# COBOL GUI

# 画面構築ガイド

ご注意

(1) 本書の内容の一部または全部を無断転載することは禁止されています。
(2) 本書の内容に関しては将来予告なしに変更することがあります。
(3) 本書は内容について万全を期して作成いたしましたが、万一ご不審な点や誤り、記載もれなどお気付きのことがありましたらご連絡ください。
(4) 運用した結果の影響については、(3)項にかかわらず責任を負いかねますのでご了承ください。





輸出する際の注意事項

本製品 (ソフトウェア)は日本国内仕様であり、外国の規格等には準拠しておりません。
本製品は日本国外で使用された場合、当社は一切責任を負いかねます。また、当社は本製品に関して海外での保守サービスおよび技術サポート等は行っておりません。

# はじめに

本書は、COBOL GUI についての補足説明書です。COBOL GUI は、COBOL 言語の知識を活かし、Windows® 、Windows Server® OS 上の GUI を作成し、COBOL 言語で記述したプログラムと組み合わせてアプリケーションを開発するための製品です。 本書は、COBOL GUI を使用した画面構築の方法について説明するもので、開発ツール View Generator の操作方法や処理の記述については、「COBOL GUI ユーザーズガイド」を参照してください。

なお、第 2 章で使用するサンプルは本書がインストールされたフォルダ内にコピーされます。

サンプルのご使用の際には、添付の README ファイルをよく読んでからご使用願います。

関連説明書としては次のものがあります。

- iew Generator 全般について ：
  「COBOL GUI ユーザーズガイド」
- COBOL プログラムの開発方法 ：
  「COBOL ユーザーズガイド」
- COBOL 言語の文法 ：
  「COBOL 言語説明書」
- COBOL 言語の補足説明 ：
  「COBOL プログラミングの手引」

2015 年　4 月　初 版
2018 年　10 月 第 2 版
2022 年　4 月 第 3 版
2023 年　4 月 第 4 版

# 目　次

# 第1章 画面の設計方針について

この章では、従来からCOBOLでアプリケーションを開発されてこられた方々が、COBOL GUIを使用してGUIアプリケーションを作る際に、推奨する考え方を提案しています。すべての場合に、この考え方が適用できるわけではありませんが、1つの方向性として読んでいただければと考えます。

## 1.1. 一般的なGUIアプリケーションとCOBOL

従来のCOBOLアプリケーションは、画面節(SCREEN SECTION)を使用した画面入出力を行っていたものがほとんどでした。それらアプリケーションを Windows 上に移行する場合、GUI 化するという要件が出てくる場合が多いと思われます。

ところで、Windows上でアプリケーションをGUI化するには、大抵の類似製品がそうであるように、画面(フォーム)上にプッシュボタンやリストボックスなどのGUIコントロールを貼り、そのGUIコントロールに対する各種イベント毎に処理を記述する、イベント駆動型のプログラミングが求められます。

しかし、イベント駆動型のプログラミングを行うということは、従来のCOBOLアプリケーションのファイル入出力部分などが構造化されたプログラムであれば、一部活用できるとはいえ、ほとんどをプログラミングしなおすことになり、従来の COBOL アプリケーションの移行性が大きく損なわれます。そればかりか、イベント駆動型になった結果、各イベント毎に記述した処理の順序性を確認するための処理が必要になるなど、従来の COBOL アプリケーションのプログラミング方法とは考え方も変える必要が出てきます。

## 1.2. COBOL GUIでのGUIアプリケーション

このように、Windows 上でアプリケーションを GUI 化するというのは、一般的には単純な作業ではありません。そこで、COBOL GUIでは、大きく分けて2つの形態のGUIアプリケーションを構築できるようにしています。

(1) 一般的なGUIアプリケーションを構築する機能

画面にプッシュボタンやリストボックスなどの GUI コントロールを貼り、そのコントロールに対する各種イベント毎に処理を記述し、GUI アプリケーションを開発する形態。

(2) 既存資産をそのままGUIアプリケーションを構築する機能(画面節移行機能)

従来のアプリケーションで使用していた画面節(SCREEN SECTION)をほぼそのままGUI化する形態。ただし、この場合のアプリケーションで使われるコントロールは限られますが、プッシュボタンやリストボックスなど GUI コントロールを追加することもできます。

## 1.3. 提案する開発形態

COBOL GUI で提供している2つの形態に、COBOL GUI を使用せず単純移行した CUI アプリケーションの形態も含めて、以下のようにアプリケーションを分類し、適した形態を選択することを提案します。

(1) 頻繁に使用し、画面の見栄えを重視したいアプリケーション

画面設計を含めて、アプリケーションを構築するためにより多くの開発期間を要しますが、一般的な GUI アプリケーションを構築します。

(2) 頻繁に使用するわけではないが、画面の見栄えを重視したいアプリケーション

画面節移行機能を使用して、従来のアプリケーションを活用し、その上で必要に応じて、GUI コントロールを付加または差し替えるなどして、GUI アプリケーションを構築します。

(3) 画面の見栄えより、入力処理のスピードや従来からのキー操作の互換性を重視したいアプリケーション

GUI アプリケーション化せず、従来の CUI のままで、COBOL アプリケーションカスタマイザでキー操作の互換性を保ちます。開発工数も少なくてすみます。

いずれも、エンドユーザの要求する操作性を考慮して画面デザインを行うことが重要です。

# 第2章 アプリケーション概観

## 2.1. アプリケーションの処理の流れ

View Generator を使用して作成したアプリケーションは、通常の COBOL プログラムと同じように COBOL のメインプログラムの先頭から処理を開始します。(＊「初期化」イベントから「外部モジュール呼び出し」手続きを実行した場合は、ここで呼び出された COBOL サブプログラムが最初に実行されます。)

この時すでにメインウィンドウはオープン状態となりますが、COBOL メインプログラム側で処理を実行している間は、コントロールに対する操作はできません。

コントロールに対する操作を可能にするには、COBOL メインプログラム側から以下のいずれかのシステムサブルーチンを使用します。

| | |
|---|---|
| CBL_WINDOW_EXECUTE | ウィンドウ処理実行システムサブルーチン |
| CBL_WINDOW_EVENT | ユーザ定義イベント発行システムサブルーチン |
| CBL_WINDOW_OPEN | サブウィンドウのオープンシステムサブルーチン |
| CBL_DIALOG_OPEN | ダイアログボックスのオープンシステムサブルーチン |

これらのシステムサブルーチンの呼び出しが成功すると、コントロールに対する操作が可能になります。

COBOL メインプログラム側の処理は、システムサブルーチンの呼び出し後、停止状態となります。

COBOL メインプログラム側の処理の再開は、「オブジェクト復帰」手続きを実行することにより行います。

例:

COBOL メインプログラム

```
PROCEDURE DIVISION.
MAINLOOP.
初期処理の記述等                    ←メインウィンドウはオープンされて
        :                               いるが、コントロールの操作は不可
        :
  CALL “CBL_WINDOW_EXECUTE” USING ～  . ←ここでコントロール操作を可能にする
   EVALUATE  RECCODE  OF  CWWINCOM      ←ここへ「オブジェクト復帰」手続き実
                                            行により処理が戻る
   WHEN  PUSHBUTTON1
     プッシュボタン１押下イベントに対する処理
```

手続き記述

```
【ボタン押下(BUTTON1)】//プッシュボタン 1 を押した
 オブジェクト復帰(PUSHBUTTON1)
【ボタン押下(BUTTON2)】//プッシュボタン 2 を押した
 オブジェクト復帰(PUSHBUTTON2)
```

上述の COBOL プログラムにおいて、「RECCODE OF CWWINCOM」というデータ項目を参照していますが、このデータ項目には、メインプログラムに処理が戻る際に「オブジェクト復帰」手続きに指定した引数の値が設定されます。この値を参照することにより、メインプログラムに処理を戻した後、発生したイベントによって処理を振り分けることが可能です。

「オブジェクト復帰」手続きに指定する引数は、手続きエディタのシンボル登録で登録を行います。

生成作業を行うと自動的に名前付き定数(レベル番号 78)としてコントロール用登録集原文内に定義されるので、これをメインプログラム側に取り込み、メインプログラム側からシンボル名で参照することができます。

## 2.2. ウィンドウ側の制御の流れについて

COBOL メインプログラムからシステムサブルーチンを使用し、コントロールに対する操作を可能にした後の制御については、エンドユーザの操作に一任されます。

エンドユーザの操作によって発生したイベントにより、発生したイベントに対応する手続き処理が実行されます。対応する手続き処理がひととおり実行されると、「オブジェクト復帰」手続きにより処理を COBOL メインプログラム側に戻さない限り、再度エンドユーザの操作待ち状態となります。

イベントが発生しても手続きエディタでそのイベントに対応する手続き処理を指定しない場合、処理はなにも実行されません。

## 2.3. 画面節移行機能を使用した場合の入出力について

画面節移行機能を使用すると、COBOL 画面制御機能(SCREEN SECTION)を使用して記述した入出力項目がコントロール(画面節コントロールと呼びます)としてウィンドウに貼り付けられます。他のコントロールとは異なり、画面節コントロールに対する入出力処理は従来通り、ACCEPT/DISPLAY 文で行います。

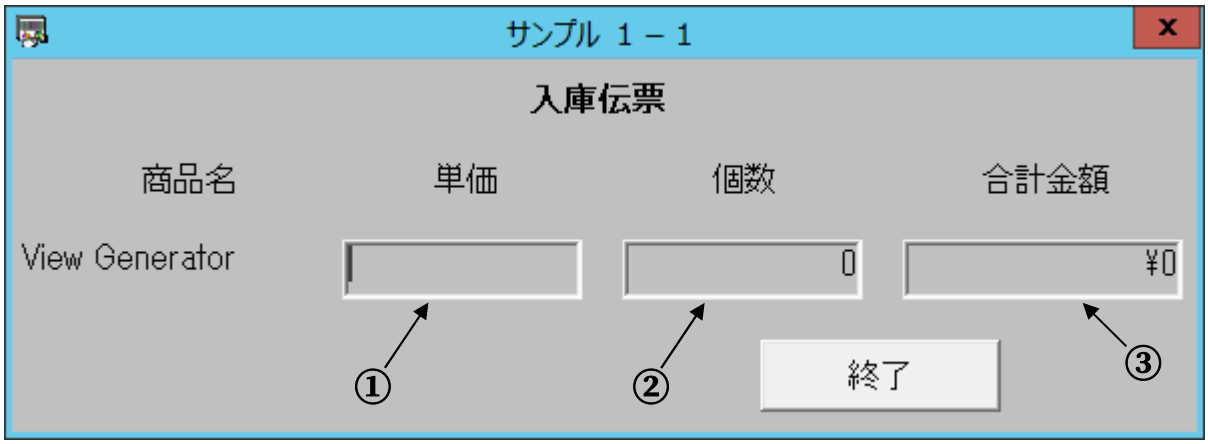
COBOL メインプログラム側で処理を行っている時のみ、画面節コントロールに対する入出力処理が可能です。

## 2.4. 手続きからの COBOL サブプログラムの呼び出し

「外部モジュール呼び出し」により、COBOL サブプログラムを呼び出すことが可能です。

例えば以下の例では、単価と個数を入力し、その入力データから総額を計算しています。

例:




① COBOL データ名 … TANKA
② COBOL データ名 … KOSU
③ COBOL データ名 … KINGAKU、コントロール識別名…WIN001_CS_GOUKEI

COBOL サブプログラム

```
IDENTIFICATION      DIVISION.
PROGRAM-ID.         SMPL1SUB.          ←プログラム名を
                                         「外部モジュール呼び出し」の引数に
                                         指定する。
                      :
DATA                DIVISION.
WORKING-STORAGE     SECTION.
    COPY            SMPL01_CP1 .
LINKAGE SECTION.
01  DUMMY_PARAM     USAGE COMP-1.
PROCEDURE DIVISION  USING  DUMMY_PARAM.
HAJIME.
*---合計金額の計算（単価×個数）
    COMPUTE  STATIC_DATA  OF  KINGAKU
          = EDIT_DATA  OF  TANKA  *  EDIT_DATA  OF  KOSU.
OWARI.
    EXIT  PROGRAM.
```

手続き記述

```
【フォーカス喪失(WIN001_CE_KOSU)】
/-- 「合計金額の計算処理」呼び出し --/
外部モジュール呼び出し("SMPL1SUB",DUMMY)
表示更新(WIN001_CS_GOUKEI)
```

＊引数「DUMMY」は COBOL サブプログラムに渡されます。 渡されたデータは LINKAGE SECTION に記述した一意名で COBOL サブプログラム中で参照することができます。

## 2.5 リストボックス操作時のCOBOL サブプログラム呼び出し

リストボックスやコンボボックスに対して連続メンバ追加や連続メンバ取り込み等の操作を行う場合、COBOL サブプログラムが必要になります。

リスト操作系の手続き命令のうち、引数に外部モジュール名を指定するものがあります。この外部モジュール名にCOBOL サブプログラムを指定します。

以下に「連続メンバ追加」手続きを使用した例を示します。

例:

COBOL サブプログラム

```
 IDENTIFICATION      DIVISION.
 PROGRAM-ID.         SMPL11SUB.                          ①
 ENVIRONMENT         DIVISION.
  CONFIGURATION      SECTION.
  SOURCE-COMPUTER.   PC-9821AP2C9W.
  OBJECT-COMPUTER.   PC-9821AP2C9W
      WITH RETURN VALUE.                                 ②
  INPUT-OUTPUT       SECTION.
  FILE-CONTROL.
      SELECT 商品マスタ ASSIGN TO DATAFILE-MSD
       FILE STATUS FSTS.
  DATA               DIVISION.
  FILE               SECTION.
  FD  商品マスタ LABEL RECORD IS STANDARD
      VALUE OF IDENTIFICATION IS "SYOHIN".
  01  商品           PIC  X(20).
 *
  WORKING-STORAGE    SECTION.
      COPY           SMPL11_CP1 .
  01  FSTS           PIC X(02).       *> FILE STATUS
  01  EODC           PIC X(01).       *> EOD FLAG
  88  EOD            VALUE "1".
  LINKAGE SECTION.
  01  PARAM_FNC      USAGE COMP-1.                      ③
  PROCEDURE DIVISION  USING PARAM_FNC.
```

```
HAJIME SECTION.
SUBMAIN-PROC.
     EVALUATE  PARAM_FNC
*--初期化(ファイルのオープン)
       WHEN  FNC_OPEN
           OPEN INPUT 商品マスタ
           MOVE ZERO  TO EODC
*--初期化(リストボックス作成)                                    ④
       WHEN  FNC_CREATE
           PERFORM READ-PROC THRU READ-PROC-EXIT
           IF NOT EOD
              THEN MOVE CREATE_CONTINUE TO RETURN-CODE     ⑤
                 MOVE 商品       TO LIST_DATA OF LB_GOODS
              ELSE MOVE CREATE_END TO RETURN-CODE          ⑥
                 MOVE ZERO       TO LIST_INDEX OF LB_GOODS
                 CLOSE 商品マスタ
           END-IF
                    :
                    :
     END-EVALUATE.
 OWARI.
     EXIT PROGRAM.

*-----------------
*  順読み込み処理
*-----------------
 READ-PROC.
     READ 商品マスタ NEXT
       AT  END  MOVE "1" TO EODC
     END-READ.
 READ-PROC-EXIT.
     EXIT.
```

手続き記述

```
【初期化】 /-- ﾘｽﾄﾎﾞｯｸｽの作成 --/
       外部モジュール呼び出し("SMPL11SUB",FNC_OPEN)
       連続メンバ追加(DLG000_LB_GOODS,"SMPL11SUB",FNC_CREATE,CREATE_CONTINUE)
```

「連続メンバ追加」手続きは、インタフェース領域に設定された値をリストボックスに追加します。

引数4で指定した値が返却されている間は、引数2に指定したCOBOLサブプログラムを繰り返し呼び出します。なお、一度の呼び出しで、追加されるデータは1件づつであることに注意してください。

引数の内容は以下のとおりです。

引数1…対象となるコントロール識別名。
ここではリストボックスの識別名DLG000_LB_GOODSです。

引数2…呼び出すCOBOLサブプログラムのプログラムID名。ここではSMPL11SUB。
記述例のCOBOLサブプログラム①の部分に対応します。

引数3…呼び出すCOBOLサブプログラムに渡す引数。手続きエディタのシンボル登録で登録を行います。
記述例のCOBOLサブプログラム③に記述したように、LINKAGE SECTIONで定義したCOBOLデータ名を使用し、記述例のCOBOLサブプログラム④のように参照を行います。

引数4…COBOLサブプログラムから返却される値。手続きエディタのシンボル登録で登録を行います。
ここで記述した値が返却される限り、メンバ追加処理を繰り返します。COBOLサブプログラムから値を返却するには特殊レジスタ「RETURN-CODE」を使用します。記述例のCOBOLサブプログラム②にあるように、実行用計算機段落に「WITH RETURN VALUE」句を記述します。これにより特殊レジスタ「RETURN-CODE」が使用可能となります。この例ではシンボル値「CREATE-CONTINUE」が返却される間、メンバ追加処理を繰り返し、「CREATE-CONTINUE」以外の値（ここでは「CREATE-END」）が返却されるとメンバ追加処理を終了します。記述例のCOBOLサブプログラム⑤、⑥の部分で特殊レジスタ「RETURN-CODE」に戻り値を設定しています。

# 第3章 CUI 画面の移行

以下に、COBOL 画面制御機能(SCREEN SECTION)を使用したプログラムを View Generator の機能を使用して GUI に移行する例を示します。
本サンプルには画面節移行ファイルおよび画面情報ファイルが付属していますが、各々の作成方法に関しては、COBOL GUI ユーザーズガイド「第 10 章 画面制御機能の移行」を参照してください。
元となるサンプルプログラムの画面および動作は以下のとおりです。

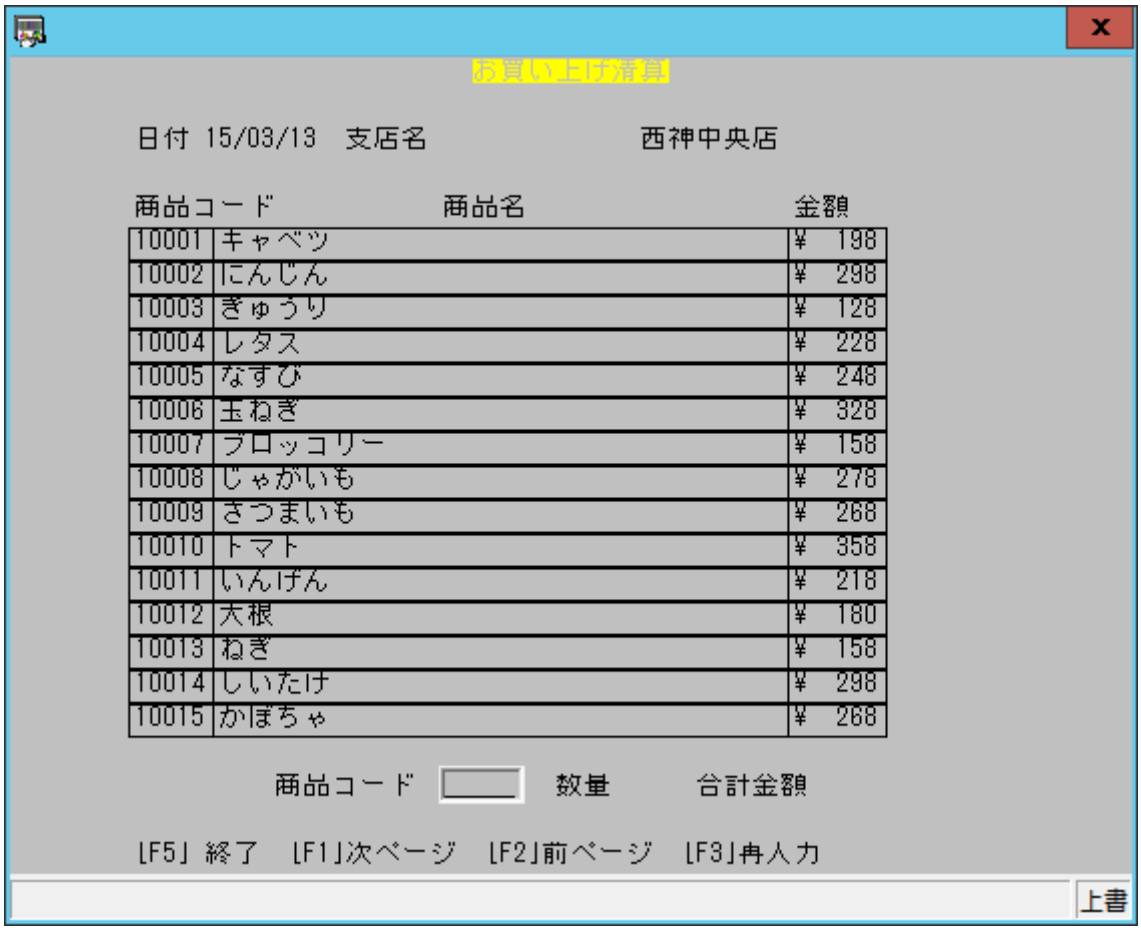

索引ファイルからデータを読み込み、リスト形式に商品一覧を表示します。
商品コードと数量を入力すると合計金額が表示されます。
商品コード入力時に
　[F5] キーを押すとアプリケーションは終了します。
　[F1] キーを押すと次の商品一覧を表示します。
　[F2] キーを押すと直前に表示していた商品一覧を表示します。
数量入力時に
　[F3] キーを押すと商品コードを再入力します。

## 3.1. 画面節移行機能

まず、画面節移行能を使用して、単純なGUI画面を作成します。

画面節移行機能を使用すると COBOL プログラムに修正を加えること無く、以下のような GUI 画面が作成されます[1]。

[1] 但し、移行時に画面節ウィンドウの属性定義で「ウィンドウ固有属性」の背景色と「線」の扱いを既定値から変更してあります


お買い上げ明細
日付 15/03/13 支店名 西神中央店
商品コード 商品名 金額
10001 キャベツ ¥ 198
10002 にんじん ¥ 298
10003 きゅうり ¥ 128
10004 レタス ¥ 228
10005 なすび ¥ 248
10006 玉ねぎ ¥ 328
10007 ブロッコリー ¥ 158
10008 じゃがいも ¥ 278
10009 さつまいも ¥ 268
10010 トマト ¥ 358
10011 いんげん ¥ 218
10012 大根 ¥ 180
10013 ねぎ ¥ 158
10014 しいたけ ¥ 298
10015 かぼちゃ ¥ 268
商品コード 数量 合計金額
[F5] 終了 [F1]次ページ [F2]前ページ [F3]再入力
上書

この画面を元に、商品一覧をリストボックスに変更し、ファンクションキーをプッシュボタンに変更したものが以下の画面です。


リストボックスに表示された商品一覧から商品選択を行い数量を入力し、実行ボタンを押すと合計金額を表示します。実行ボタンを押さずに商品再選択ボタンを押すと、ふたたび商品選択処理となります。

単純移行したGUI画面からの変更は以下の手順で行います。

1. COBOL プログラムのうち、今回不要となる商品コード入力欄および、ファンクションキーのガイド表示を SCREEN SECTION から削除します。
2. COBOL 開発環境を使用して画面節移行ファイルを作成し直し、View Generator で読み込んだのちリストボックスとプッシュボタンの追加、および手続きを記述し、生成作業を行います。
3. COBOL プログラムを修正します。

以下のコーディングは、元になるプログラムにおいて、商品一覧の表示を行って商品コードを入力する部分です。

```
001390     MOVE      IX   TO      WIX IY.
001400     MOVE      0    TO      IX.
001410     MOVE      5    TO      W-LINE.
001420     PERFORM LIST-DISP-RTN THRU LIST-DISP-RTN-END
001430       VARYING COUNTER FROM 1 BY 1
001440         UNTIL IX = WIX  OR COUNTER = 16.
001450     MOVE      1    TO      PAGECOUNTER.
001460 INPUT-URI-CODE.
001470     MOVE   ZERO   TO   W-URICODE.
001480     ACCEPT URI-CODE .
001490     IF ESTS = KEY-F5
001500          GO       TO   PROG-END.
001510     IF ESTS = KEY-F3
001520          GO       TO   INPUT-URI-CODE.
001530     IF ESTS = KEY-F1 AND WIX > PAGECOUNTER * 15
001540       THEN
001550          COMPUTE  IY  =  WIX - (PAGECOUNTER * 15)
001560          COMPUTE  IX  =  PAGECOUNTER * 15
001570          ADD      1    TO    PAGECOUNTER
001580          MOVE     5    TO    W-LINE
001590          DISPLAY  CLEAR-SYODATA
001600          PERFORM LIST-DISP-RTN THRU LIST-DISP-RTN-END
001610            VARYING COUNTER FROM 1 BY 1
001620              UNTIL IY  = 1 OR COUNTER = 16
001630          GO      TO    INPUT-URI-CODE
001640     END-IF.
001650     IF ESTS = KEY-F2 AND PAGECOUNTER > 1
001660       THEN
001670          COMPUTE   IX =  (PAGECOUNTER - 2) * 15
001680          MOVE      WIX   TO      IY
001690            *>  IYはここでは使用しないが、
001700            *>  PERFORM内で使用しているので不正にならない値を設定
001710          SUBTRACT  1     FROM    PAGECOUNTER
001720          MOVE      5     TO      W-LINE
001730          DISPLAY  CLEAR-SYODATA
001740          PERFORM LIST-DISP-RTN THRU LIST-DISP-RTN-END
001750            VARYING COUNTER FROM 1 BY 1
001760              UNTIL COUNTER = 16
001770          GO      TO    INPUT-URI-CODE
001780     END-IF.
001790
001800     IF  W-URICODE = ZERO
001810          GO      TO    INPUT-URI-CODE.
```

```
002160 LIST-DISP-RTN .
002170     COMPUTE W-LINE  = W-LINE + 1.
002180     COMPUTE IX = IX + 1.
002190     DISPLAY SYOHIN-DATA .
002200     SUBTRACT   1       FROM IY.
002210 LIST-DISP-RTN-END .
002220     EXIT.
```

商品一覧はリストボックスに表示し、商品コードはそこから選択するように、以下の様に変更します。

COBOL メインプログラム:

```
001490     MOVE      1     TO     STATE OF SYOHINLIST_BTOK.
001500     MOVE      1     TO     STATE OF SYOHINLIST_BTAGAIN.
001510     CALL      "CBL_CONTROL_ENABLE" USING SYOHINLIST_ENABLE.
```

まだ商品一覧が表示されていないため、実行ボタンと商品再選択ボタンを無効化しておきます。

```
001520     MOVE      IX    TO      WIX.
001530     MOVE      LIST_ADD     TO  USEREVENT OF SYOHINLIST_EVENT.
001540     CALL      "CBL_WINDOW_EVENT" USING  SYOHINLIST_EVENT.
```

リストボックスに商品の一覧を追加するには、リストボックスに関連付けられた手続きを実行します。手続きはウィンドウ・コントロールに対するイベント発生時に実行されるため、COBOL メインプログラムよりイベントを発行し強制的にイベントを発生させます。

```
001550     IF   RECCODE OF CWWINCOM  = SEL_END
001560          GO        TO       INPUT-URI-KOSU.
001570     IF   RECCODE OF CWWINCOM  = PROGRAM_END
001580          GO        TO       PROG-END.
```

リストボックスへ商品一覧を追加し、商品選択を行うと COBOL メインプログラムに制御が移りますので、「オブジェクト復帰」手続きに指定した引数の値によって処理を切り分けます。

```
001590 INPUT-URI-KOSU.
001600     MOVE   ZERO   TO   W-URIKOSU.
001610                ACCEPT URI-KOSU .
001620     MOVE      2     TO     STATE OF SYOHINLIST_BTOK.
001630     MOVE      2     TO     STATE OF SYOHINLIST_BTAGAIN.
001640     CALL      "CBL_CONTROL_ENABLE" USING SYOHINLIST_ENABLE.
```

個数を入力した後、コントロールの有効化／無効化システムサブルーチンを呼び出して「実行」プッシュボタンと「商品再選択」プッシュボタンを有効化します。

```
001650     CALL      "CBL_WINDOW_EXECUTE" USING EXEC_STATUS.
```

また、コントロール操作を可能にするために、ウィンドウ処理実行システムサブルーチンを呼び出します。いずれかのプッシュボタンを押すと COBOL メインプログラムに制御が移ります。

```
001660     IF   RECCODE OF CWWINCOM  = PROGRAM_END
001670          GO       TO   PROG-END.
001680     IF   RECCODE OF CWWINCOM  = INPUT_OK
001690          GO       TO   DISP-GOUKEI.
001700     IF   RECCODE OF CWWINCOM  = LIST_SELECT
001710          GO       TO   LIST-SEL.
```

「オブジェクト復帰」手続きに指定した引数の値によって処理を切り分けます。

```
001740 DISP-GOUKEI.
001750     MOVE     LIST_DATA OF SYO-LIST  TO    W-LISTDATA   .
001760     MOVE     W-CODE     TO    SCODE-KEY.
001770     READ     SYOHIN-F.
001780     COMPUTE  W-GOUKEI = W-GOUKEI + ( SYO-KINGAKU * W-URIKOSU ).
001790     DISPLAY  URI-GOUKEI.
```

合計金額を計算し表示します。

COBOL サブプログラム:

```
000010 IDENTIFICATION              DIVISION.
000020 PROGRAM-ID.                 SMPLGUIS.
000070 ENVIRONMENT                 DIVISION.
                    :
000240 DATA                        DIVISION.
                    :
000360 WORKING-STORAGE             SECTION.
000370 01  FSTS                   PIC X(02).
000380 01  WIX                    PIC   9(02) EXTERNAL.
000390 01  I                      PIC   9(02) VALUE ZERO.
000400 01  W-SYOHINDATA EXTERNAL.
000410     02  W-SYOHIN           OCCURS  60.
000420       03 W-SYOCODE         PIC  9(05).
000430       03 W-SYONAME         PIC  N(20).
000440       03 W-SYOKINGAKU      PIC  9(04).
```

データ項目『WIX』、『W-SYOHINDATA』は COBOL メインプログラム内で、ファイルから読み込んだ商品の件数および内容が設定されています。このデータをサブプログラム内でも参照できるように、EXTERNAL 句を指定します[2]。

[2] COBOL では COBOL85/COBOL85 Pro とは EXTERNAL 句の扱いが変わっており、任意のオブジェクトモジュールで明示的にデータを持たせる必要があります(本サンプルではメインプログラム側)。指定は翻訳時オプション "-Sx "および "-Si "にて行います。詳しくは「COBOL プログラミングの手引」1.4.2『翻訳用コマンドとその使用法』をご覧ください。

```
000450 01  W-LISTDATA.
000460     02 W-CODE          PIC  9(05).
000470     02 FILLER          PIC  X(03) VALUE SPACE.
000480     02 W-NAME          PIC  N(20).
000490     02 FILLER          PIC  X(03) VALUE SPACE.
000500     02 EN              PIC  X(01) VALUE "¥".
000510     02 W-KINGAKU       PIC  9(04).
000520     COPY     SMPLGUI1_CP1 .
000530 LINKAGE                    SECTION.
000540 01  PARAM                  USAGE COMP-1.
000570 PROCEDURE                  DIVISION USING PARAM.
000580 PROG-RTN.
000590     EVALUATE PARAM
```

手続き命令からの呼び出しにより、渡されたパラメータの値によって処理を切り分けます。

```
000600     WHEN  LIST_INIT
000610           MOVE    1        TO     I
000620     WHEN  DATA_ADD
000630       IF  I NOT = WIX
000640       THEN
000650           MOVE    W-SYOCODE(I)     TO     W-CODE
000660           MOVE    W-SYONAME(I)     TO     W-NAME
000670           MOVE    W-SYOKINGAKU(I)  TO     W-KINGAKU
000680           MOVE    W-LISTDATA       TO     LIST_DATA OF SYO-LIST
```

リストボックスのインタフェース領域に、リストボックスに追加するデータを設定します。

```
000690           ADD     1      TO       I
000700           MOVE    ADD_CONTINUE     TO    RETURN-CODE
000710           GO      TO       OWARI
000720       ELSE
000730           MOVE    ADD_END          TO    RETURN-CODE
```

連続メンバ追加処理を続行する場合にはシンボル値「ADD_CONTINUE」を、終了するには「ADD_END」を手続き命令に返却します。

```
000740           GO      TO       OWARI
000750       END-IF
000760     WHEN  PROGRAM_END
000770           CLOSE   SYOHIN-F
000780     END-EVALUATE.
000790 OWARI.
000800     EXIT PROGRAM.
```

切り分け処理を終了します。

手続き記述

```
【ユーザ定義(LIST_ADD)】
        外部モジュール呼び出し("SMPLGUIS",LIST_INIT)
        連続メンバ追加(SYOHINLIST_LB,"SMPLGUIS",DATA_ADD,ADD_CONTINUE)
【リスト選択(SYOHINLIST_LB)】
        選択メンバ取り込み(SYOHINLIST_LB)
        オブジェクト復帰(SEL_END)
【ボタン押下(SYOHINLIST_BTEND)】
        オブジェクト復帰(PROGRAM_END)
【ボタン押下(SYOHINLIST_BTOK)】
        オブジェクト復帰(INPUT_OK)
【ボタン押下(SYOHINLIST_BTAGAIN)】
        オブジェクト復帰(LIST_SELECT)
【終了】
        外部モジュール呼び出し("SMPLGUIS",PROGRAM_END)
```

エンドユーザがリストボックス内の商品を選択すると、「リスト選択」イベントが発生します。「リスト選択」イベントの手続き記述にある「選択メンバ取り込み」手続きにより、選択したデータ内容がインタフェース領域に設定されます。

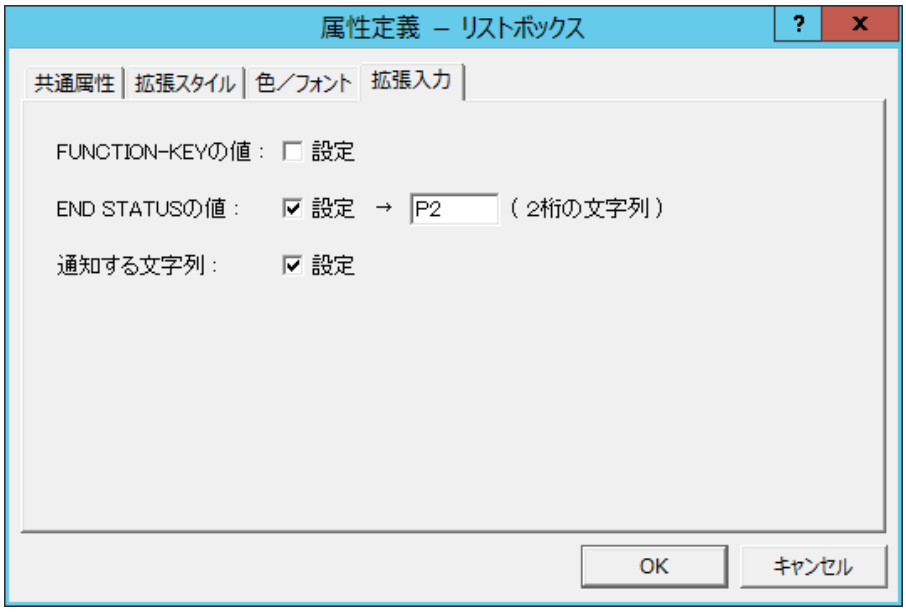
「連続メンバ追加」手続きについては「2.5 リストボックス操作時の COBOL サブプログラム呼び出し」を参照してください。

### 3.2. 画面節移行機能2

一部の GUI コントロールのデータは、COBOL の ACCEPT 文で取得することができます。ここでは、この機能の使用例を、前述のプログラムをもとに紹介します。

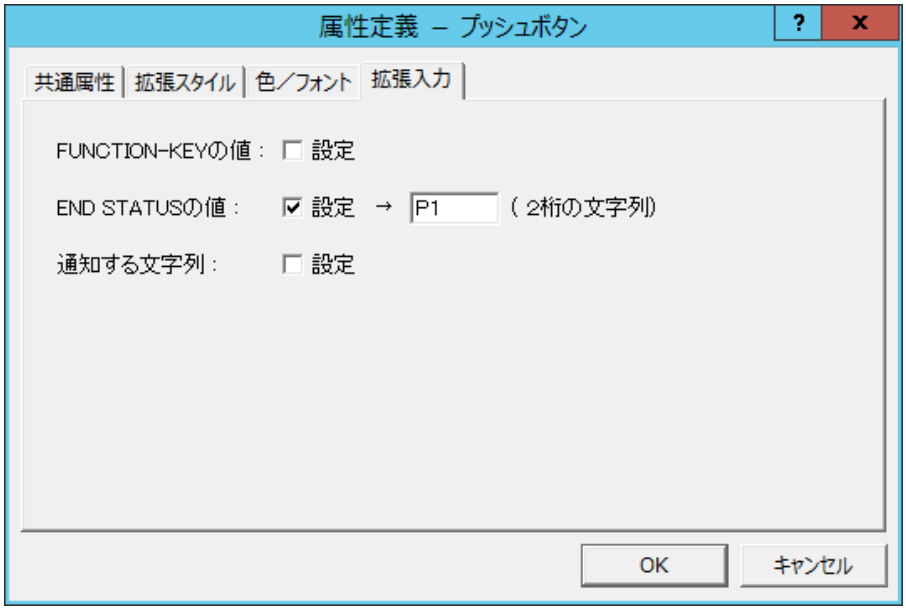
単純移行したGUI画面からの変更は以下の手順で行います。

1. COBOL プログラムの SCREEN SECTION にリストボックスおよびプッシュボタン入力用の入力データ項目を記述します。COBOL 開発環境を使用して画面節移行ファイルを作成し直し、View Generator で再度読み込みます。
2. 画面エディタを起動し、リストボックスの属性定義ダイアログボックスを開きます。拡張入力タブをクリックし「END STATUS の値」設定チェックボックスのチェック、END STATUS の値の設定、「通知する文字列」設定チェックボックスのチェックを行います。

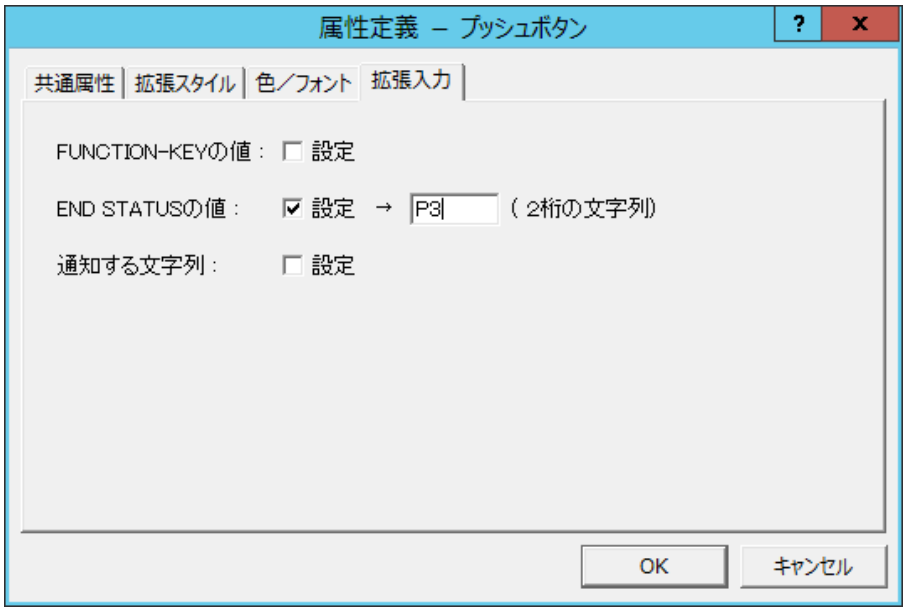



3. 1 と同じようにプッシュボタンの属性定義ダイアログボックスを開き、拡張入力タブをクリックし「END STATUS の値」設定チェックボックスのチェック、END STATUS の値の設定を行います。

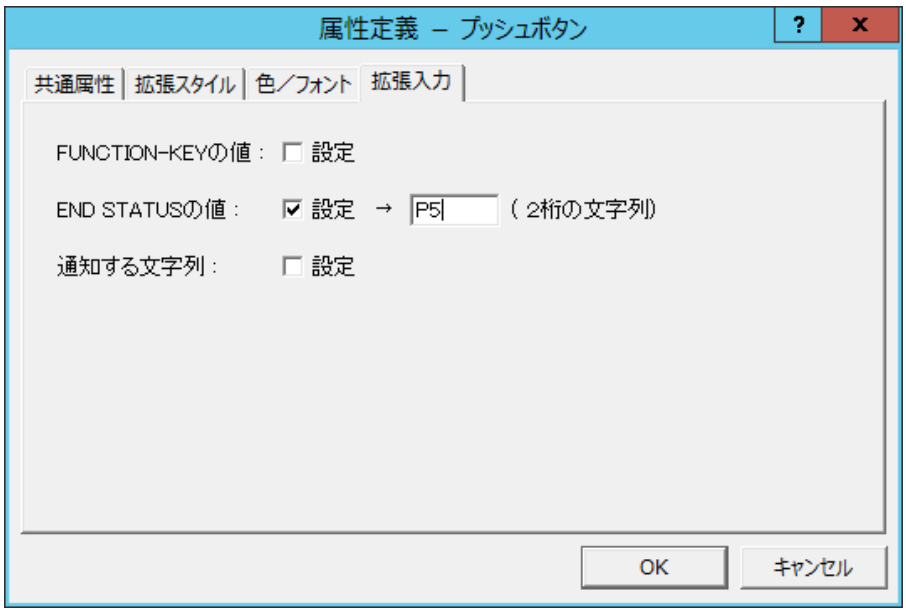
a. 「実行」プッシュボタンの属性定義



b.「商品再選択」プッシュボタンの属性定義




c.「終了」プッシュボタンの属性定義




4. COBOL 開発環境を使用して画面節移行ファイルを作成し直し、View Generator で読み込んで手続きを修正し、生成作業を行います。

5. COBOL プログラムを修正します。

```
001650     MOVE      LIST_ADD     TO  USEREVENT OF SYOHINLIST_EVENT.
001660     CALL      "CBL_WINDOW_EVENT" USING  SYOHINLIST_EVENT.
```

以前のプログラムと同じく、リストボックスに商品の一覧を追加するために、ユーザイベントの発行を使用しています。ただし、手続き命令でリストボックスの選択イベントは処理せず、リストボックスに商品一覧を追加し、すぐに COBOL メインプログラムに制御を移します。

```
001670     ACCEPT    GUI-DATA
```

リストボックスから商品を選択すると、GUI-DATA に内容が設定されます。
GUI-DATA は SCREEN SECTION で定義しています。
このときキーボードからデータを直接入力することも可能です。

```
001680     IF   ESTS = KEY-F2
001690          MOVE      W-GUIDATA   TO     W-LISTDATA
001700          GO        TO       INPUT-URI-KOSU.
001710     IF   ESTS = KEY-F5
001720          GO        TO       PROG-END.
```

リストボックスでの商品選択により、END STATUS 値が設定されます。
なお、ACCEPT の対象となるコントロールは、画面エディタで拡張入力を設定したコントロールすべてです。
入力対象のコントロールを制限するためには、「コントロールの有効化／無効化」システムサブルーチンを使用してください。
ACCEPT 文を実行する段階では「終了」プッシュボタンも有効になっており、リストボックスから商品選択が行われず、「終了」プッシュボタンが押された場合には「終了」プッシュボタンの属性定義で設定した END STATUS 値が返却されます。

```
001730 INPUT-URI-KOSU.
001740     MOVE   ZERO   TO   W-URIKOSU.
001750                 ACCEPT URI-KOSU .
001760     IF ESTS = KEY-F5
001770          GO       TO   PROG-END.
001780     MOVE      1     TO     STATE OF SYOHINLIST_LB.
001790     MOVE      2     TO     STATE OF SYOHINLIST_BTOK.
001800     MOVE      2     TO     STATE OF SYOHINLIST_BTAGAIN.
001810     CALL      "CBL_CONTROL_ENABLE" USING SYOHINLIST_ENABLE.
```

リストボックスの無効化、「実行」プッシュボタン、「商品再選択」プッシュボタンを有効化します。

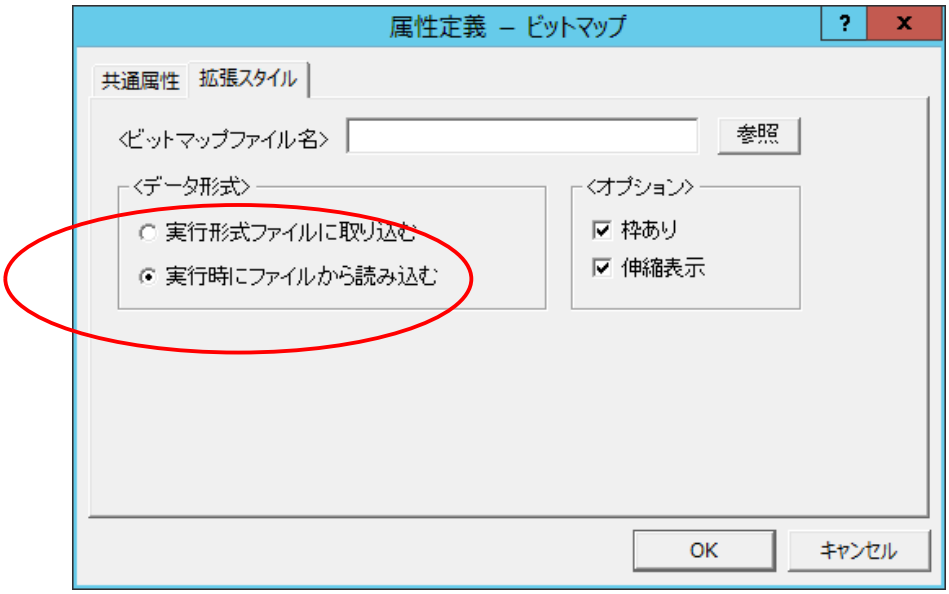
```
001820     ACCEPT   GUI-BUTTON.
001830     IF ESTS = KEY-F1
001840          GO       TO   DISP-GOUKEI.
001850     IF ESTS = KEY-F3
001860          GO       TO   LIST-SEL .
001870     IF ESTS = KEY-F5
001880          GO       TO   PROG-END.
```

いずれかのプッシュボタンが押されると、そのプッシュボタンに設定された END STATUS 値が返却されます。

# 第4章 よくある質問

この章では View Generator でのアプリケーション開発時によくある質問とその回答を示します。

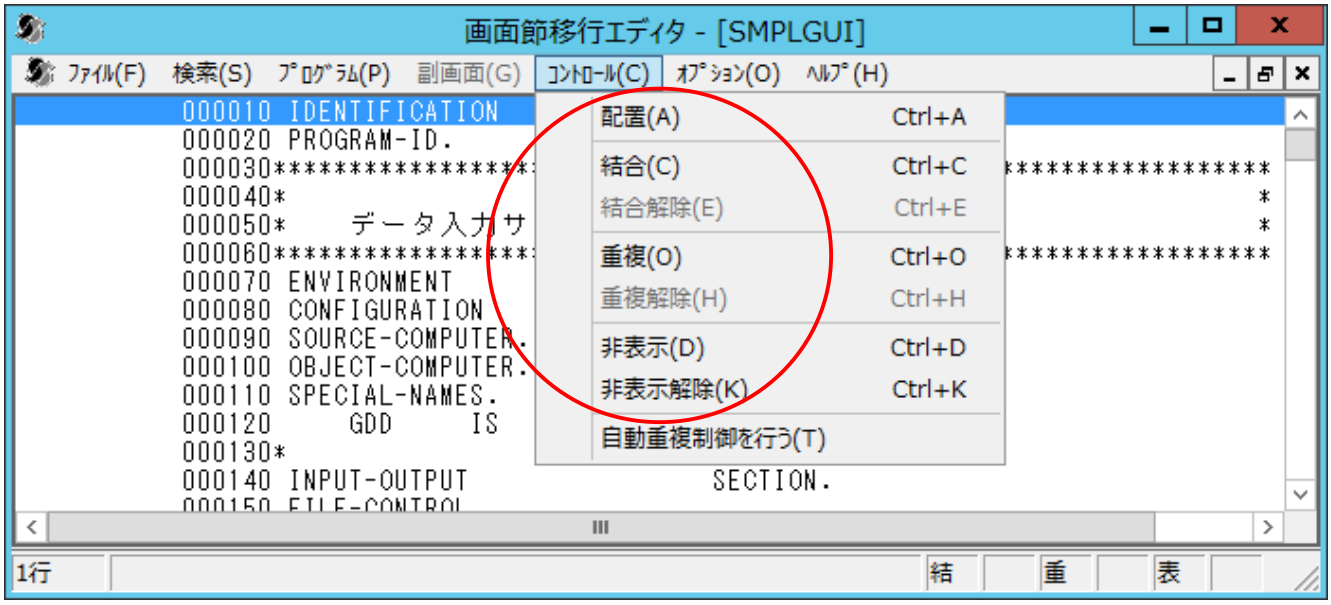
・ 画像データを貼り付け、実行時に動的に変更したい。
画像データを貼り付けるにはビットマップコントロールを使用します。
設計時:ビットマップコントロール属性定義ダイアログの拡張スタイルタブで、データ形式を「実行時にファイルから読み込む」にチェックします。




実行時:COBOL インタフェース領域に表示したいビットマップファイル名を指定し、「表示更新」手続きを行うか、あるいはシステムサブルーチン使用による変更も可能です。

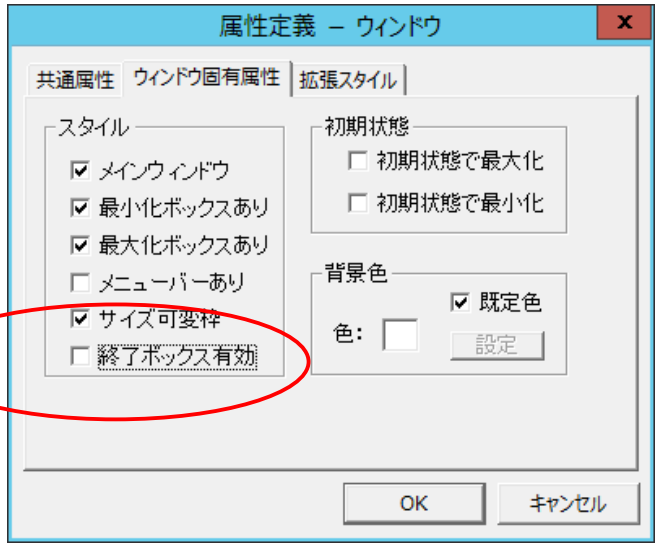
・ ボタンに表示している文字列を実行時に動的に変更したい。
ボタン(プッシュボタン、ラジオボタン、チェックボックス、グループボックス)に表示している文字列を変更するには「ラベル変更」手続きを行うか、あるいはシステムサブルーチン使用による変更も可能です。

・ 画面節移行機能を使用したプログラムを実行すると、表示が不正になったり、入力が正しくできない。
現象が発生している項目の領域に、複数の項目(画面節コントロール)が重なっている可能性があります。画面節移行したアプリケーションの入出力項目は、画面節のデータ項目単位にそれぞれ独立した画面節コントロールとして実装されます。複数の画面節コントロールがウィンドウ上に表示された場合、画面節コントロールが描画される順は OS に依存するため、プログラムで記載した順に表示されるとは限らず、期待通りに表示されないことがあります。
これを回避するには、COBOL プログラムを修正するか、または View Generator の画面節移行エディタを使用して、重なっている入出力項目に対し「重複」指定を行います。「自動重複制御を行う」メニューを選ぶと重なっている入出力項目すべてが自動重複の対象になります。




・　表を作成したい。
リストボックスを使用します。
リストボックスの１レコード内を空白で区切って使用します。具体的にはCOBOLインタフェース領域にデータを設定する際、各フィールド間に空白を挿入します。このとき、表示フォントにはMS明朝、MSゴシックなどの非プロポーショナルなフォントを使用してください。フォントはリストボックスの属性定義ダイアログの「色／フォント」タブで設定します。

・　ウィンドウのタイトルバーにある終了ボタン（右上隅の「×」ボタン）を使えないようにしたい。
終了ボタンの無効化には、設計時にウィンドウの属性定義ダイアログの「ウィンドウ固有属性」タグにある「終了ボックス有効」チェックボックスのチェックをはずしてください。
また、タイトルバー無しのウィンドウも作成可能です。ウィンドウの属性定義ダイアログの「共通属性」タグにある「タイトルバー有り」チェックボックスのチェックをはずしてください。



COBOL GUI
画面構築ガイド

2015 年 4 月　初 版
2023 年 4 月 第4版
日本電気株式会社
東京都港区芝五丁目 7 番 1 号
TEL (03) 3454-1111 (大代表)