD99] + 11) << 8) & 0xff00);
	[w:GD162] = ((([w:GD99] + 12) >> 8) & 0xff) + ((([w:GD99] + 12) << 8) & 0xff00);
	[w:GD163] = ((([w:GD99] + 13) >> 8) & 0xff) + ((([w:GD99] + 13) << 8) & 0xff00);
	[w:GD164] = ((([w:GD99] + 14) >> 8) & 0xff) + ((([w:GD99] + 14) << 8) & 0xff00);
	[w:GD165] = ((([w:GD99] + 15) >> 8) & 0xff) + ((([w:GD99] + 15) << 8) & 0xff00);
	[w:GD166] = ((([w:GD99] + 16) >> 8) & 0xff) + ((([w:GD99] + 16) << 8) & 0xff00);
	[w:GD167] = ((([w:GD99] + 17) >> 8) & 0xff) + ((([w:GD99] + 17) << 8) & 0xff00);
	[w:GD168] = ((([w:GD99] + 18) >> 8) & 0xff) + ((([w:GD99] + 18) << 8) & 0xff00);
	[w:GD169] = ((([w:GD99] + 19) >> 8) & 0xff) + ((([w:GD99] + 19) << 8) & 0xff00);
	[w:GD170] = ((([w:GD99] + 20) >> 8) & 0xff) + ((([w:GD99] + 20) << 8) & 0xff00);
	[w:GD171] = ((([w:GD99] + 21) >> 8) & 0xff) + ((([w:GD99] + 21) << 8) & 0xff00);
	[w:GD172] = ((([w:GD99] + 22) >> 8) & 0xff) + ((([w:GD99] + 22) << 8) & 0xff00);
	[w:GD173] = ((([w:GD99] + 23) >> 8) & 0xff) + ((([w:GD99] + 23) << 8) & 0xff00);
	[w:GD174] = ((([w:GD99] + 24) >> 8) & 0xff) + ((([w:GD99] + 24) << 8) & 0xff00);
	[w:GD175] = ((([w:GD99] + 25) >> 8) & 0xff) + ((([w:GD99] + 25) << 8) & 0xff00);
	[w:GD176] = ((([w:GD99] + 26) >> 8) & 0xff) + ((([w:GD99] + 26) << 8) & 0xff00);
	[w:GD177] = ((([w:GD99] + 27) >> 8) & 0xff) + ((([w:GD99] + 27) << 8) & 0xff00);
	[w:GD178] = ((([w:GD99] + 28) >> 8) & 0xff) + ((([w:GD99] + 28) << 8) & 0xff00);
	[w:GD179] = ((([w:GD99] + 29) >> 8) & 0xff) + ((([w:GD99] + 29) << 8) & 0xff00);
	[w:GD180] = ((([w:GD99] + 30) >> 8) & 0xff) + ((([w:GD99] + 30) << 8) & 0xff00);
	[w:GD181] = ((([w:GD99] + 31) >> 8) & 0xff) + ((([w:GD99] + 31) << 8) & 0xff00);
	//候補文字を格納デバイス32個に設定
	rst([b:GB100]);　//トリガデバイス初期化
	rst([b:GB108]);
	rst([b:GB109]);
	rst([b:GB120]);

8 - 33

メ　モ

8 - 34

サービスネットワーク（三菱電機システムサービス(株)）



北海道支店
☎(011)890－7515

新潟サービスステーション
☎(025)274－9165

北陸支店
☎(076)251－0559

京滋機器サービスステーション
☎(075)611－6211

関西機電支店
☎(0726)41－0441
大阪機器サービスステーション
☎(06)6458－9728
堺機器サービスステーション
☎(0722)29－5992

姫路機器サービスステーション
☎(0792)81－1141

中国支店
☎(082)285－2111

東北支店
☎(022)236－3818

東京機電支店
☎(03)3454－5521
千葉機器サービスステーション
☎(043)232－6101
神奈川機器サービスステーション
☎(042)779－9711

関越支店
☎(048)651－3221
関東機器サービスステーション
☎(048)652－0378

静岡機器サービスステーション
☎(054)287－8866

浜松サービスステーション
☎(053)463－8455

名古屋製作所
☎(052)721－2111
中部支社
☎(052)722－7601

神戸機器サービスステーション
☎(078)651－0332

四国支店
☎(087)831－3186

倉敷機器サービスステーション
☎(086)448－5532

北九州機器サービスステーション
☎(093)642－8825

九州支社
☎(092)483－8208

長崎機器サービスステーション
☎(095)865－3667

GOT-A900シリーズ

スクリプト機能事例集

三菱電機株式会社 〒100-8310 東京都千代田区丸の内2-7-3（東京ビル）

お問い合わせは下記へどうぞ

本社機器営業部 …… 〒100-8310 東京都千代田区丸の内2-7-3（東京ビル） …… (03) 3218-6760
北海道支社 …… 〒060-8693 札幌市中央区北二条西4-1（北海道ビル） …… (011) 212-3794
東北支社 …… 〒980-0011 仙台市青葉区上杉1-17-7（仙台上杉ビル） …… (022) 216-4546
関越支社 …… 〒330-6034 さいたま市中央区新都心11-2（明治安田生命さいたま新都心ビル ランド・アクシス・タワー） … (048) 600-5835
新潟支店 …… 〒950-8504 新潟市中央区東大通2-4-10（日本生命ビル） …… (025) 241-7227
神奈川支社 …… 〒220-8118 横浜市西区みなとみらい2-2-1（横浜ランドマークタワー） …… (045) 224-2624
北陸支社 …… 〒920-0031 金沢市広岡3-1-1（金沢パークビル） …… (076) 233-5502
中部支社 …… 〒450-8522 名古屋市中村区名駅3-28-12（大名古屋ビル） …… (052) 565-3314
豊田支店 …… 〒471-0034 豊田市小坂本町1-5-10（矢作豊田ビル） …… (0565) 34-4112
関西支社 …… 〒530-8206 大阪市北区堂島2-2-2（近鉄堂島ビル） …… (06) 6347-2882
中国支社 …… 〒730-8657 広島市中区中島町3-25（ニッセイ平和公園ビル） …… (082) 248-5445
四国支社 …… 〒760-8654 高松市寿町1-1-8（日本生命高松駅前ビル） …… (087) 825-0055
九州支社 …… 〒810-8686 福岡市中央区天神2-12-1（天神ビル） …… (092) 721-2247

インターネットによる三菱電機FA機器技術情報サービス

MELFANSwebホームページ:http://www.MitsubishiElectric.co.jp/melfansweb
MELFANSwebのFAランドでは、体験版ソフトウェアやソフトウェアアップデートのダウンロードサービス、MELSECシリーズのオンラインマニュアル、Q&Aサービス等がご利用いただけます。FAランドのID登録（無料）が必要です。

電話技術相談窓口

対象機種		電話番号	受付時間※1
GOT表示器	GOT1000/A900シリーズなど	052-712-2417	月曜～金曜 9:00～19:00
	MELSOFT GTシリーズ		
FGOT/DU表示器	GOT-F900シリーズなど	052-725-2271	月曜～木曜 9:00～19:00 金曜 9:00～17:00
MELSEC-FX/F	Fシーケンサ全般		
MELSEC-Q/QnA/A シーケンサ	シーケンサー般（下記以外）	052-711-5111	月曜～金曜 9:00～19:00
	ネットワーク、シリアルコミュニケーションユニット	052-712-2578	
	位置決めユニット ※2	052-712-6607	
	アナログ、温調、温度入力、高速カウンタユニット	052-712-2579	
	C言語コントローラ/MESインタフェースユニット	052-712-2370	月曜～木曜 9:00～19:00 金曜 9:00～17:00
MELSOFTシーケンサプログラミングツール	MELSOFT GXシリーズ	052-711-0037	月曜～金曜 9:00～19:00
	SW□IVD-GPPA/GPPQなど		
MELSOFT通信支援ソフトウェアツール	MELSOFT MXシリーズ	052-712-2370	月曜～木曜 9:00～19:00 金曜 9:00～17:00
	SW□D5F-CSKP/OLEX/XMOPなど		
MELSECパソコンボード	Q80BDシリーズなど		
MELSEC計装/Q二重化	プロセスCPU（Q12/Q25PHCPU）	052-712-2830	
	二重化CPU（Q12/Q25PRHCPU）		
	MELSOFT PXシリーズ		
MELSEC Safety	安全シーケンサ（MELSEC-QSシリーズ）	052-712-3079	
ACサーボ	MELSERVOシリーズ	052-712-6607	月曜～金曜 9:00～19:00
モーションコントローラ	モーションCPU（Q/Aシリーズ）		
	MELSOFT MTシリーズなど		
インバータ	FREQROLシリーズ	052-722-2182	

FAX技術相談窓口

対象機種	FAX番号	受付時間※1
上記対象機種	052-719-6762	9:00～16:00（受信は常時※3）

※1: 土・日・祝祭日、春期・夏期・年末年始の休日を除く通常業務日　※2: ACサーボ、モーション窓口にて対応します
※3: 春期・夏期・年末年始の休日を除く

本マニュアルは、輸出する場合、経済産業省への役務取引許可申請は不要です。

お断りなしに仕様を変更することがありますのでご了承ください。

2002年12月作成

GOT-SCRIPT-JIREI BCN-P5464-B