ｏ’ｔａｓｕｋｅスタッフ

Ver 2.0

デジラボ高知
DIGITAL CONTENTS LABO KOCHI

おたすけスタッフを作ろうとしたきかっけは

２０１４年にソフトウェア会社を辞め、自分に何が出来るかを考えＷＥＢの勉強そしておたすけの前身であるＶｅｒ１．０を開発しました。

このＶｅｒ１．０では開発のし易さなどはまったく考慮せず、機能的に新しさを求めたコントロール群を作成したものでした。

その時の開発は生産性はあがらずコーディング量はただ多くうんざりするほどの力仕事となりました。

そこで今回、今現在自分に出来るできる技術の集大成をという思いもあり
幅広く誰にも使ってもらえるように、Ｃ＃．Ｎｅｔにて一から作り直してきたのが
このＶｅｒ２．０になります。

このフレームワークをたくさんの人に使ってもらい機能の拡充とより良い改善が進むことを願っています。

2018年10月
溝渕　豊

その主な特徴は．．．

Winndows標準コントロールを継承し作成された多数のコントロールにあります。

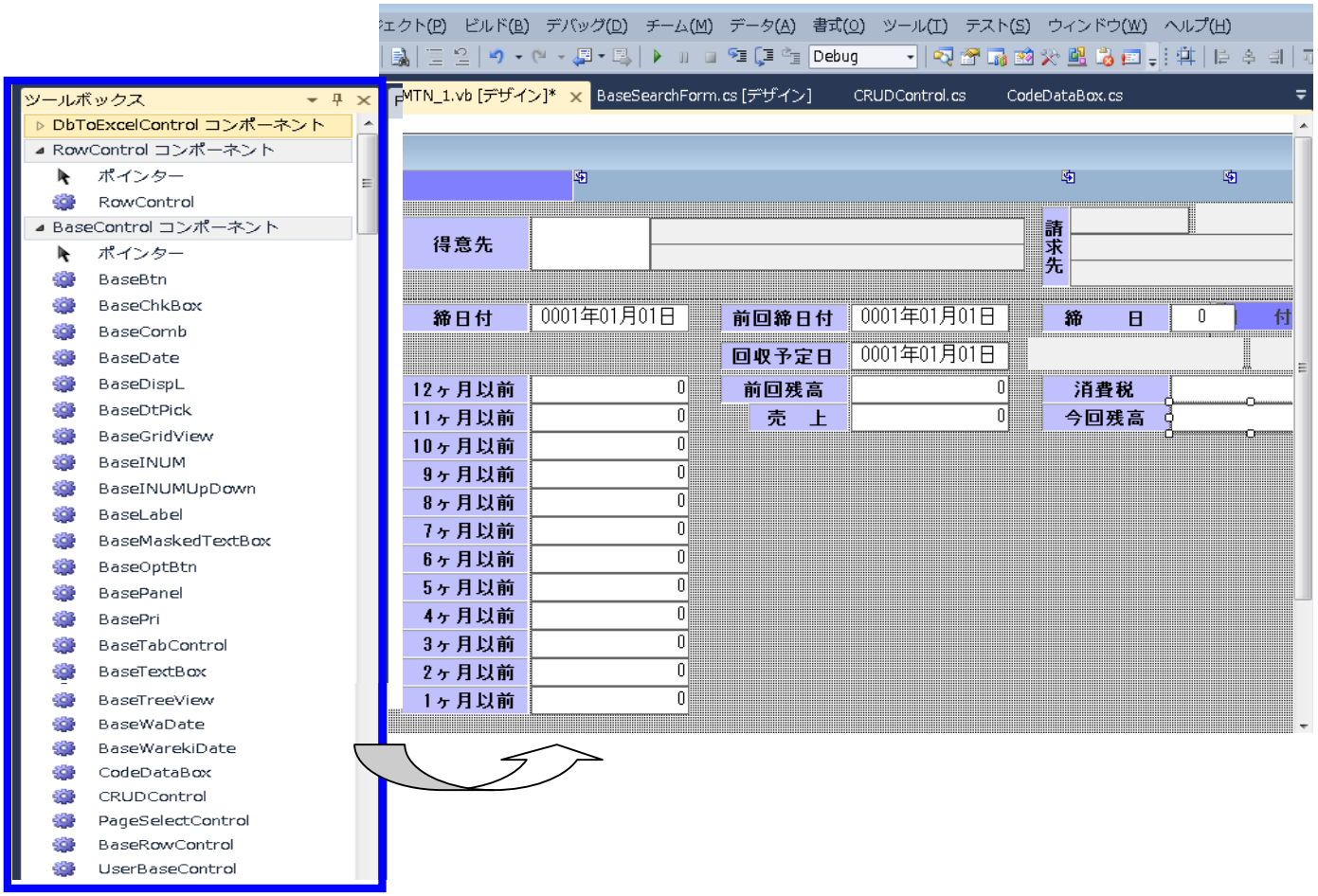


図1 Visual Studio IDE にてBaseFormにコントロールを張付けた模様

極力コーディング量を減らせれるようなそんなツールができればとの思いで考えられたコントロール群です。

その主な特徴は．．．

まず画面上の動きですが

ひとつひとつのコントールに対してコーディングが必要なのは無駄ですから、移動の場合は”エンターキー”と”下向き矢印”で次の項目へ進みます。

上向き矢印では一つ前の項目へ戻ります。”タブキー”と”ｓｈｉｆｔ”＋”タブキー”と同じ役割を持っています。

これはテキストボックスをはじめ全ての入出力可能なコントロールについてインタフェースにより実装するようになっています。

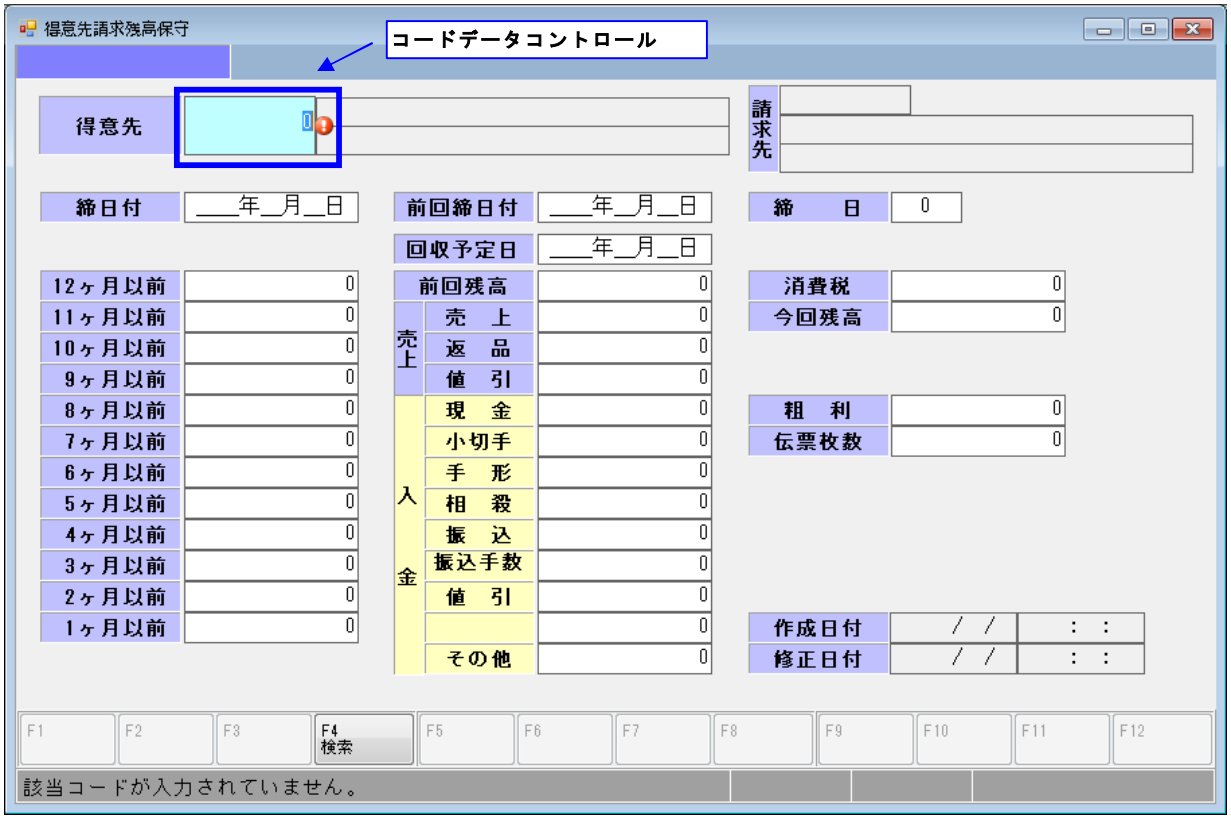


図2 開発画面（ｺﾝﾄﾛｰﾙを張付けた状態）

画面へ配置するだけで前後のコントロールへ移動
チェック条件の記述はチェックイベントにのみ記述すればよい

そして図2にもあるコードデータコントロールですが、マスタなどのテーブルアクセス機能を持たしました。

プロパティにもつテーブル名により該当テーブルのテーブル・キー情報やフィールド情報などを保持するようなっています。

図2のコードデータコントロールを配置した状態のみ(コーディングせずに)で「検索」ＰＦキーの表示がされています。
エンターで”該当コードが入力されていません”のエラーメッセージが表示されています。

検索機能（検索画面を表示し検索済みコードを取得する）
取得した名称などを指定コントロールへ表示する

その主な特徴は．．．

　マスタ保守や入力処理では上記コードデータコントロールに加えＣＲＵＤコントロールが有効です。
コードの追加・修正・削除ロジックや画面制御を行います。

下図はCRUDコントロールをツールボックスから画面へ張付けた状態です。

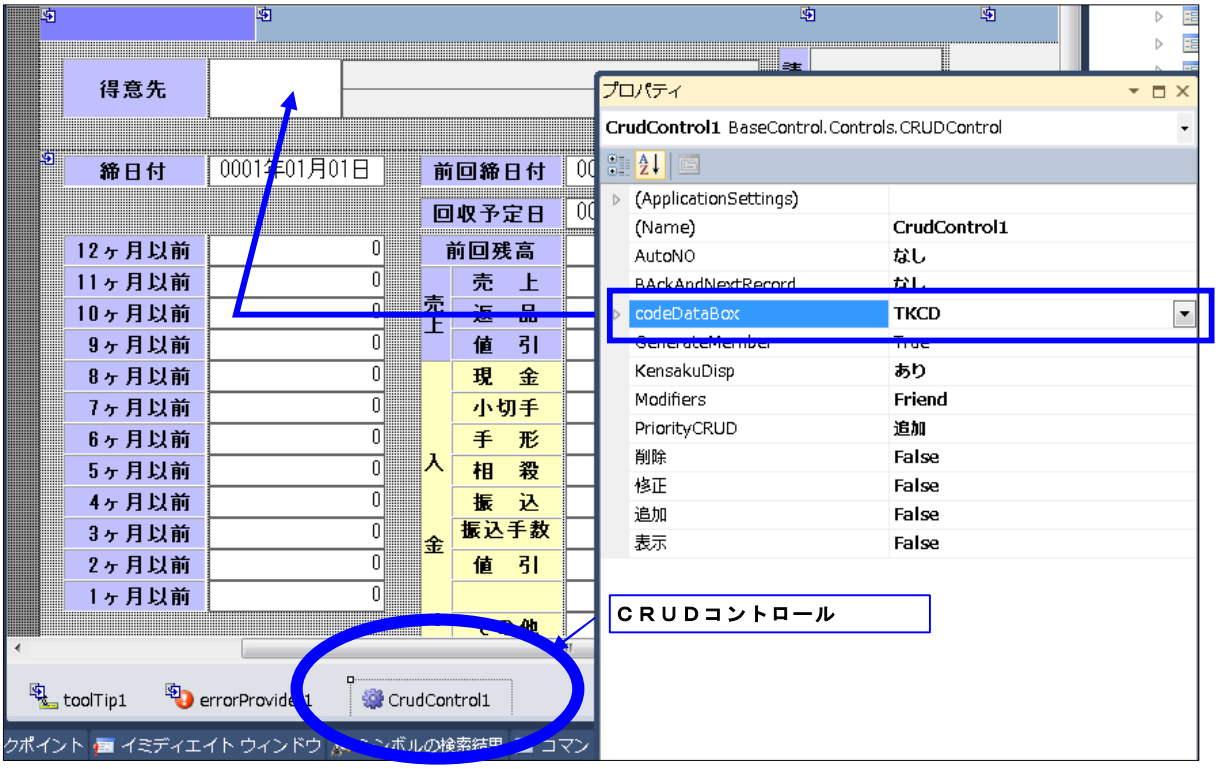


図3 開発画面（CRUDｺﾝﾄﾛｰﾙを張付けた状態）

プロパティのCodeDataBoxにて画面上のｺﾝﾄﾛｰﾙ”TKCD”を選択します。
この画面で選択可能なモード｛追加、修正、削除｝等をそれぞれ選択します。

> 追加・修正・削除モードそれぞれの更新制御を行う
> 画面の入力制御（保護・非保護）なども行う

その主な特徴は．．．

開発システムではそのシステム独自に設定したい項目、でもそれだけにプログラムを作るほどでもないような、そんな項目は昔でいうところのｉｎｉファイルに設けておけばなんて考えもありますが開発で増えていく項目数も気がつけば大きくなっていて管理も大変だったってことは無いでしょうか？

そういった場合に便利なのが項目マスタです。この項目マスタには各種いろいろなデータを登録することができます。

例えば名称マスタ(いろいろなコードと名称の組み合わせ、地区、分類、担当,…etc)など
名称マスタのコードが数値か文字か、名称の桁数などを項目マスタに設定しておくことで名称マスタ保守でメンテナンスできるような仕組みになっています。他にシステム固有区分なども登録しておくと便利です。

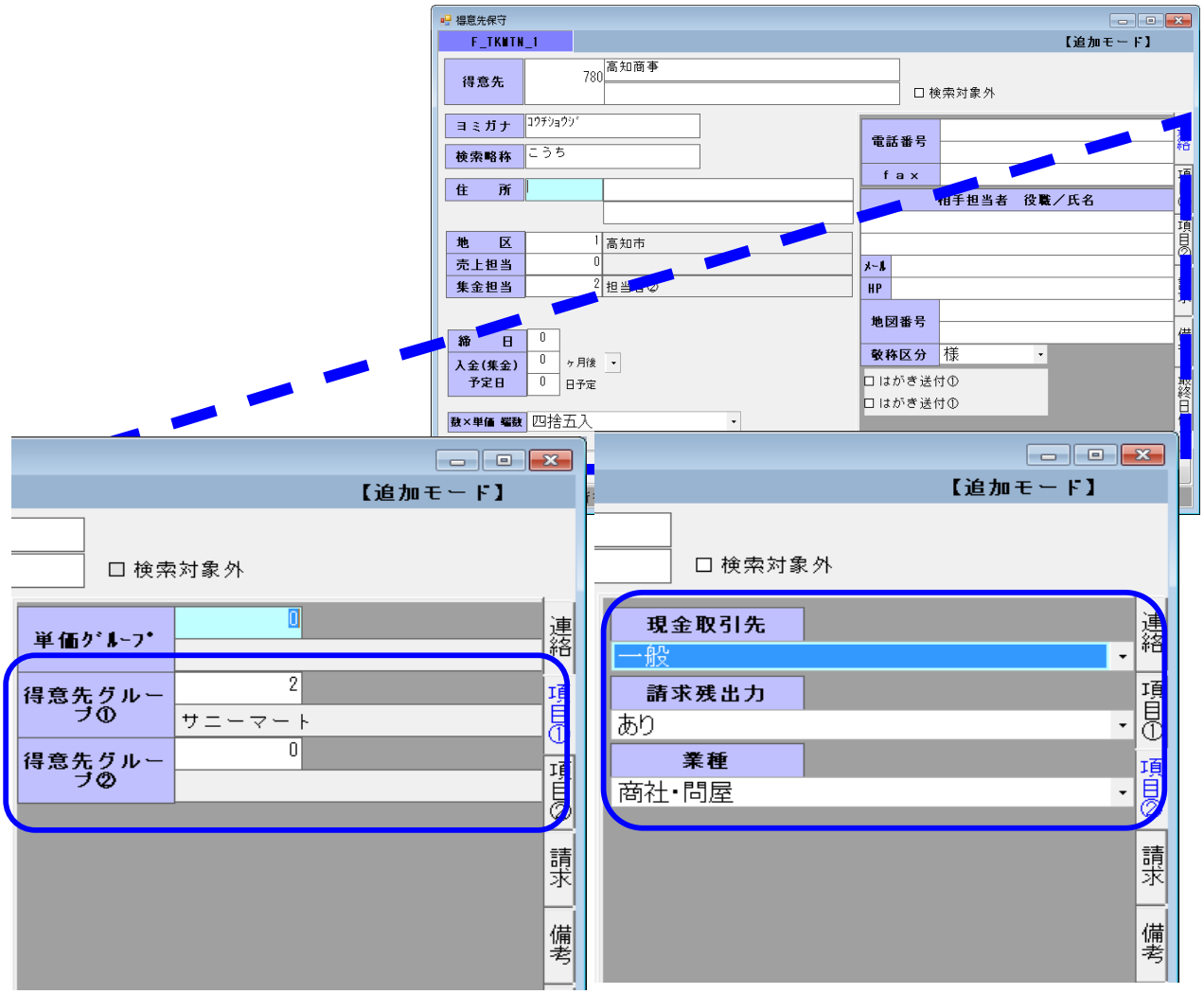


図4 得意先保守画面(例)

これらはプログラム固定で表示されているものではありません。項目マスタにある得意先マスタ用に設けた５つのコード入力と５つのドロップダウン項目の設定に名称マスタに登録済みそれぞれの候補により構成されています。

システム固有で必要となる項目もこうやって任意に設定すればそれだけ手間も省かれることになるかと思います。

汎用的に使える項目マスタを利用する

その主な特徴は．．．

しかし全てを項目マスタに集中して持たしてしまうと、整理・管理ができているとはいえず、一般の担当者が設定を訳もわからず修正してしまう恐れもあります。

本フレムーワークではテスト時も含めログインするユーザーには３種類の権限からいずれか一つの権限を持つようになっています。

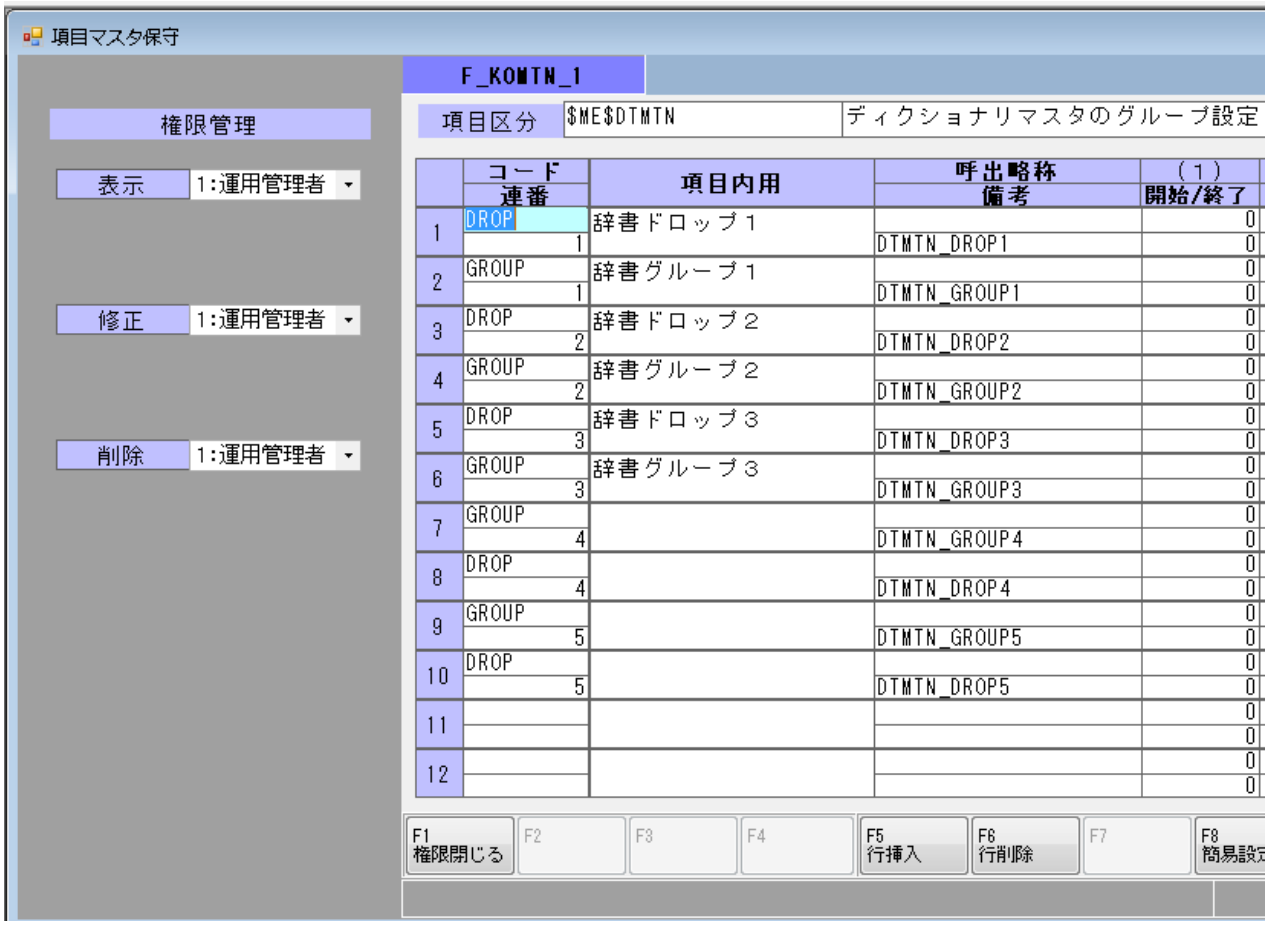


図5 項目マスタ画面

一般担当者は導入後業務を操作する担当者を、また運用担当者は業務全般に対して権限を持つエンドユーザーをそしてシステム担当者として開発し収める側のシステムエンジニアを想定し、それぞれの区分に応じて検索・表示から修正・削除に関して扱えるデータを分けています。

ログイン担当者の権限により項目マスタの表示、修正、削除ができるか制御する

その主な特徴は．．．

エンドユーザーの要望で担当者毎にメニューを分けたいなど個別に設定するのは面倒です。

さきほどのログインユーザーにはレベル項目を設け″０″から″９″までのレベルを設定できるようにしています。

メニューの個々のボタンにも扱い可能なレベルが設定できますからそれ未満のレベルのログインユーザーの場合にはボタンの表示すらされない仕組みになっています。

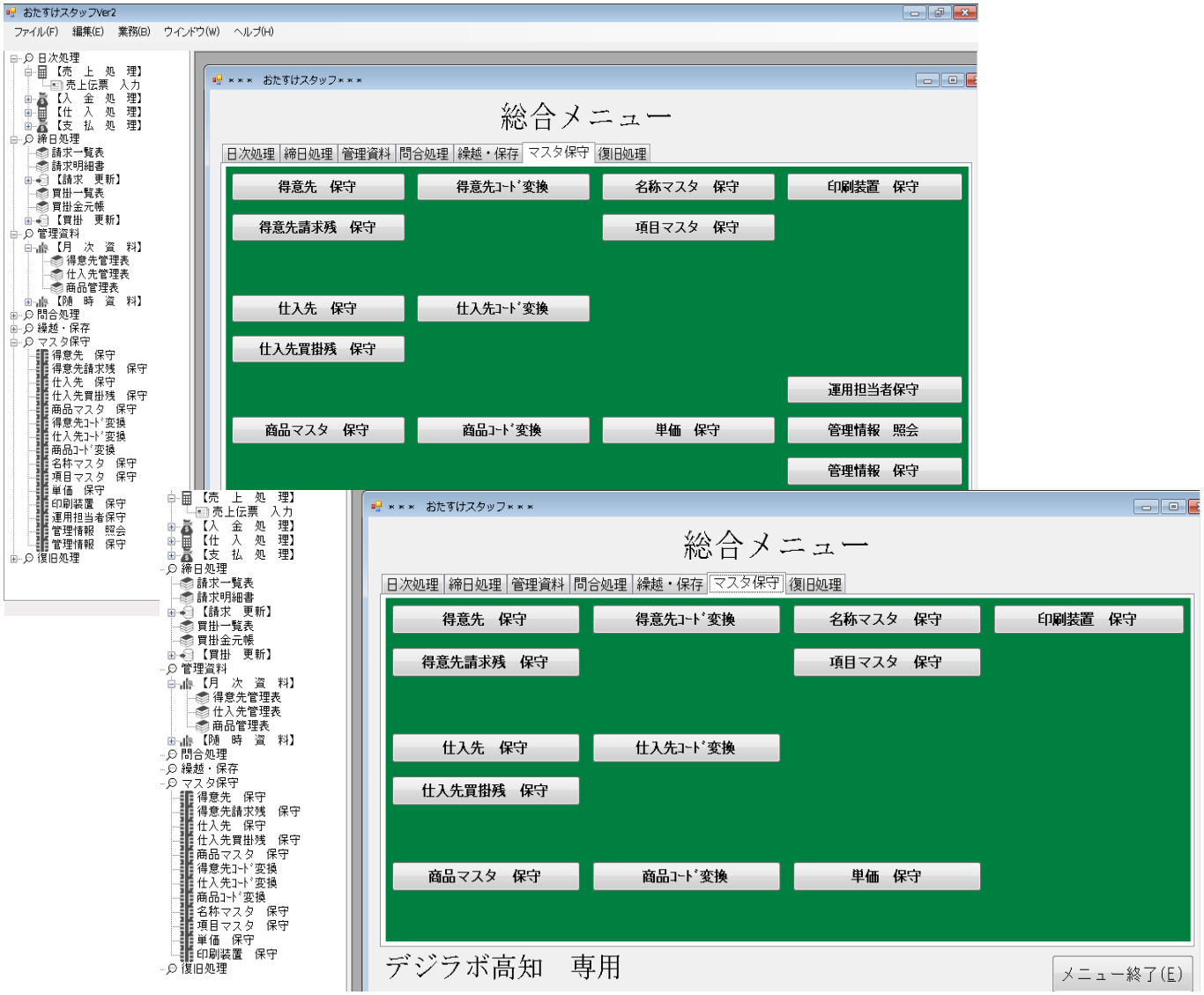


図6 メニュー画面

ログイン担当者のレベルによりメニューボタンの表示、非表示かを制御する

●コードデータコントロールとは

　コードデータコントロールではプロパティでテーブル名をセットするだけで、テーブル構造を解析し内部でテーブルのアイテム名や属性を保持します。

※自動解析される為、テーブル毎にレイアウトソースを作る作業は不要です。

　またＰＦキーに”検索”表示を行い、検索キーが押されると内部の検索アクションメソッドを実行し検索画面を表示します。

検索された結果、コードが選択された場合はテーブルのアイテム名に一致する各フィールドにその内容を表示する役目を担っています。（プロパティの設定のみで表示されます）

（画面に配置されたコントロールにはDatafieldプロパティを持っていてセットされたアイテム名が一致すればその内容がセットされます。）

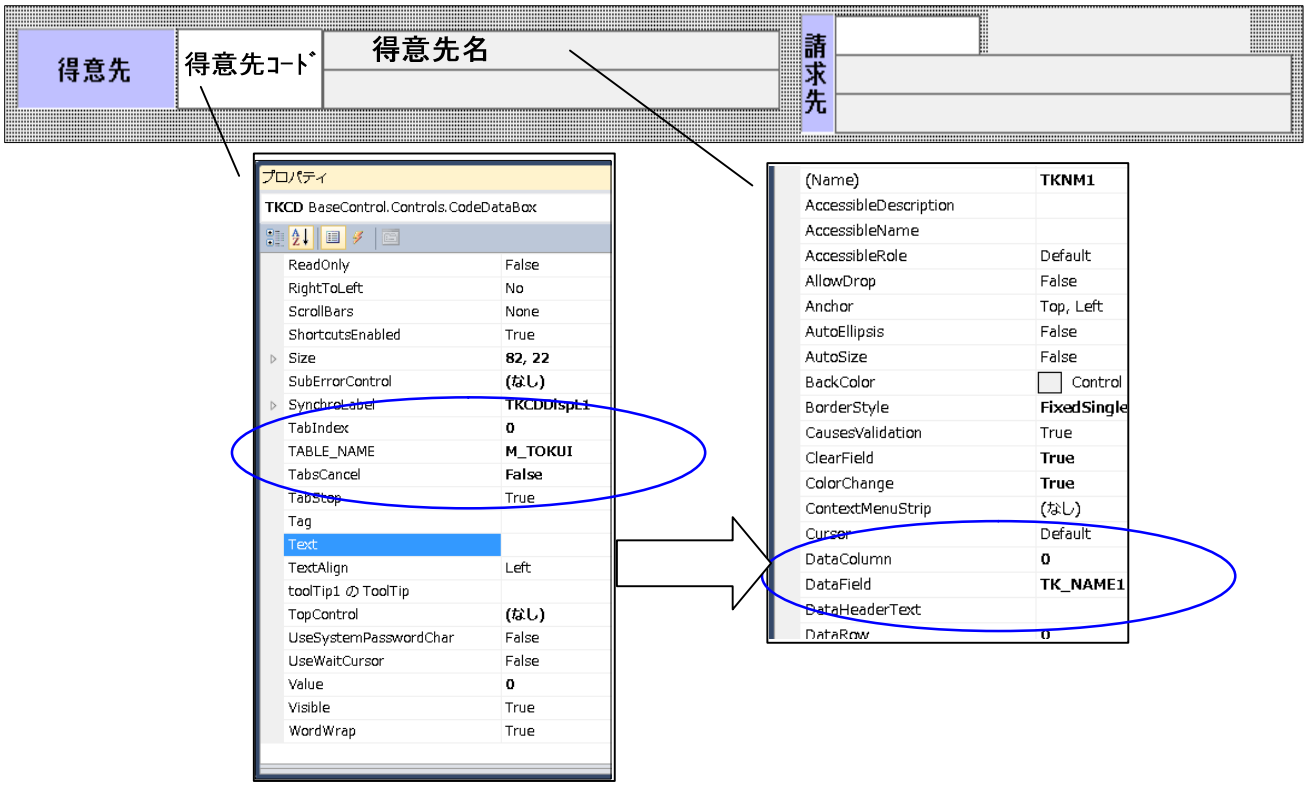


図1 コード部と名称部の各プロパティ

検索画面を共有し各テーブルでは表示する項目などセレクトする記述をコーディングしておきます。

検索画面では以下のように”あ”,”か”,”さ”．．．と「かな別」に表示されます。

図2 検索画面外観

●コードデータコントロールとは

　コードデータコントロールは全て自動にＳＱＬを組み立て．．．といっても全てが完全に思惑どうりになるものではありません。

以下、画面例をご欄ください。

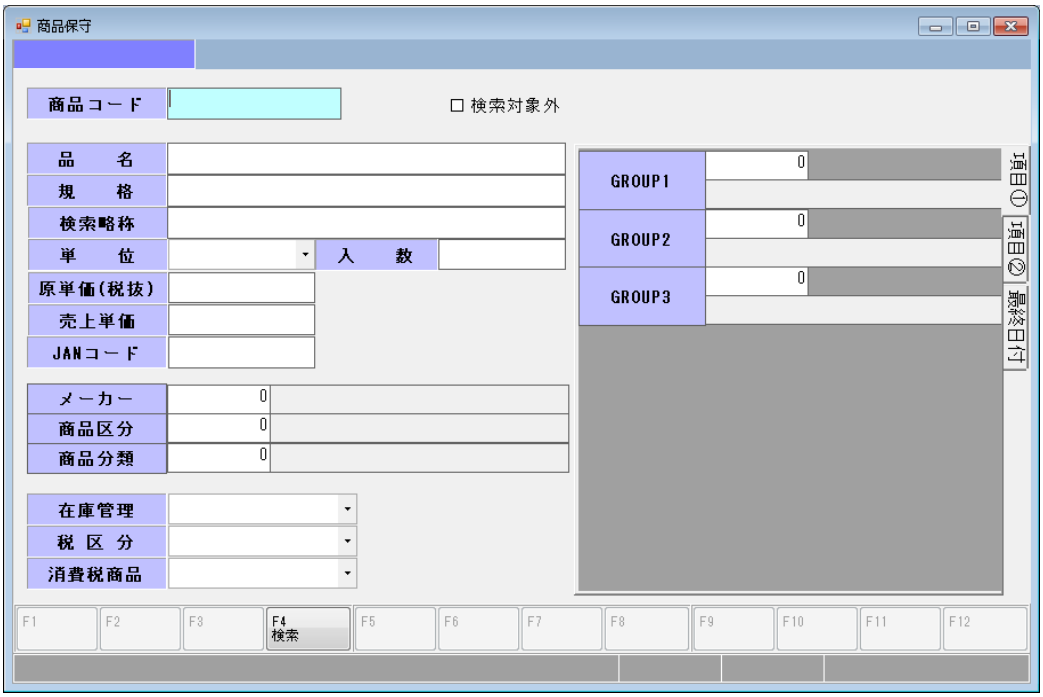


図3 商品マスタ保守画面

この場合、最上部の商品コードがコードデータコントロールになります。
その他にもメーカー、商品区分、商品分類やGROUP1～GROUP3などもデータコントロールです。

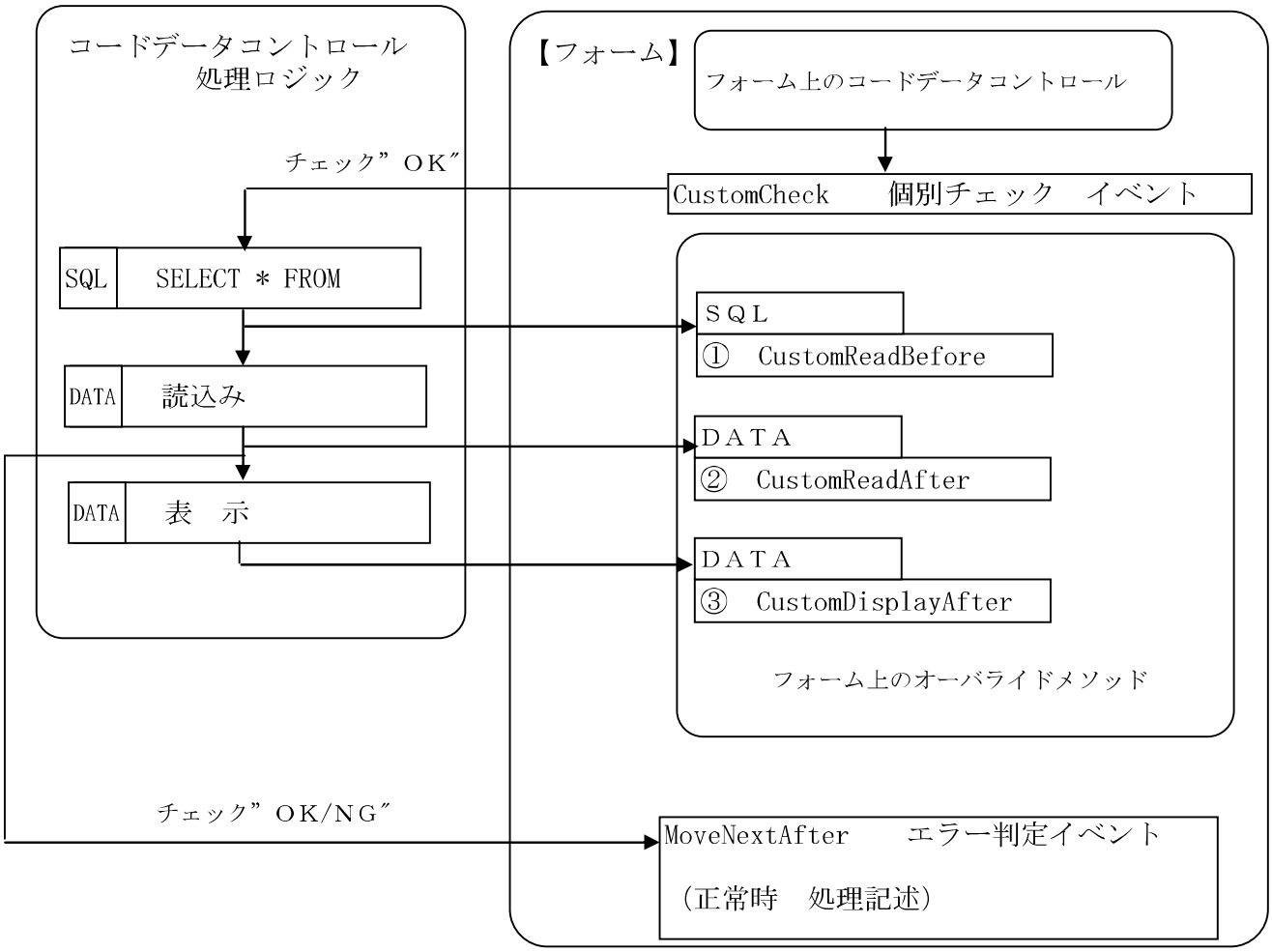
　品名から下の各項目のプロパティには"DataField"プロパティをそれぞれ持っておりこの場合は商品マスタのテーブルフィールドＩＤをセットしていきます。

ただ、メーカーや商品区分などの"DataField"プロパティには名称マスタの"ME_CD_NUM"のフィールド名が必要であり商品マスタのフィールド名を指定することができません。

こような場合や商品マスタ以外の項目をセレクトしたい場合もあります。
　そのような要求に対応する為、ＳＱＬを組み立て（①）テーブル読込み（②）画面へ表示後（③）というように、その各事象のイベントの前後に処理を記述することが可能となっています。

●コードデータコントロールとは

ちょっと解りにくいので図にしてみることにします。



・コードデータコントロール読み込み処理

① CustomReadBefore：CodeDataBoxにて読み込み前に、呼び出されるメッソド
・読込みSQLなどを変更できます。

② CustomReadAfter ：CodeDataBoxにて読み込み後(1)に、呼び出されるメッソド
・読込み後の内容について処理します。

③ CustomDisplayAfter：CodeDataBoxにて読み込み後(2)に、呼び出されるメッソド
・画面表示後の処理を行います。

●コードデータコントロールとは

コードデータコントロールをシステムに導入するうえでの制限事項

コードデータコントロールは複数キーが存在するテーブルは基本的には扱えません。
（本システムでは項目・名称マスタなどについて一部対応しています）

コードデータコントロールは文字かまたは数値を入力するコードのみ対応しています。（日付入力などを必要とするキー項目は扱えません。

以上の点をふまえてシステム設計していくことが必要です。

●CRUDコントロールとは

「ＣＲＵＤ」とはおなじみのフレームワークで特にＰＨＰなどでの「Ｃｒｅａｔｅ（新規作成）」、「Ｒｅａｄ（読み込み）」、「Ｕｐｄａｔｅ（更新）」、「Ｄｅｌｅｔｅ（削除）」のイニシャルが元となったもので有名ですが、それぞれのロジックを担うのが本コントロールの役目です。

ＣＲＵＤコントロールはプロパティで指定されたコードデータコントロールの各イベントを捕捉して動作するコントロールです。
各モードにより項目を入力できない状態とする(保護)やそれを解除するなどの制御を行います。

図1 IDEにてフォームに追加されたCRUDコントロール

本コントロールのプロパティにさきほどのコードデータコントロールをセットします。(図2)

※ 対象画面で追加、修正、削除（モード）の主となるメインのテーブルを指定するものです。

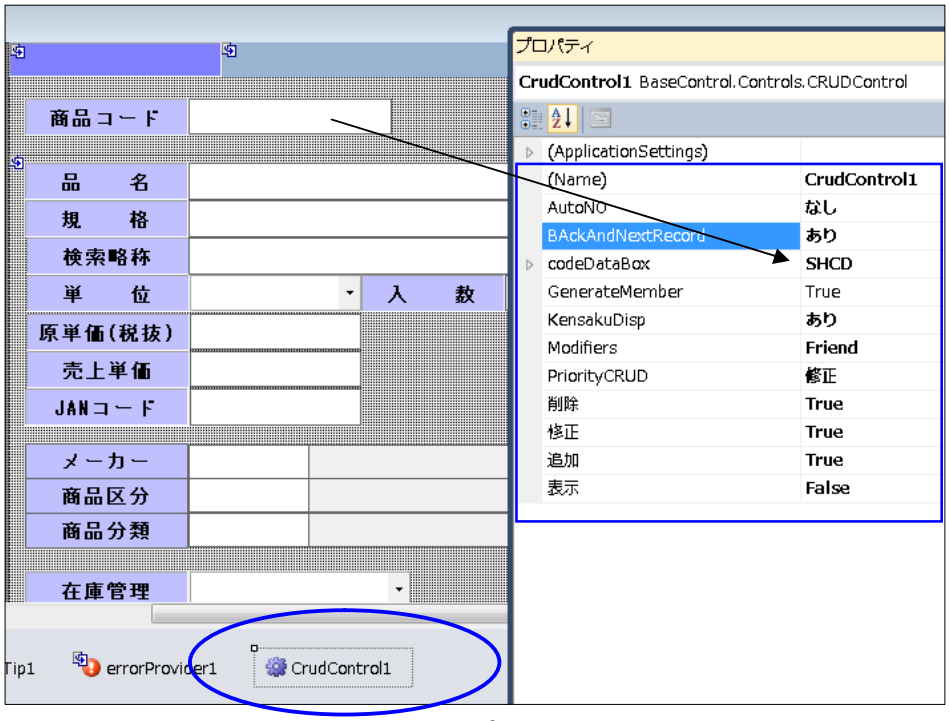


図2 CRUDプロパティ設定

各モードにて存在しなかった場合や既に存在する場合をコードデータコントロールと連携しエラー判定を行います。

エラー判定後、更新を行うか問い合せをし、”ＯＫ”の場合、対象のテーブルに対して各モード追加・修正・削除により更新処理を行います。

コードデータコントロールで読込みＳＱＬ文を修正する場合などがあると記述しましたが、書き込みなどの処理にもそのような仕組みが必要となります。

ＣＲＵＤコントロールにも書き込み前の独自チェック記述用メソッドや書き込み時に内容を上書き・追加するなどの処理を記述することが可能となっています。

●CRUDコントロールとは

やはり解りにくいので図にしてみることにします。

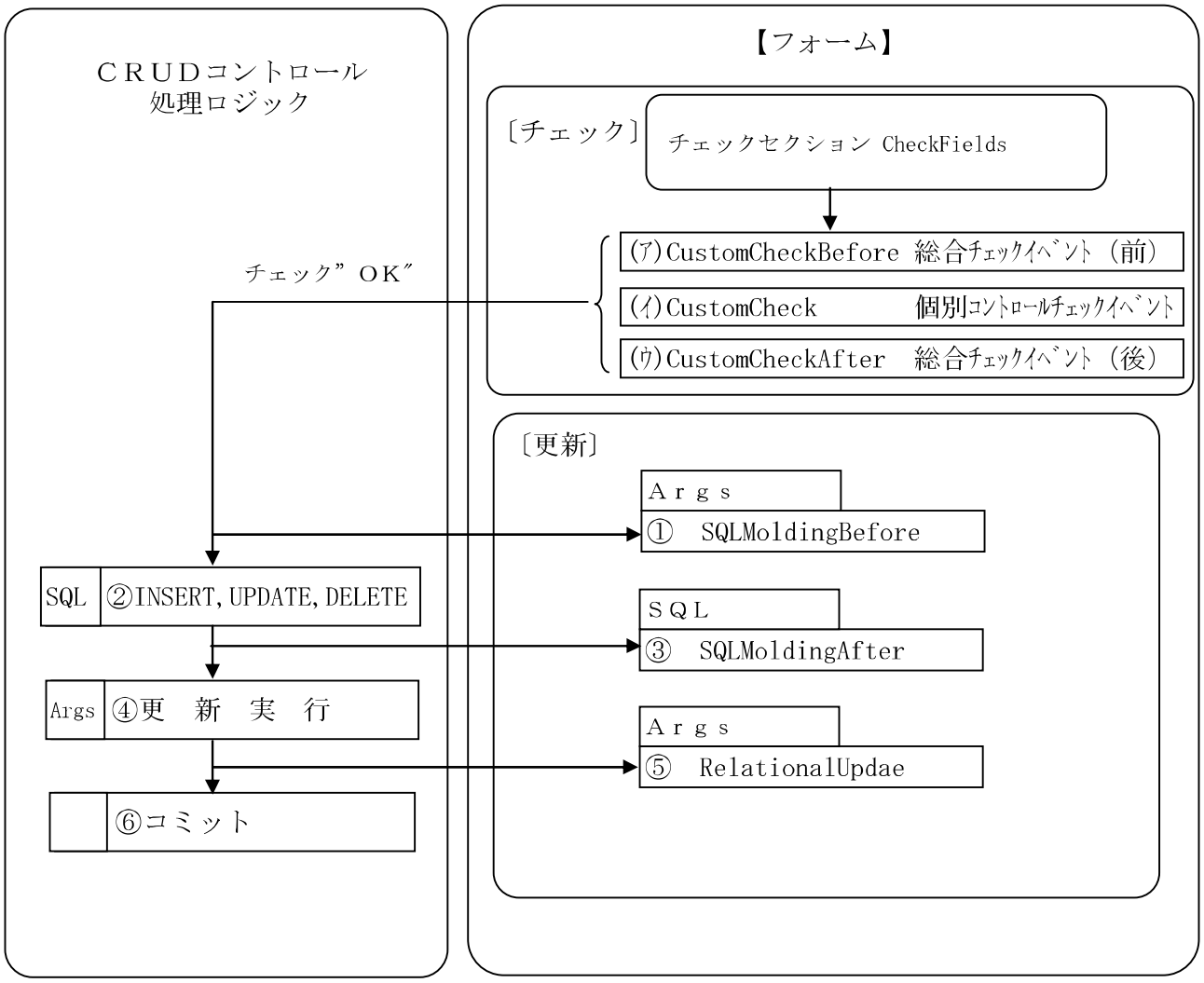


図3 CRUDフロー図

・チェック時

(ｱ)　CustomCheckBefore：ALLチェック前に、呼び出されるメッソド
(ｲ)　CustomCheck　　：個別チェック
(ｳ)　CustomCheckAfter　：ALLチェック後に、呼び出されるメッソド

・書き込み時

①　SQLMoldingBefore　：　ＳＱＬ成型前・編集処理① ＳＱＬ成型前・編集処理
②　　：　ＳＱＬ成型処理（CRUD側　処理）
③　SQLMoldingAfter　：　ＳＱＬ成型後（更新前）・編集処理
④　　：　更新(ＥＸＥＣ)処理（CRUD側　処理）
⑤　RelationalUpdae　：　コミット前処理（関連ファイル更新）
⑥　　：　コミット

●CRUDコントロールとは

以下はＣＲＵＤコントロールを配置し実行した様子です。

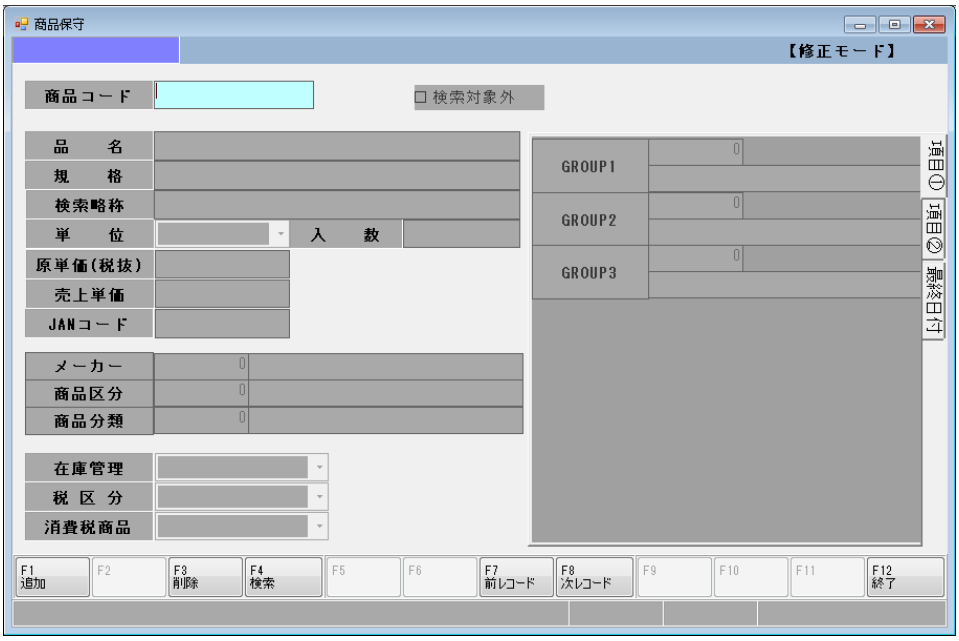


図4 CRUDコントロールを設定し実行

・コード部以外は入力できずモード選択や「PF12 終了」が表示されました。

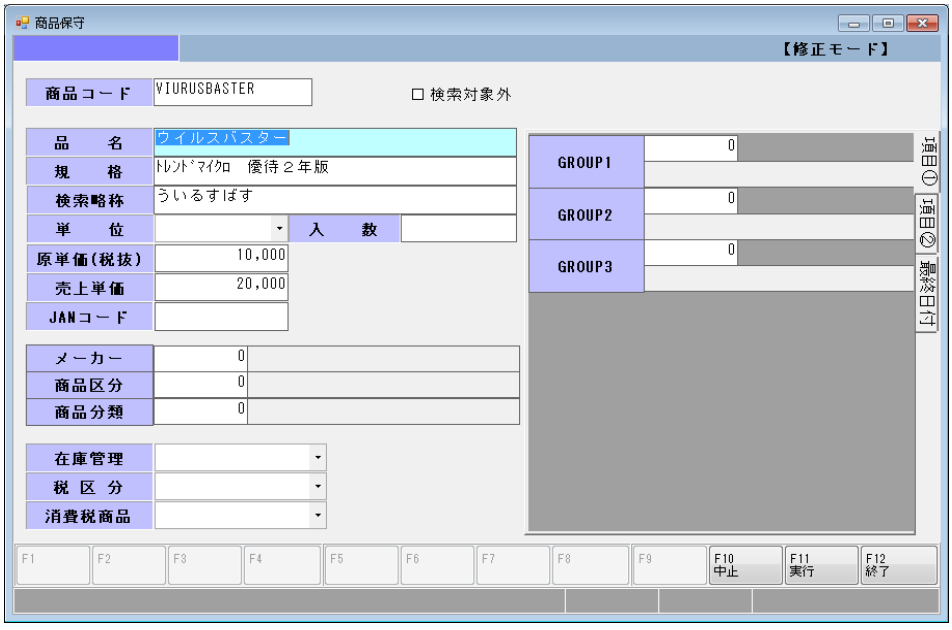


図5 検索選択後の画面

・ＰＦ１２を押して終了することができます。

この状態でソースは未だ１行も書かれていない状態です。さすがに「PF11 実行」はできませんが、「PF10 中止」「PF12 終了」は機能します。

●ROWコントロールとは

概要でも述べたように、主に多段の画面入力設計を実現すべく且つ、DataGridViewより入力ロジックに難なくコーディングできることを目標に開発したコントロールです。

実際の開発作業も簡単でユーザーコントロール上に配置済みのヘッダー、ボディ、フッターの３つのパネル内(図 1)にベースとなるコントロールを配置(図 2)して、１行の入力と外観が決定されます。
さらにツールボックスより画面に貼り付けたＲＯＷコントロールのユーザーベースコントロールプロパティにこのユーザーコントロールをセットするのみです。

実行時画面は(図 3) のように展開されます。

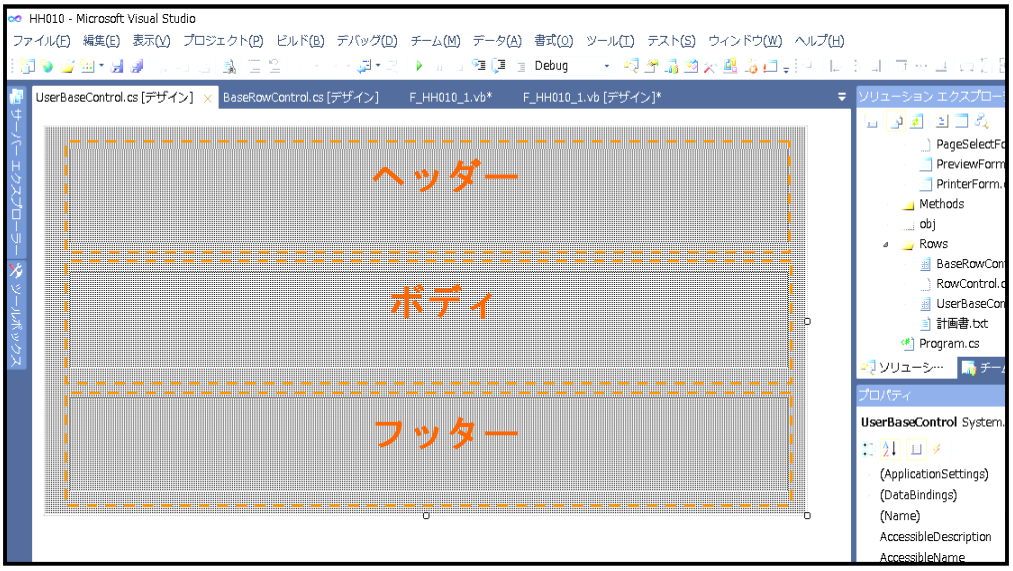


図1 ユーザーコントロール

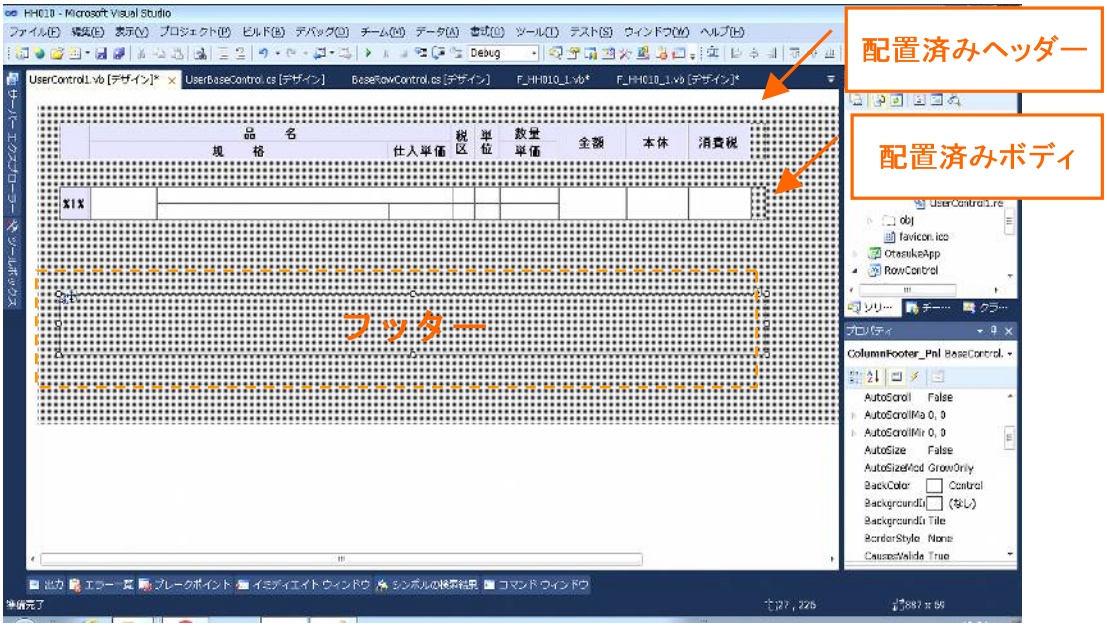


図2 配置済みユーザーコントロール

●ROWコントロールとは

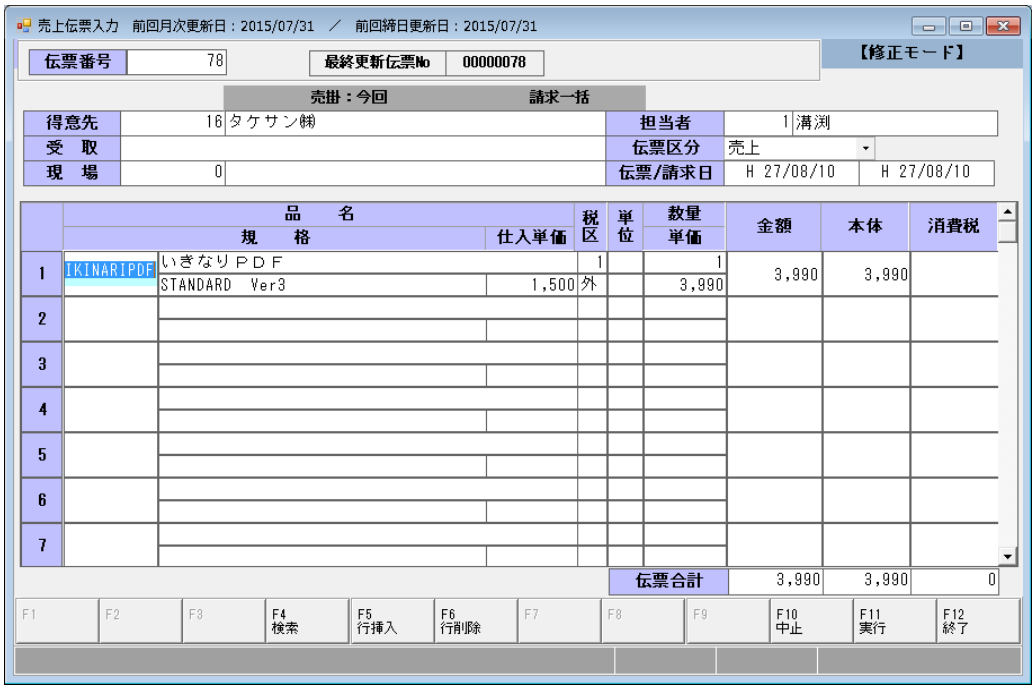


図3 実行時のROWコントロール

内部的には簡単に配置してそれだけで処理が完結するわけもなく、実際の開発においてはそれぞれの項目毎に移動時にチェックが必要であるとか実際の動作に求められる細かな要求は多いのも事実です。
各項目毎に捕らえられるイベントや値設定などのメソッドをもうけて実装できるように配慮されています。

ただＣＲＵＤコントロールは単一のテーブルしか処理の対象とはなっておらず、上記の場合など売上ヘッダーと売上明細テーブルに分かれている場合、売上テーブル部分の読込みや更新については別途コーディングする必要があります。

※またWindowsコントロールの配置上限がある為、上記の例では300行がmax値となります。行数の制限値はコントロールのプロパティLimitMaxMrowsを参照により求められます。

行数制限の無い場合などについてはRowコントロールは適切ではありません。
DataGridView などを使うようにしましょう。

保守入力処理

保守入力処理について
以下は得意先マスタ保守の画面です。

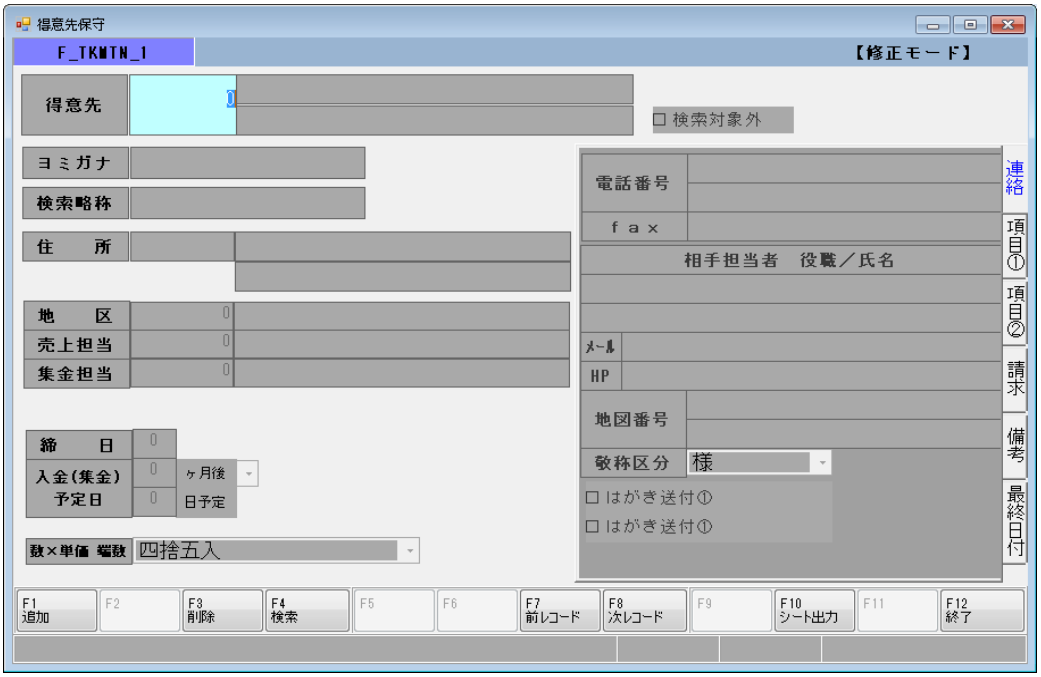


図1 得意先保守画面(1)

この画面でのコードデータコントロールは得意先コード、地区、売上担当、入金担当など他タブコンロールにもあります。
またＣＲＵＤコントロールにより右上に「修正モード」の表示がなされ得意先コード以外がグレーの保護状態になっているのもわかります。
追加モード「F1 追加」に切替えて入力してみます。

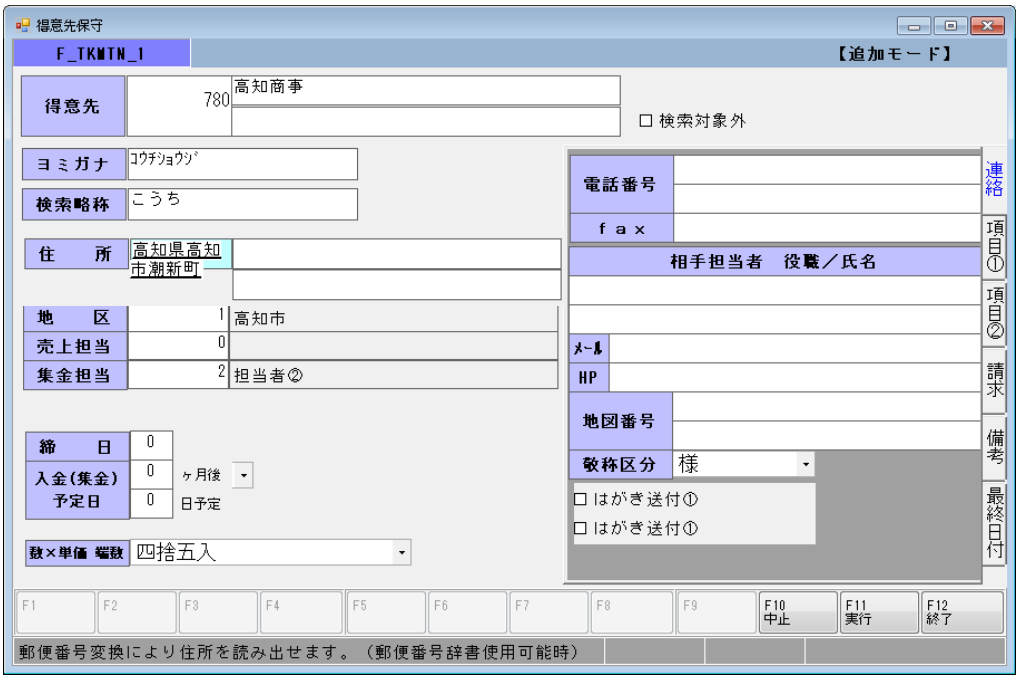


図2 得意先保守画面(2)

得意先名のﾌﾘｶﾞﾅ”ｺｳﾁｼｮｳｼﾞ”がヨミガナ変換機能により自動入力されましたが、住所の郵便番号項目に位置づけた時にガイダンスに「郵便番号変換により．．．」の表示がなされています。この項目でもヨミガナ変換機能（郵便番号辞書機能）を使っています。
そのまま７８０－８００と入力し変換すると、郵便番号の欄に漢字で”高知県高知市..”と表示されますが、そのままエンターキーを押すと住所１欄に表示されます。

各種印刷処理

印刷処理について
以下は請求書作成の画面です。

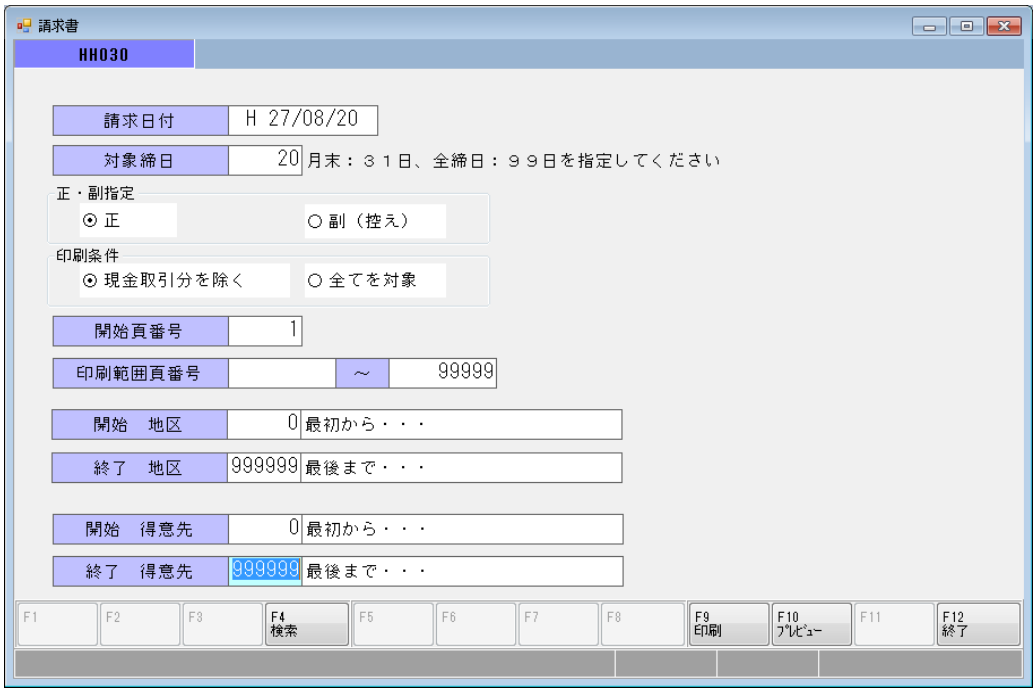


図1 請求書印刷指示画面

この画面でもコードデータコントロールを使っています。

地区と得意先の開始・終了については項目がアクティブになった場合、ＰＦキーへ”検索”を表示し検索が可能となります。

また値がMin値やMax値の場合に”最初から・・・”、”最後まで・・・”などのメッセージを出力することがプロパティの設定により可能となっています。

「F10 プレビュー」 を押してください。

図2 請求書プレビュー画面

各種印刷処理

定型化され各PFキーに処理が割当てられています。
「F4 印刷頁指定」を押してください。

図3 請求書プレビュー画面(2)

続いて「F1 印刷」を押します。

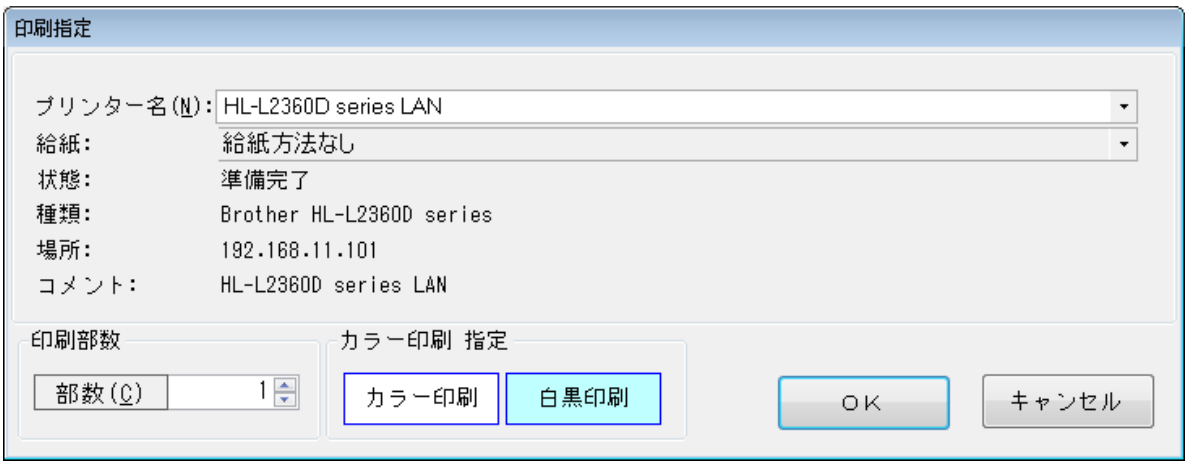


図4 印刷ダイアログ画面

「ＯＫ」印刷します。

プリンターについては端末・業務プログラム毎に記憶され次回印刷時にデフォルトプリンターとして表示されます。

かんたんメニュー作成機能

まずはメニューの主な機能をを見てください。
①起動時にログイン画面の表示がされます。

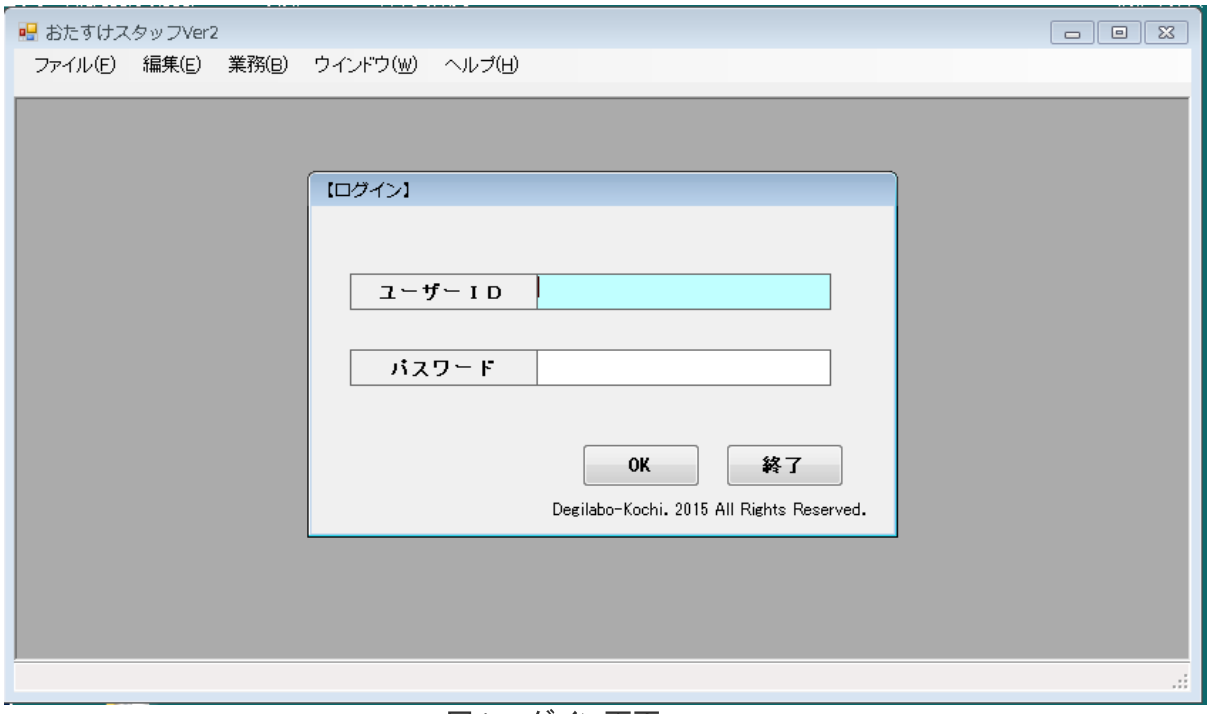


図1 ログイン画面

登録済みの担当者を入力します。
担当者権限マスタに権限者が登録されていない場合は"system/manager"でログインできます。

注意）ﾊﾟｽﾜｰﾄﾞは大文字／小文字に注意してください。

かんたんメニュー作成機能

②メニュー全体が表示されます。

・プログラムの起動方法には三通りあり左「ツリービュー」、正面「SDIメニュー」もしくは上部の「業務(B)」より起動することができます。

起動済み業務は「ウインドウ(W)」>「画面の整列」>「ウインドウ選択」>一覧から選ぶことで最前面に表示することができます。

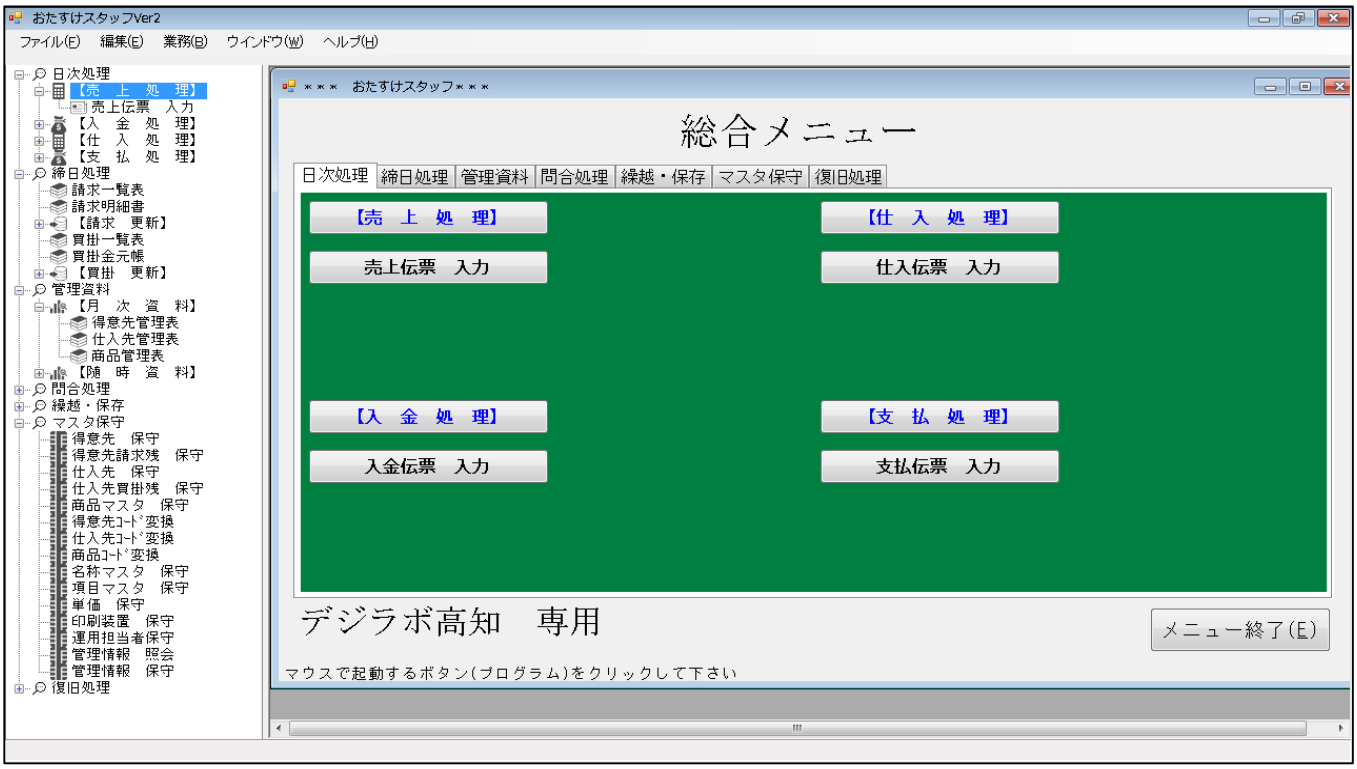


図 2 メニュー全体

「ファイル(F)」>「ログアウト」でさきほどのログイン画面の状態になります。

「ファイル(F)」>「終了(X)」もしくはF12キーを押すとメニュー画面を終了します。

右下の「メニュー終了(E)」はＳＤＩメニューのみを消すことが出来、再度表示する場合は「ウインドウ(W)」>「メニュー画面」を選択すると表示されます。

他 にも「編集(E)」>「画面印刷」を選択しておくことで「Print Screen」キーを押してハードコピーをプリンターに出力ことが出来ます。

上記のようなメニュー画面をいちいち手作業で作っていたらこれも大変ですね。

かんたんメニュー作成機能

ここで
メニュー保守プログラムを起動します。

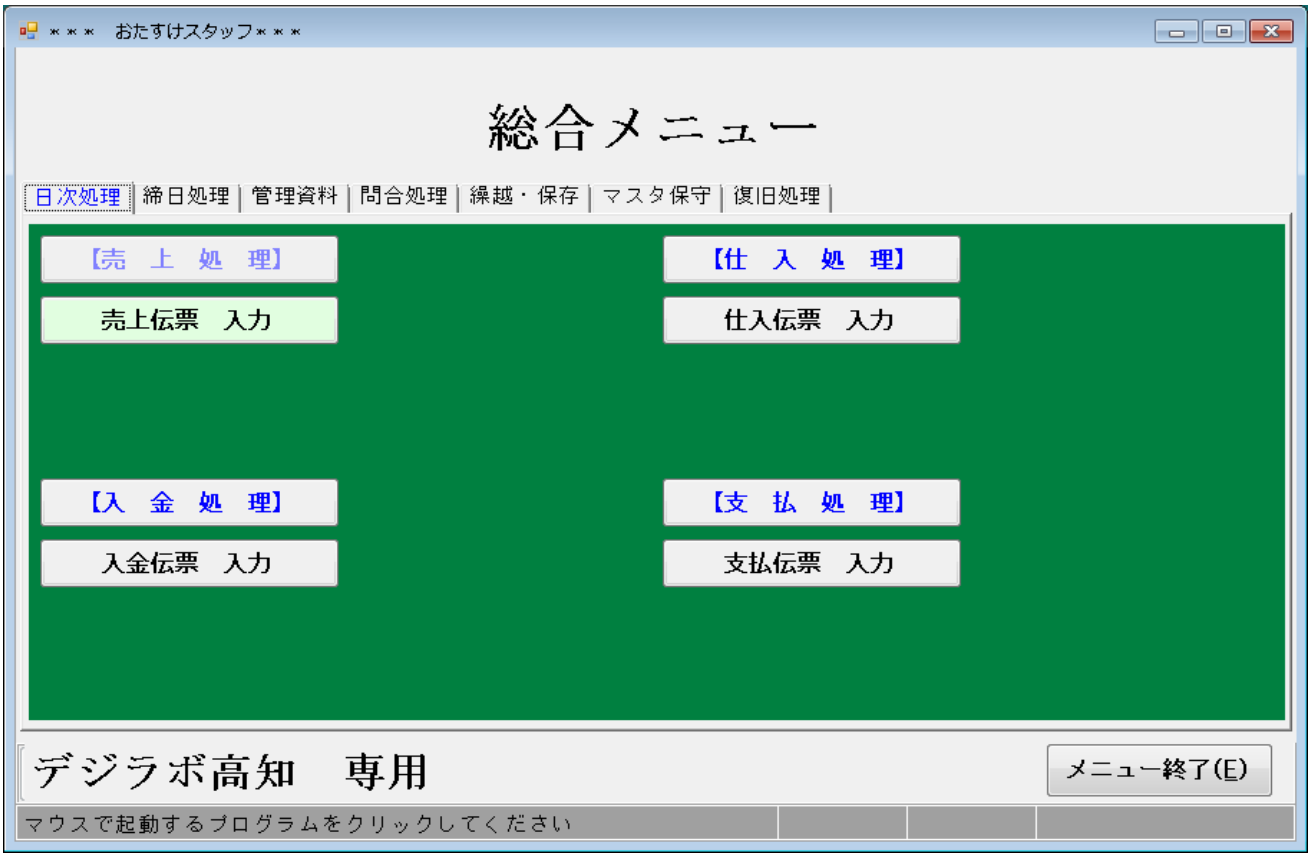


図 3 メニュー保守(1)

さきほどのメニューと同じような画面が表示されます。
ここでマウスの右ボタンをクリックします。

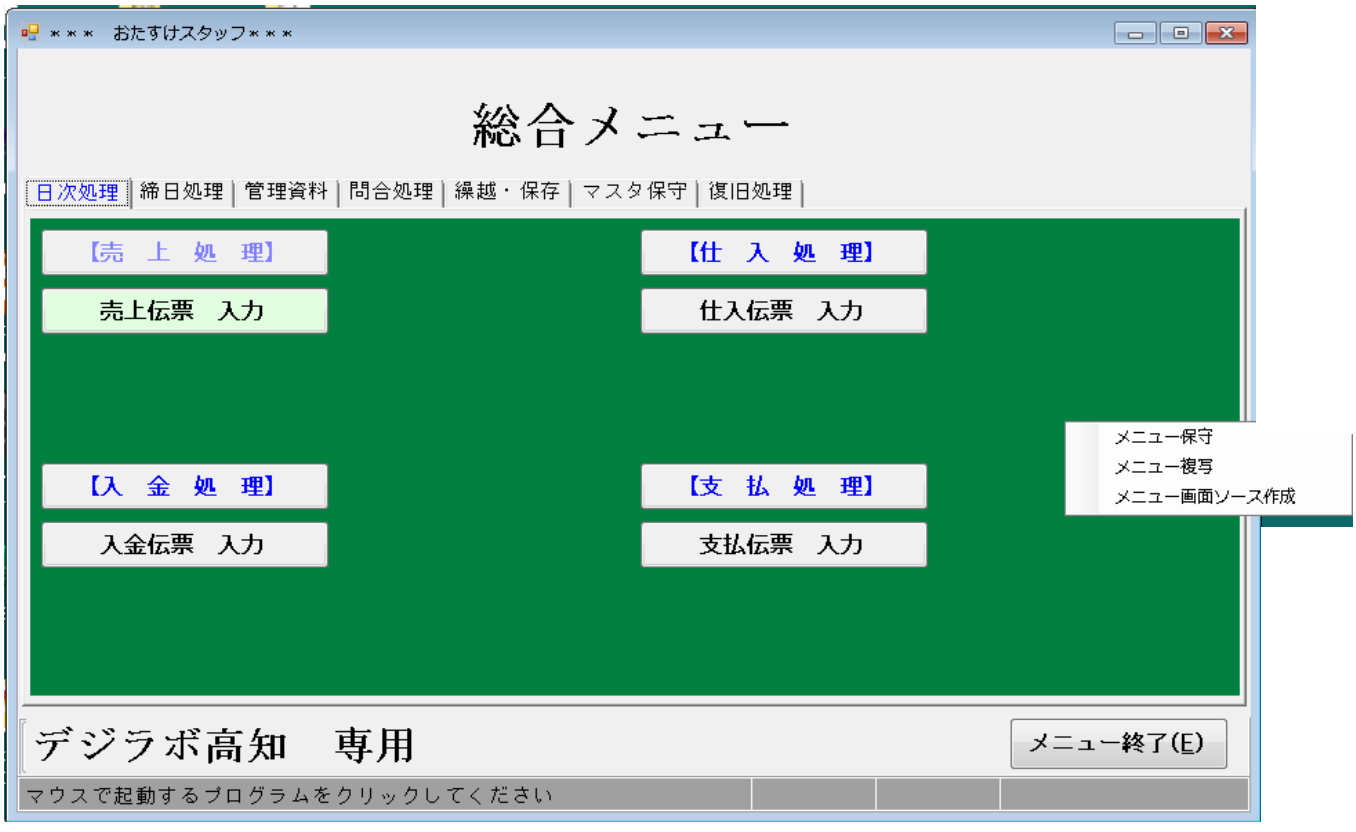


図 3 メニュー保守(2)

かんたんメニュー作成機能

ポップアップ画面より
メニュー保守をクリックします。

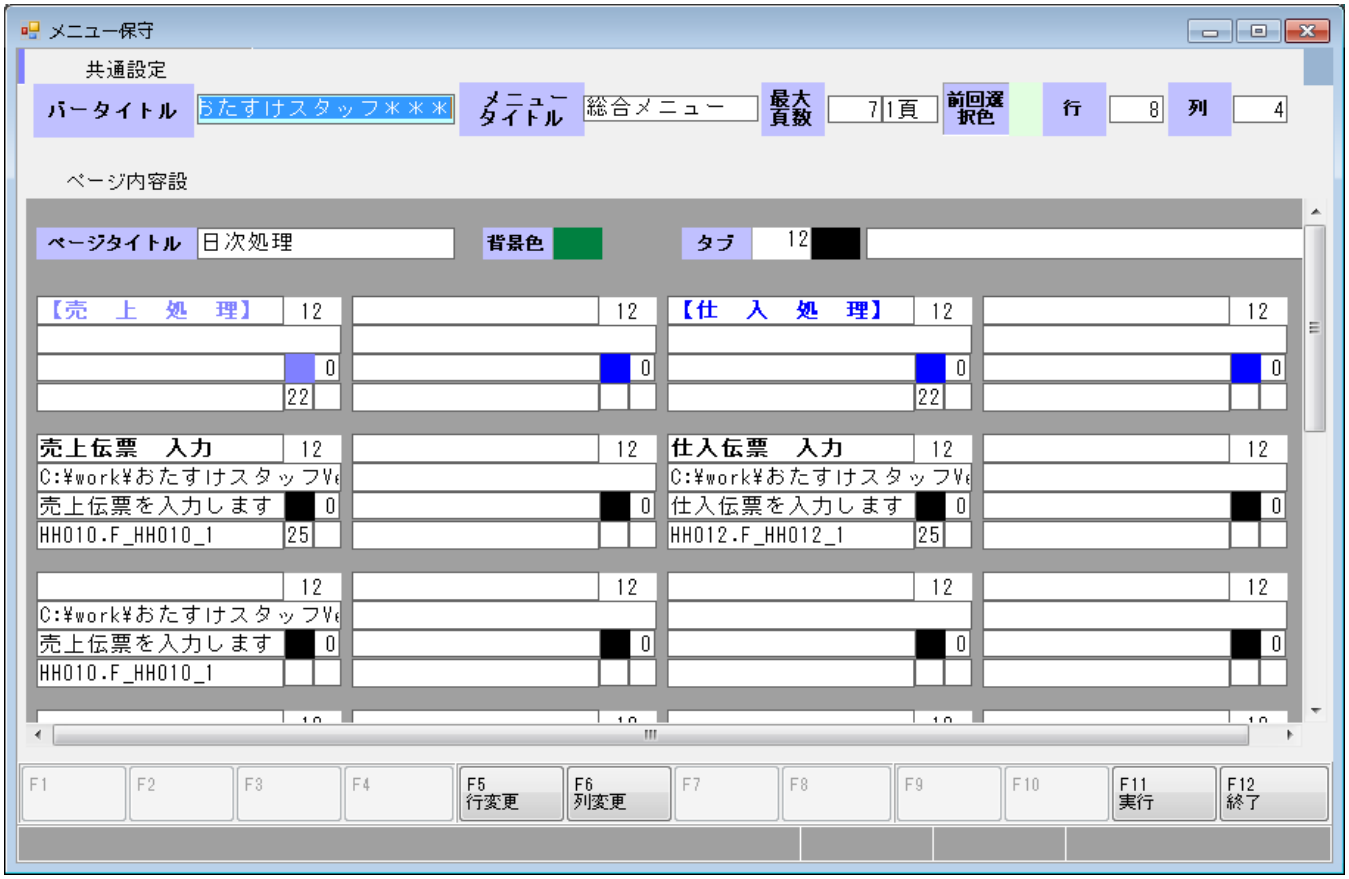


図 3 メニュー保守(3)

現在表示されているページのメニュー項目の保守ができます。

何の設定項目かはカーソルを持っていくとﾂｰﾙﾁｯﾌﾟにより表示確認できます。

最大ページ数がタブ数となります。

行・列を変更することでボタン数（全ての画面に影響します。）を変更できます。

「F11 実行」で書き込みます。（さきほどのメニュー画面に戻ります）

かんたんメニュー作成機能

今度はポップアップ画面より
メニュー画面ソース作成をクリックします。

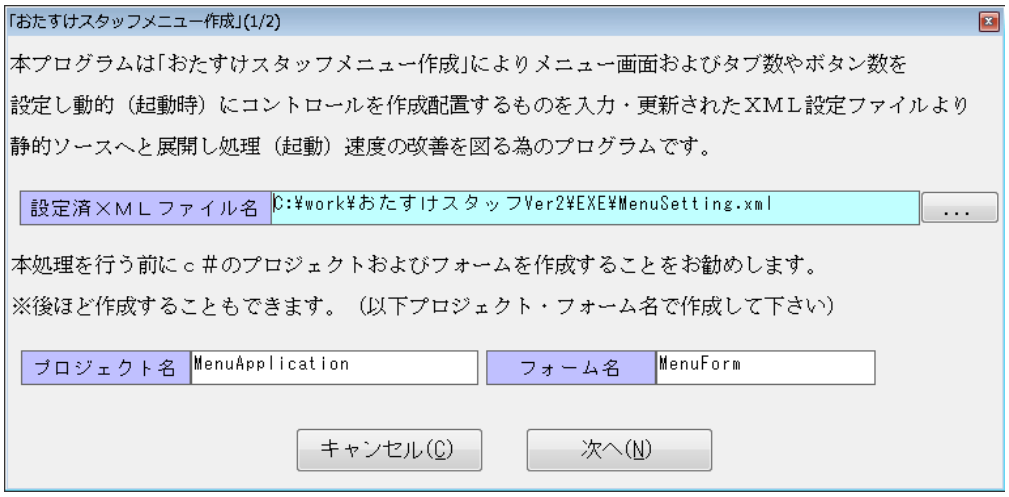


図 4 メニュー画面ソース作成(1/2)

画面のそのままの状態で次へをクリックします。

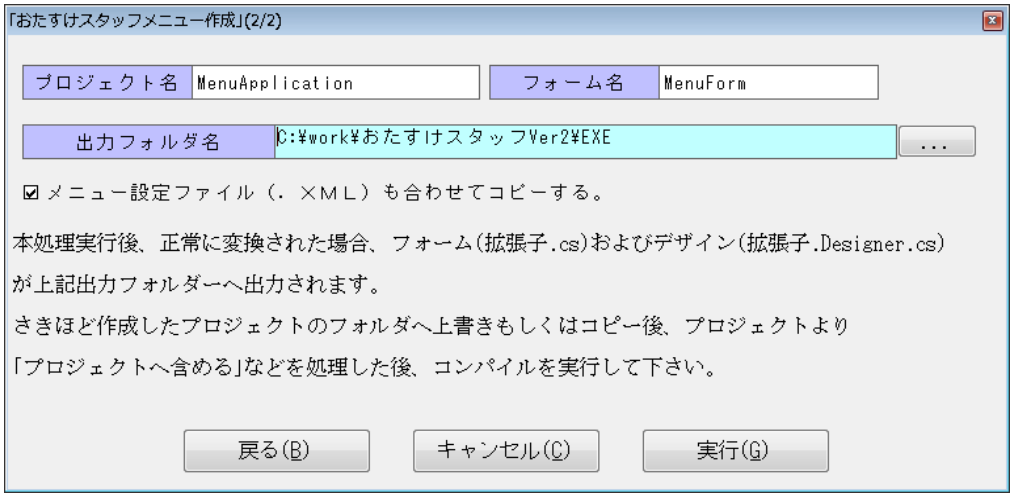


図 4 メニュー画面ソース作成(2/2)

出力フォルダは現在起動中のディレクトリが初期表示されていますのでどこかフォルダを作成して保存してみてください。

変換は正常に終了しました。が表示されるはずです。

出力されたフォルダを覗いてみると以下の３つのファイルがあるはずです。
MenuForm.cs
MenuForm.Designer.cs
MenuSetting.xml
MenuForm.ｘｘの２つのファイルはメンテナンス後のメニューソースとなります。

OtasukeApp_おたすけスタッフソリューション下のフォルダにコピーしてプロジェクトに含める作業を行いビルドします。

MenuSetting.xmlはメンテナンスメニューのＸＭＬ形式のデータ構造といえますが　これも OtasukeApp_おたすけスタッフソリューション下のフォルダにコピーすることで「ツリービュー」や「上部メニュー」の内容を自動的に作成することになるわけです。

こうやって以外と簡単にメニューが作成できることおわかり頂けたかと思います。

エクセル出力機能（DbToエクセル）とは

マスターひとつひとつについて一覧表を作ることは当たり前なのかもしれません。

　これをエクセルへ出力して一覧表の代用として好きなように加工してくれというのも制作側の単なる手抜きとも言えると思います。

出力するたびにセル幅を直すのもひと手間だと思われるからです。

　いずれにしても作る側としては少しでも手間をかけず、エンドユーザーとしてはこれも定型化しときたい要望があります。

　まず抽出画面ですが簡単に作れます。

また表示する項目を選択したりコード項目には検索機能を付けるなどの拡張した方法もとれます。

さきに説明したコードデータコントロールにはテーブルのフィールド情報を全て持っており。

テーブル名から解析した情報から以下の(図1)のような画面が自動的に作られます。

図 1 エクセル出力条件画面（1）

この場合、テーブルの全ての項目が表示されすべての項目が抽出条件の対象となります。

エクセル出力機能（DbToエクセル）とは

また対となったテーブルがある場合にその項目を表示させたいとか、コード項目に対して検索機能を付けたいなどの要求が発生した場合などの対応策として次のリスト1のように出力する項目名を記述する事で可能です。

```
 SearchArgs _SearchArgs;
int iix = 0;
try
{
    //出力項目を以下のとおりセット
    Args.SQL.SQLItem = new List<string>();
    Args.SQL.SQLItem.Add("DURIH_TKCD ");                    //☆CodeX(1)
    Args.SQL.SQLItem.Add("DURIH_DATE ");                    //●Date(1)
    Args.SQL.SQLItem.Add("DURIH_YM ");
                :
                :
                :


    Args.SQL.SQLItem.Add("DURIM_NYUKIN_DNO ");
    Args.SQL.SQLItem.Add("DURIM_SHKB ");                    //☆CodeX(7)
    Args.SQL.SQLItem.Add("DURIM_SBCD ");                    //☆CodeX(8)
    Args.SQL.SQLItem.Add("DURIM_TEKI ");

    Args.SQL.FromSQL = " FROM D_URIKAKE_H LEFT JOIN D_URIKAKE_M ON (DURIH_DNO = DURIM_DNO) ";
    Args.SQL.WhereSQL = " WHERE ROWNUM = 1";
```

リスト1 エクセル出力条件

このような方法で開発者は少ない労力で出力機能を追加することができます。

ただ、もう一つ問題がエンド・ユーザー側の要望である定型化された出力形式が出来るようにしなければなりません。

今度は出力されたエクセルに対して列幅などを変更し保存しておきます。
これを「F1：読込」みＰＦキーを押し読込みんだ後に「F11：出力」で新たに出力します。

すると先ほどのセル幅が保持されているのがお分かり頂けると思います。

エクセル出力機能（DbToエクセル）とは

それともうひとつ機能があります。

テンプレートとなるエクセルシートにあらかじめ以下のようにタイトルと出力項目名をセットしておきます。

出力項目はテーブルのフィールド名の頭に'@'を付加したもです。
例えばフィールド名が DURIH_TKCD であれば　@DURIH_TKCD　となります。

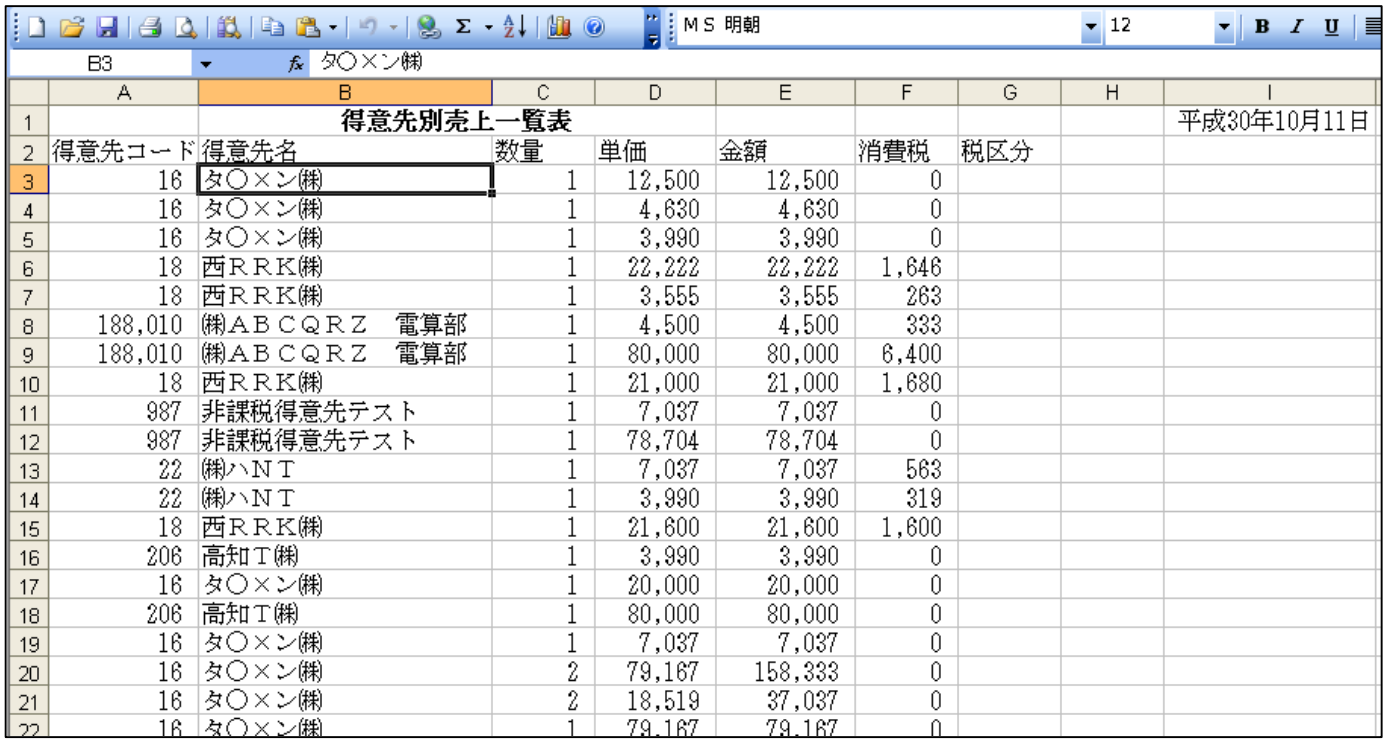
ただこのままでセルに入力するとエラーとなるのでもうひとつ頭にシングルクオンテーションをつけます。

日付などは=NOW()　などの関数でもかまいません。

	A	B	C	D	E	F	G	H	I
1		得意先別売上一覧表							平成30年10月11日
2	得意先コード	得意先名	数量	単価	金額	消費税	税区分		
3	@DURIH_TKCD	@F_GETTK_DURIH_TKCD	IM_SURYO	JRIM_TANKA	@DURIM_KIN	RIM_SZEI	URIM_ZKB		IM_TEKIYOUAYAMARI
4									
5									
6									

シート1 エクセルテンプレート

このテンプレートを入力して出力した結果が以下のようになります。



	A	B	C	D	E	F	G	H	I
1		得意先別売上一覧表							平成30年10月11日
2	得意先コード	得意先名	数量	単価	金額	消費税	税区分		
3	16	タ〇×ン㈱	1	12,500	12,500	0			
4	16	タ〇×ン㈱	1	4,630	4,630	0			
5	16	タ〇×ン㈱	1	3,990	3,990	0			
6	18	西ＲＲＫ㈱	1	22,222	22,222	1,646			
7	18	西ＲＲＫ㈱	1	3,555	3,555	263			
8	188,010	㈱ＡＢＣＱＲＺ　電算部	1	4,500	4,500	333			
9	188,010	㈱ＡＢＣＱＲＺ　電算部	1	80,000	80,000	6,400			
10	18	西ＲＲＫ㈱	1	21,000	21,000	1,680			
11	987	非課税得意先テスト	1	7,037	7,037	0			
12	987	非課税得意先テスト	1	78,704	78,704	0			
13	22	㈱ハＮＴ	1	7,037	7,037	563			
14	22	㈱ハＮＴ	1	3,990	3,990	319			
15	18	西ＲＲＫ㈱	1	21,600	21,600	1,600			
16	206	高知Ｔ㈱	1	3,990	3,990	0			
17	16	タ〇×ン㈱	1	20,000	20,000	0			
18	206	高知Ｔ㈱	1	80,000	80,000	0			
19	16	タ〇×ン㈱	1	7,037	7,037	0			
20	16	タ〇×ン㈱	2	79,167	158,333	0			
21	16	タ〇×ン㈱	2	18,519	37,037	0			
22	16	タ〇×ン㈱	1	79,167	79,167	0			

セルの書式等も適用されているのがわかります。

保存・復元処理

保存復元処理の機能について

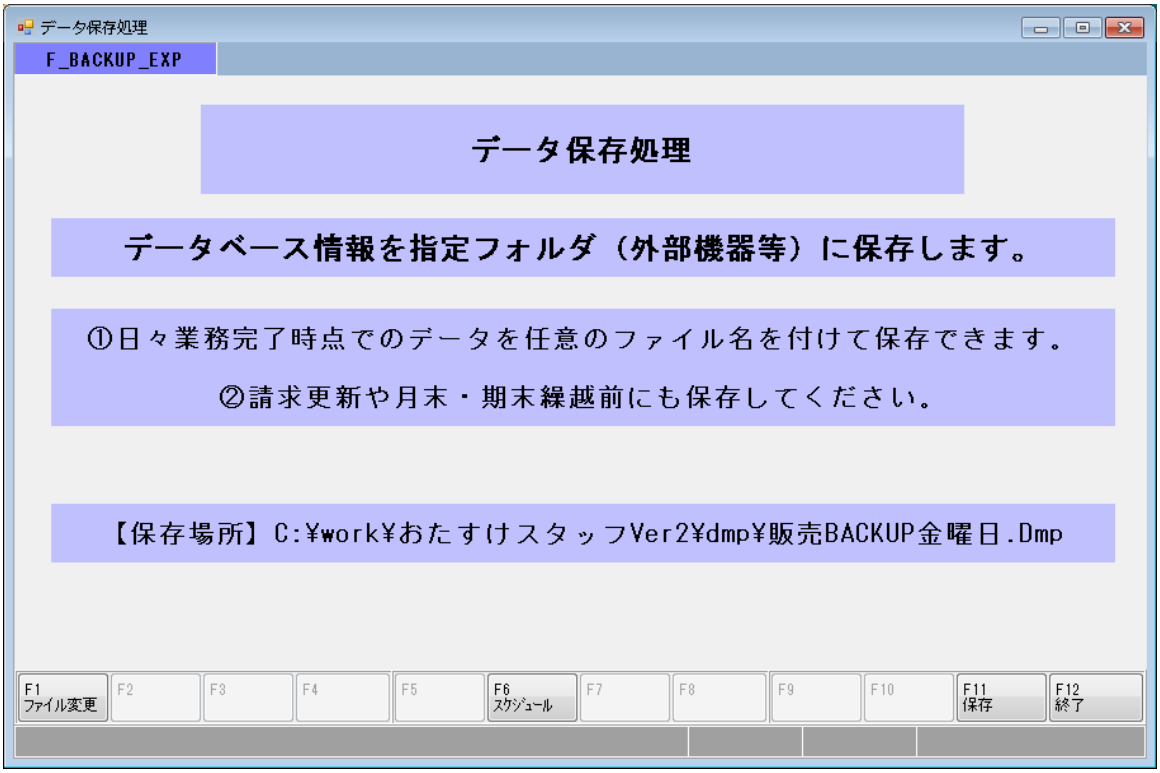


図 1 保存画面(1)

「F6 スケジュール」を押してください。

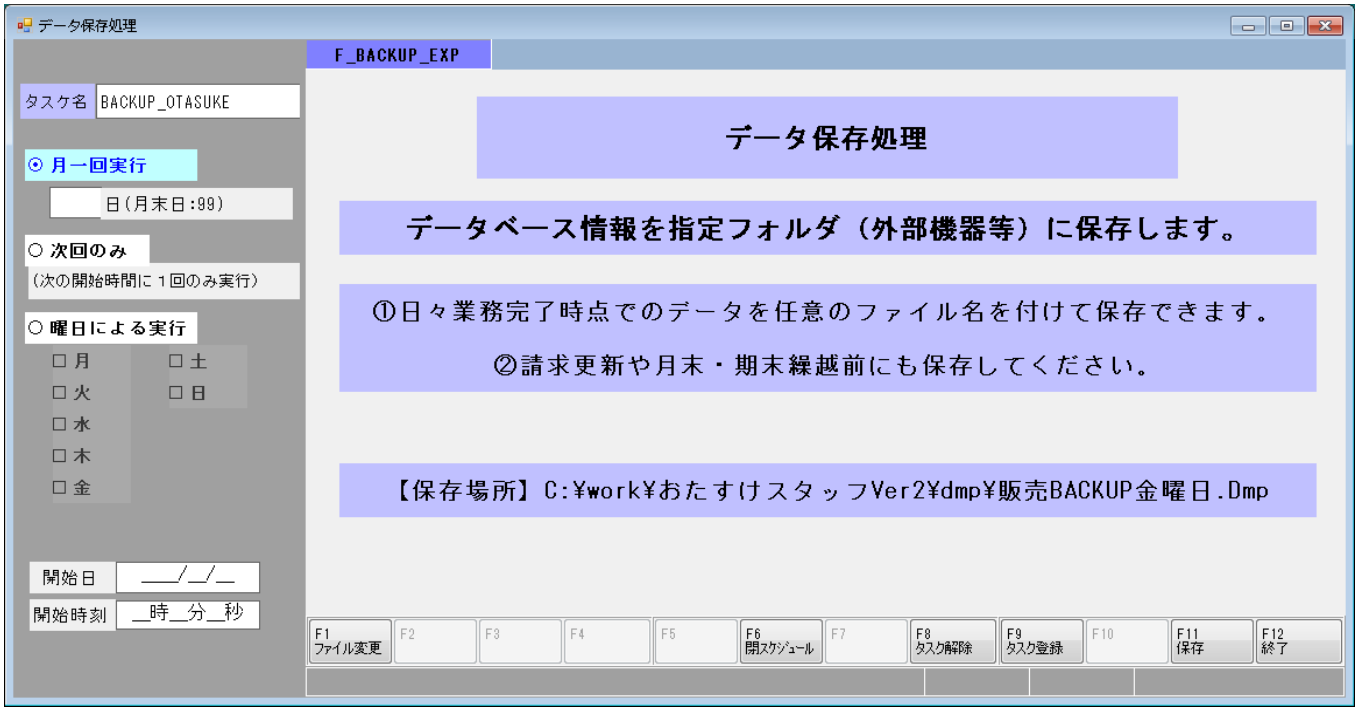


図2 保存画面(2)

左袖がオープンされタスクスケジューラへの登録が可能になります。
例えば曜日による実行を選択し月曜日から土曜日までチェックを付けて「F9 タスク登録」を押します。

保存・復元処理

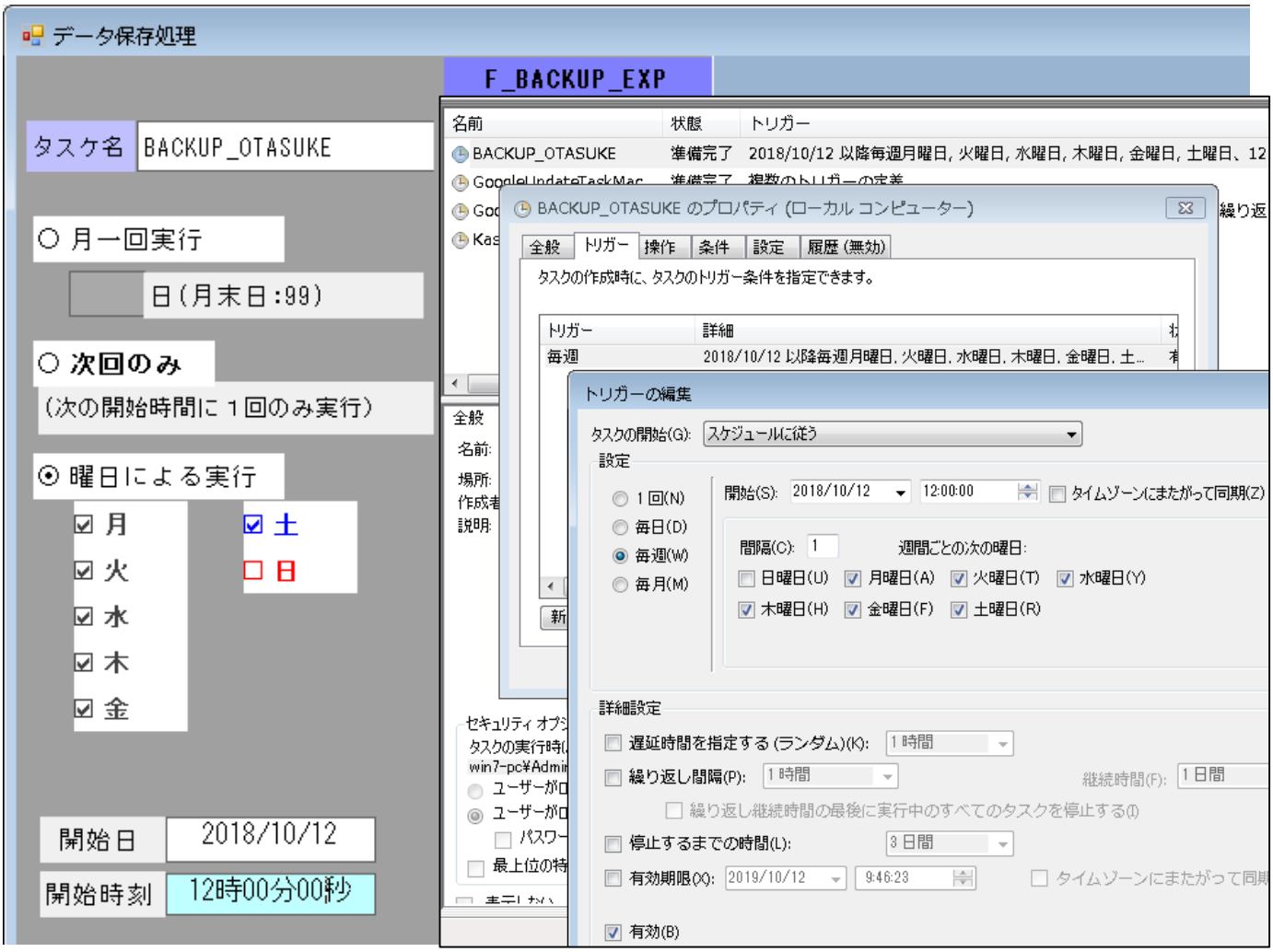


図3 保存画面(3)

タスクスケジューラで確認すると曜日毎に設定されていることが確認して頂けると思います。

保存ファイルは上記の場合、”BACKUPTASK_EXPDAT_金曜日.Dmp”のように指定フォルダーへ出力されるようになります。

その他の指定時にはファイル名の"曜日"部分が"日付"_"時間"のように出力されます。

※1 オラクルの場合、オラクルのユーティリティによるExpにより出力されます。
※2 保存・復元時にはいずれかの端末で業務メニューが起動しているとエラーとなります。
(現在起動中の端末がある場合は接続ログ確認(LOGDISP)で確認することができます。)

保存・復元処理

復元画面については以下(図4)のように１ファイルより選択復元可能となっています。

図4 復元画面

※1 オラクルの場合、オラクルのユーティリティによるImpにより復元されます。
※2 保存・復元時にはいずれかの端末で業務メニューが起動しているとエラーとなります。
（現在起動中の端末がある場合は接続ログ確認(LOGDISP)で確認することができます。）
※3 メニュー項目のレベルで操作可能担当者を制限することができます。

接続ログ表示

接続ログ表示を起動します。
初期値のままエンターキーで進むとプログラム名でエンター後、(図1)のような表示がされました。

日付範囲内、起動されたプログラムです。終了日付時間が表示されていないものはプログラムがまだ起動中であるか何らかの異常終了が考えられます。

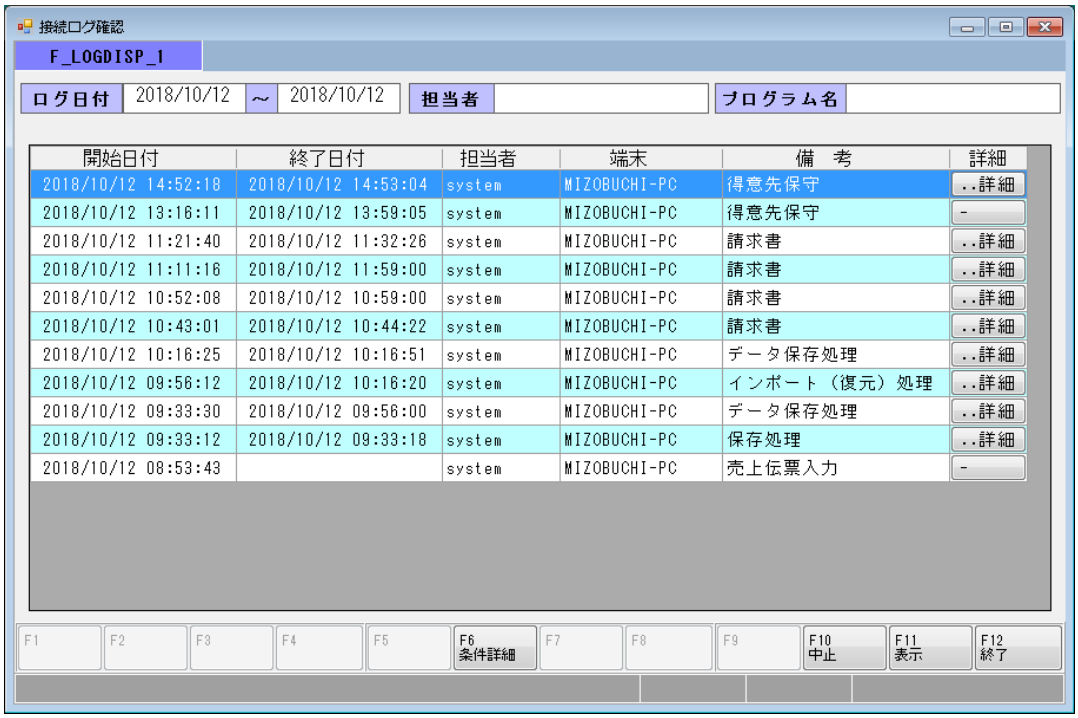


図 1 接続ログ画面(1)

ここでは行の詳細をクリックしてみます。

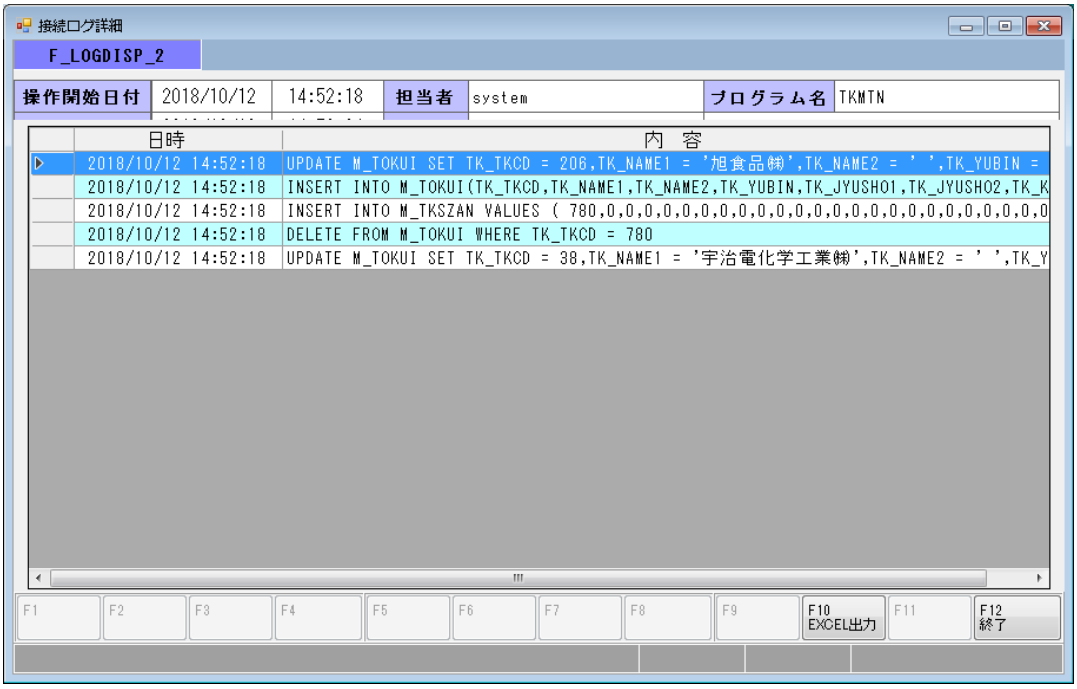


図2 接続ログ画面(2)

この場合、マスタ保守処理中に行われた更新処理がＳＱＬ文として確認することができます。
内容をより詳細に見たい場合は「F11 EXCEL出力」を押下してエクセルにて確認できます。

接続ログ表示

「F12 終了」を押し前画面に戻ります。
ここでは「F6 条件詳細」を押してください。

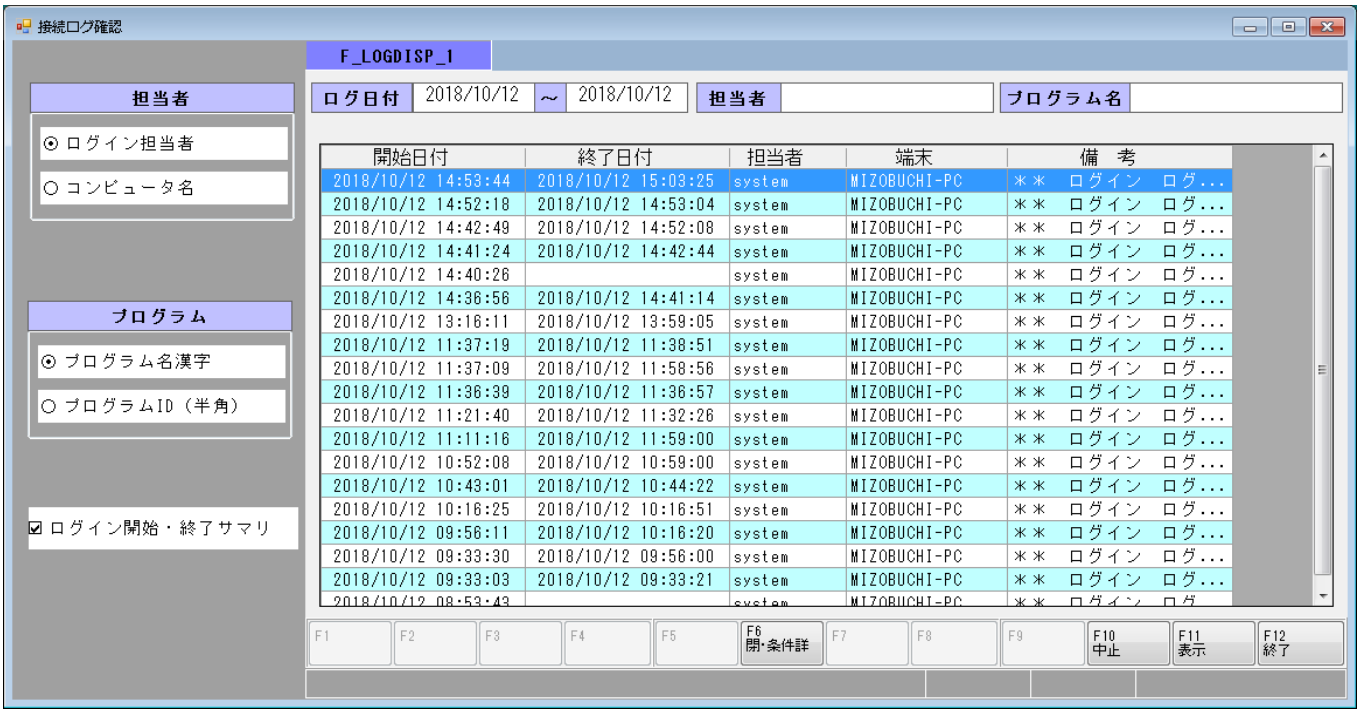


図3 接続ログ画面(3)

左袖がオープンされ抽出詳細条件の入力が可能となります。

ここでは最下部のログイン開始・終了サマリを選択し、「F11 表示」を押してください。

図3のようにプログラム名ではなく備考にはログインログと表示されます。
ログインの開始・終了時間が表示されます。現在から過去へと降順にならんでいますので最上部が最新のログということがわかります。
（ヘッダー部をクリックすることで昇順・降順など切り替えることができます）

最上部で終了時間がない場合は現在起動中であると思われます。更新や管理マスタの排他エラー時には確認してください。

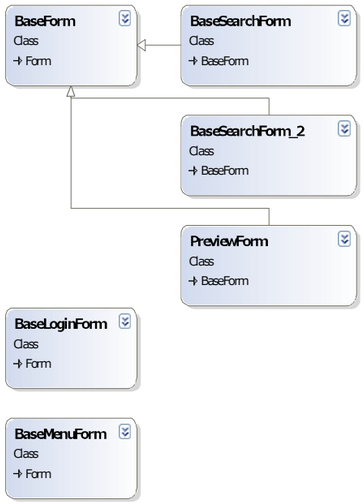
ログ出力に関しては業務プログラムで意識して組み込むことはありません。すべてシステム側で出力します。

※接続ログとは別にエラーログがありますが上記「接続ログ」とは関係ありません。
Try ～ Catch ex As Exceptionによるエラー時に”Log.Out(eLogOutput.エラー, ″error Exception″, ex)”の記述をします。

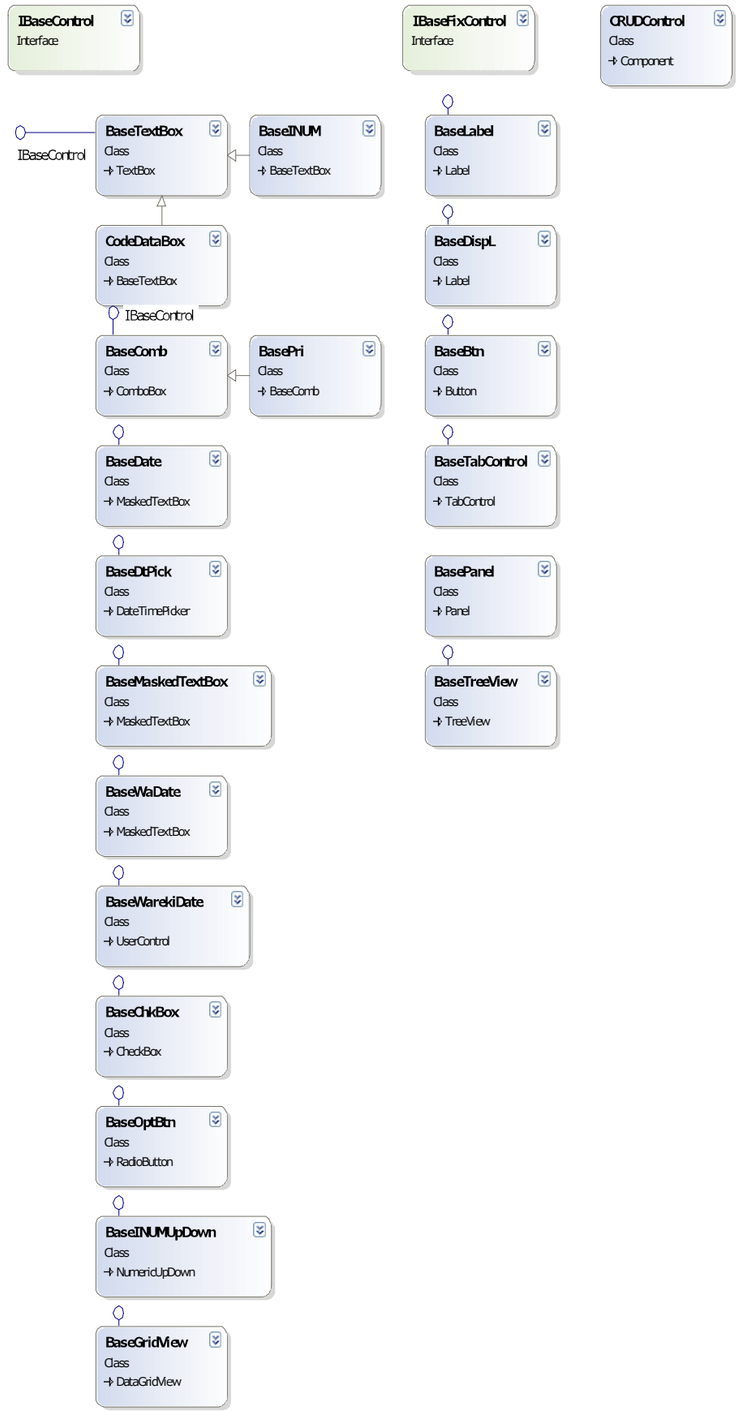
これはエラー時のログを実行時ディレクトリ配下の”LogFile.LOG”ファイルに出力する為の記述であり、ほとんどのメソッド内に記述することをお勧めします。

設計書	システム名称		業務名称		変更日		作成日		Page
	システムID		業務ID		変更者		作成者		

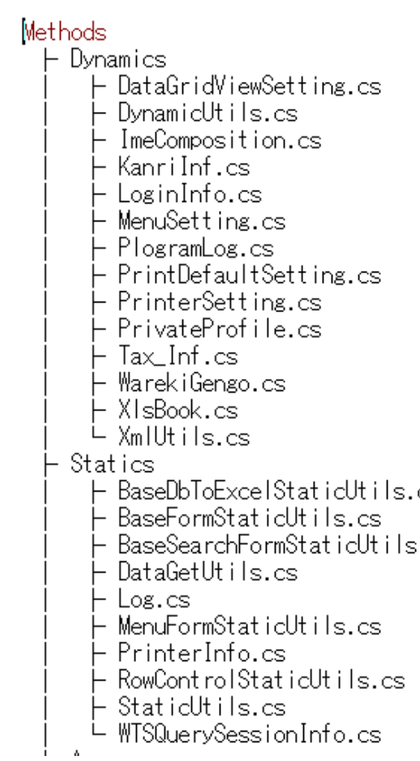
フォーム



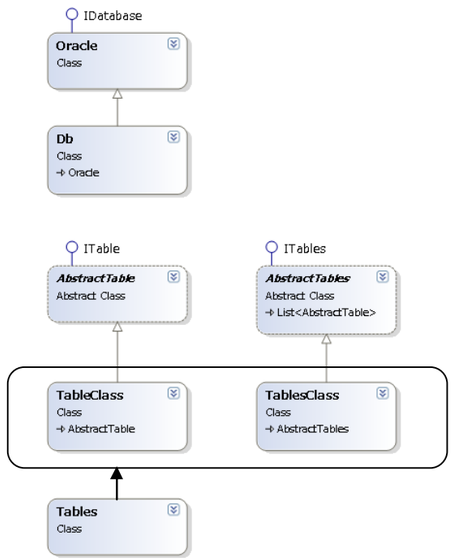
コントロール



メソッド



データベース



おたすけスタッフ Ver2.0

記　号	プログラム名称	dll記号	備　　考		用紙	仕様書	本数	本数S
	入　　力							
HH010	売上入力	F_HH010_1		1	－		1	
HH011	入金入力	F_HH011_1			－		1	
HH012	仕入入力	F_HH012_1					1	
HH013	支払入力	F_HH013_1					1	
	日　　報							
	売上伝票確認票	F_HH010_2			B4横			1
	入金伝票確認表	F_HH011_2			B4縦			1
	請　求　管　理							
HH030	請求書発行	F_HH030_1		1	A4		1	
HH031	請求一覧表	F_HH031_1			B4		1	
SHIME_SEIKYU	請求締日繰越	F_SHIME_SEIKYU_1			－		1	
HH040	買掛金元帳	F_HH040_1					1	
HH041	買掛一覧表	F_HH041_1					1	
SHIME_KAIKAKE	買掛締日繰越	F_SHIME_KAIKAKE_1					1	
	月　　報							
HH050	得意先管理表	F_HH050_1			B4		1	
HH060	得意先売上順位表	F_HH060_1			B4縦		1	
HH038	取引先別請求入金一覧表	F_HH038_1			B4		1	
HH052	商品管理表	F_HH052_1			B4		1	
HH051	仕入先管理表	F_HH051_1					1	
HH620	未入金一覧表	F_HH620_1					1	
	問い合わせ							
HH035	売掛明細問合せ	F_HH035_1						1
HH045	買掛明細問合せ	F_HH045_1						1
	月末繰越							
SHIME_GETUMATU	月末／期末繰越	F_SHIME_GETUMATU_1					1	
	保　　守							
TKMTN	得意先保守	F_TKMTN_1		1	―		1	
SRMTN	仕入先保守	F_SRMTN_1					1	
CD_HEN_TKCD	得意先コード変更	F_CD_HEN_TKCD_1					1	
CD_HEN_SRCD	仕入先コード変更	F_CD_HEN_SRCD_1					1	
CD_HEN_SHCD	商品コード変更	F_CD_HEN_SHCD_1					1	
TKSMTN	得意先請求残	F_TKSMTN_1		1	―		1	
SRSMTN	仕入先買掛残保守	F_SRSMTN_1					1	
SHMTN	商品保守	F_SHMTN_1			―		1	
KOMTN	項目マスタ保守	F_KOMTN_1		1	―		1	
MEMTN	名称マスタ保守	F_MEMTN_1		1	―		1	
KANRI	管理情報保守	F_KANRI_1	ｼｽﾃﾑ区分2以上以外照会	1	―		1	
			ｵﾌﾟｼｮﾝ指定により管理情報照会					
TANTO_KENGEN	運用担当保守	F_TANTO_KENGEN_1		1	―		1	
TNMTN	単価保守	F_TNMTN_1			―		1	
	検　　索							
F_TK_SRC	得意先検索		全てBaseControlに含むBaseSearchFormクラス					1
F_SH_SRC	商品検索							1
F_KO_SRC	項目検索							1
F_ME_SRC	名称検索							1
URI_SRC	売掛明細検索							1
SHI_SRC	買掛明細検索							1
	更新取り消し							
CANCEL_GETUMATU	月末・期末更新取消	F_CANCEL_GETUMATU_1					1	
CANCEL_SEIKYU	請求締日更新取消	F_CANCEL_SEIKYU_1					1	
CANCEL_KAIKAKE	買掛締日更新取消	F_CANCEL_KAIKAKE_1					1	
	その他							
LOGDISP	ログ確認	F_LOGDISP_1		1			1	
BACKUP_EXP	データバックアップ	F_BACKUP_EXP_1		1			1	
RESTOR_IMP	データインポート	F_RESTOR_IMP_1		1			1	
OtasukeApp	メインメニュー	F_OtasukeApp_1		1			1	
PRMTN	印刷装置保守	F_PRMTN_1		1			1	
MenuMTN	メニュー保守	F_MenuMTN_1		1			1	
	※ dll記号名 の色付きは2018年10月現在制作済みです。							
				14			39	10
							計(	29

特記・制限事項

オラクルに関してははオラクルexpressまたは有料バージョンのどちらに向けても動作可能です。

ＳＱＬサーバー等、他のデータベースについてもオラクルと同様な（カプセル化された）インターフェースを開発することで業務への影響をしない開発が可能です。

各ツールは主にＣ＃.NETで作られており一部のコントロールはＶＢ．NETで作られています。開発環境は３２ﾋﾞｯﾄ(x86)環境が必要です。
開発業務も同様にいずれかもしくは両方の言語で作成可能です。

本システム開発はWindows10 Hyper/VでWindows7環境で開発しております。

外部のコントロールは「ＶＢリポート」(アドバンスソフトウェア)を使っていますが、テスト時はこのコントロールの試用版で十分です
正式リリース時には、一括コンパイルするなどの方法(１開発ライセンス)で対応することが可能です。

ＶＢリポートは設計・レビュー時などのエクセルシートをそのまま開発工程へ採用することができ、リポートを新たに作成するなど余計な作業を省くことができる優れたツールです。

※体験版実行時には体験版である旨のメッセージ表示と印刷部にも体験版メッセージが出力されます

他にはエクセル操作の為にNPOI(Officeのドキュメントの操作を行う事が出来るApache POI(Javaライブラリ)の.NET用に移植されたライブラリ)などを使用しています。

開発環境は32ﾋﾞｯﾄ環境ですが、Visual Studioでは32/64ﾋﾞｯﾄ環境へのターゲット指定が可能である為、バッチコンパイラーを用意しておりいずれの環境へも労無く作成できます。
動作検証においてもいずれの環境でも現状、問題なく動作しております。

Visual Studio のバージョンは　2010 Profetional を使って開発しています。
Visual Studio Expressでの動作検証はしておりません。