# CRT画面帳票作成ツール SimpleMaster

## 取り扱い説明書



#### 目次

	ページ
1 はじめに	1
2 レイアウト作成	
2 - 1 SimpleMasterの起動	2
2-2 メニュー画面の作成	4
2 - 3 売上げ帳画面の作成	11
2 - 4 伝票入力画面の作成	21
2 - 5 帳票レイアウトの作成	27
2-6 制御用画面の作成	32
3 ロジック作成	
3 - 1 プログラムの構造	36
3 - 2 共有データ領域	37
3 - 3 最小限のアプリケーションプログラム	39
3 - 4 コンパイラの設定	4 1
3 - 5 プログラムの実行	47
3-6 売上げ帳画面処理プログラム	50
3 - 7 伝票入力画面処理プログラム	55
3-8 帳票印字処理プログラム	58
3 - 8 制御用データ処理プログラム	61
4 デモ用プログラム	
4 - 1 ソ <b>ー</b> スプログラムの修正	63
4 - 2 コンパイラの設定	64
4 - 3 デモ用プログラムの実行	64
5 スプレッドシート	
5 - 1 概要	65
5 - 2 レイアウト作成	67
5 - 3 スプレッドシートの表示	72
5 - 4 表示データの設定	74
5 - 5 サンプルプログラム	76

#### Windows 版 と Linux 版の相違点

レイアウト作成プログラム SimpleMaster ではダイアログボックスの表示形式などに相違点がありますが、 設定する項目、内容は同一です。 作成したアプリケーションプログラムの実行内容は同一です。 その他、明確な相違点がある場合には説明書に記述しています。

#### 改訂記録

2004/01/15 Ver1.01 新規作成 2004/06/01 Ver1.11 スプレッドシート機能追加 / Linux 対応

Microsoft、Windows、Windows NT、Visual C++、MFC などは Microsoft Corp.の米国およびその他の国における登録商標です。 SimpleMaster は (㈱フジテクニカの登録商標です。

#### <u>1 はじめに</u>

SimpleMaster は CRT 画面、帳票印字用のレイアウト作成を行います。 アプリケーションプログラムでは SimpleMaster で作成したレイアウト情報を参照して CRT 画面処理、帳票印刷 を簡単に行うことができます。

SimpleMaster は次のような機能を持っています。

- (1) プロジェクト単位での管理
   SimpleMaster を使用して作成するレイアウト情報とアプリケーションプログラムをプロジェクト単位で管理 しています。
   プロジェクトにはつぎのようなファイルが含まれます。
   プロジェクト情報ファイル
   レイアウト情報ファイル
   リースプログラムファイル
   表示色データ、文字フォントデータなどレイアウト作成時、アプリケーションプログラム実行時に参照 されるデータファイル。
- (2) レイアウト作成
- レイアウトは CRT 画面、帳票を模擬した作業領域に 直線、矩形などの基本図形 固定文字列、可変文字列、数値データ スクロールバー、ボタンなどのコントロール などを貼り付けることで作成します。 レイアウトの作成はマウス操作と簡単なキーボード入力で行います。
- (3) スプレッドシート機能 簡単な設定で表を作成することができます。キーボード入力、画面のスクロールなどが自動的に行われます。
- (4) マン・マシン部を構築するスタティックライブラリ
   SimpleMaster で作成したレイアウト情報はアプリケーションプログラムに組込まれたスタティックライブラリ
   SimplePack で処理します。
   SimplePack には CRT 画面表示、帳票印刷以外に、マウス操作、キーボード入力などマン・マシン部を 構築するための機能が含まれています。
   アプリケーションプログラムでは簡単な関数を使用して SimplePack と情報の受渡しを行うだけで CRT 画面、帳票印刷の処理を行うことができます。
- (5) ウィンドウアプリケーション開発に必要な機能を提供
   アプリケーションプログラムの起動、停止処理
   ディスクファイルの操作
   CRT 画面のハードコピー
   などウィンドウアプリケーションの開発に必要な機能が SimplePack に実装されています。

[取り扱い説明書]では簡単なアプリケーションを想定してレイアウト作成からロジック作成までの手順を説明します。

詳細な仕様はレファレンスマニュアルを参照してください。

#### <u>2 レイアウトの作成</u>

SimpleMaster を使用して簡単な メニュー選択画面 を作成してみます。

#### <u>2-1</u> SimpleMaster の起動

SimpleMaster を起動するには次の3種類の方法があります。

- (1) [スタート]メニュー [プログラム] [SimpleMaster] [レイアウト編集] の順にメニュー を選択して起動します。
- (2) デスクトップ上に配置された SimpleMaster のアイコンをダブルクリックして起動します。
- (3) エクスプローラで プロジェクトファイル(拡張子:prj) または レイアウトファイル(拡張子:ptn) をダブルクリックすることで起動します。

#### <u>2-1-1 SimpleMaster</u> 起動時の画面

メレイアクト作業 [SimpleDemo] ファイネ 単単 単語	a Di si
CRT画面帳票作成ツール	
SimpleMaster version	
FUJI JUJU	ת

[ファイル]メニュー [プロジェクトの新規作成] を 選択します。

SimpleMaster を2回目以降に起動した場合は、前回に開いた プロジェクトが選択された状態で起動します。

プロジェクトファイルを直接指定して起動した場合には、指定 されたプロジェクトが選択された状態で起動します。

レイアウトファイルを直接指定して起動した場合には、レイアウト が属するプロジェクトが選択され、指定されたレイアウトが読込ま れた状態で起動します。

<u>≪レイアウト作成</u> ファイル 編集 表示 プロジェクトの新規作成 プロジェクトを開く プロジェクトの修正 ロジック作成 レイアウトの新規作成 レイアウトを同く レイアウトを閉じる レイアウトの上書き保有 出力 , ÷ EXCEL 終了

#### <u>2-1-2 プロジェクトの新規作成</u>

SimpleMaster では、適応するアプリケーションごとにプロジェクトを作成し、レイアウト情報、ロジック 情報を管理しています。

プロジェクトの内容は後で修正することができますので、ここでは メニュー画面 を作成するために 最小限の設定のみを行います。

プロジェクトの新羅作成			,
余祭  現りんやり 子りんやり その他			
加尔外名			
SimpleDemo	1		
クロジェクト情報指統先フォルダー			
D VProgram Files#Simple Master#Sample VPtm	安東		
ロジック情報検納先フォルダー			
O VProgram Files/VSimple Master/VSample VProg	- 元史		
西型の解像回 1024 × 768 ▼ 今国籍言語対応 ● 16回 C 26国 C 36国 表示色切替え ● 2所統 C 4所統		この3項目を設定すれに レイアウトの作成を開始 することができます。	ţ
「 VB用ロジック作成 表示色設定   フォント設定   既定値に戻す			
		1	at Rin

 プロジェクト名
 プロジェクトの名称を設定します。

 ここでは「SimpleDemo」と設定します。

 プロジェクト情報格納先フォルダー名
 レイアウト情報など SimpleMaster で生成、編集するファイル

 を格納するフォルダーを指定します。

 ロジック情報格納先フォルダー名

 ロジック情報(アプリケーションプログラム)を格納するフォル

 ダーを指定します。

 SimpleMaster ではレイアウト情報とプログラムの間で情報を

 共有するためにヘッダーファイル、ソースプログラムファイル

 環境設定ファイルを作成し、ここに格納します。

フォルダーの [変更]ボタン をクリックするとフォルダーの選択ダイアログボックスが開き、既存の ドライブ、フォルダーより選択することができます。(新規作成もできます。)

フォルダの副訳		x
D: [Work]	DVSIngleVTestProguranVPtm	
er ta	D     BockUp     BockUp	ここに新規に作成する フォルダー名を入力する ことができます。

指定されたフォルダーが存在しない場合には生成されます。

#### レイアウト作成

- <u>2-2 メニュー画面の作成</u>
  - 2-2-1 レイアウトの新規作成

プロジェクト登録後、 [ファイル]メニュー します。

- 親ウィンドウ / 子ウィンドウ / その他
   SimpleMaster ではフルスクリーンで表示 される CRT 画面を 親ウィンドウ、フルスク リーンよりも小さい大きさで表示される CRT 画面を 子ウィンドウ と称しています。
   [その他] は帳票用のレイアウトなどを作成 するときに使用します。
- スプレッドシート 表形式の画面を作成します。 (5 スプレッドシート を参照してください。)



#### ウィンドウの構成

SimpleMaster では次のようなウィンドウの構成を持つアプリケーションに適応することができます。



最上位ウィンドウはフルスクリーンで作成しなければなりません。 その他のウィンドウの大きさは自由に設定することができます。(フルスクリーンも可) システムウィンドウは、バージョン情報画面、エラーメッセージ表示画面などに使用します。

#### <u>2-2-2</u> レイアウトの初期設定

[レイアウトの新規作成] で 親ウィンドウ を選択すると、レイアウトの詳細設定ウィンドウが表示され ます。 この時、背景にはプロジェクトで設定した親ウィンドウのテンプレートが表示されています。

-1791AFA (SimpleDe	no / #/#/6.#]	-10
		2003年12月17日(木)10時::
		-
	マウスカーンの形成 F 先印 C 十平 C 初期計     ギー人力時 前景生 0 単発生 0      脱生 取(病 F 記録所集集 下 約7時集集	

レイアウトは後で簡単に修正できますので、ここでは タイトル名称 のみ設定します。 タイトル名称 はレイアウト情報を保存するときのファイル名(拡張子は ptn) に使用されます。 タイトルに「メニュー画面」と設定し、[起動時画面] をセット後 [登録]ボタンを押します。



<u>2-2-3 メニュー画面の編集</u>

今回作成するデモ用アプリケーションの画面構成を次のようにします。



メニュー画面にはそれぞれの画面を開くためのボタンを配置します。

CRT 画面上にボタンを配置するために、[編集]メニュー [ボタン]を選択します。

元に買す	作業領域上でマウス右ボタンな
表示自設定 支字フォント設定 マーク設定	クリックすることでも、[ホタン]る 配置するためのメニューが表示 されます。
短形枠 重結 矩形 横円 棒グラグ	<i>▲レイアウト作成(デモ用画面)メ</i> 77 <i>18 置集 表示 ヘルブ</i> メニュー画面
国定文字列 可変文字列 数値データ	<b>用表示</b> ズーム
<b>ボタン</b> ラジオボタン	方服表示
+1>9#>97 E>k=>7 -9 290-k/- 2E>#9>	短形神 単雄 短形 相円 種グラグ
	国定文字列 可変文字列 数値データ
	ボタン ラジオボタン チェックボックス ビットマップ マーク スクロールバー

冬 メニュー画面	*97		×
× 7 0	始点	66 / 182 F 🗹	2 <b>P</b>
	鞍点	261 / 178	_33
/ <b>•</b> •	#30/0文字1		-
	市920文于2	- Aug.	<u> </u>
	市部以向文字3		*
	7421	MB ゴシック 28×14 ボール	- N
	7530101946	6 1	
ここまで移動し	左上:石下林	0 0	
	市ちつの表面	0	
	ボタンの文字	0	
	洞去巴	0	
	市场之文子列	●中央 「指定」0	-
	77777574-		•
	処理種別	(40)	-
	1000-544	1	-
	100 B 1 - 1-	-	*
		눈의	
		1 建合肥玉	
		文字列一覧へ登録	

[ボタン]の表示位置と大きさを決めるため、始点でマウス左ボタンを押し、終点で離します。 表示位置や大きさは後で簡単に修正できますので、ここでは適当な位置と大きさで作成します。

次に詳細設定ダイアログボックスへの入力を行います。

(ダイアログボックスの入力後にボタンの表示位置を設定することもできます。)

ボタンの文字に [売上げ帳] と入力します。

ボタンの領域の大きさに合わせてボタンの文字の表示に使用する文字フォントを選択します。 次にボタンの色などを設定します。



ボタン文字の表示色は、ボタンの操作可能時()、操作不可時()の2種類を 設定することができます。 0



#### 処理内容の設定

つぎに、ボタンをマウス左ボタンでクリックした時の処理を設定します。

And there are the i	and the second se		
10.00 名件 10.00 名件	(なし) 種ウインドウヘメッセージ通信 子ウインドウ1を開発	-	ここをマウス左ボタンで クリックすると処理項目
1000-P	システムウィンドウをMK 数上位ウィンドウをMK 目ウィンドウをMBSを	-	の一覧が表示されます。
	「文字則一覧へ登)		
	「二 次国からの初期値として保	存	ボタンを押したとき、ラ
			[最上位ウィンドウを閉
処理種別	最上位ウィンドウを開K	•	
画面名称		*	次に画面名称を設定
歳のロード		*	現時点では画面を作
	0 🗄	_	[後で設定] を ON は
	▼ 限で設定		
	□ 文字列一覧へ登録		
	□ 次回からの初期値として保	存	

ボタンを押したとき、売上げ帳画面を開くために [最上位ウィンドウを開く] を選択します。

次に画面名称を設定しなければなりませんが 現時点では画面を作成していませんので [後で設定] を ON にしておきます。 設定内容の登録



[売上げ帳]ボタン と同じ手順で [設備図] のボタンを作成します。 また、アプリケーションプログラムを終了させるため、[終了]ボタンを作成し、処理種別に [システム 終了] を設定します。

一直面	2004年06月03日(未
	<i>69&gt;</i>
THE LAND	16.0 prz / 720 🕅 108
50上17 報	Mr.d. 962 / 760
	#320文字1 拼7 王
	#3/0文字2 +
前小/# (四)	#3>0五年3
EX IM ISI	245/# MIS 35/9 20×10 # - M/ •
	#30-0.09編  4 三日
	本上市干部 10 51
	#306歳間 12
	ポポンの文字 13 14 14 14 14 14 14 14 14 14 14 14 14 14
	(根本色) 0
	#85次平列 《中央 ( 指定 )0 出
	2019/2014- F12 💌
	福祥雅和 システム終了 王
	·测量名称 •
	ALEI-F *
	P. 土
	T Heredara
	□ 文字列一號へ登録
	「「活動からの特別値として保存



2-2-4 メニュー画面の登録

編集が終わったメニュー画面を登録します。

[ファイル]メニュー [レイアウトを閉じる]を選択します。

レイアウト保存用の画面が表示されます。 [保存]ボタンを押してレイアウト情報をハードディスクに 保存します。

モジジノ情報		
シテナンス	ここに保存する時の	
	名称を入力することもできます。	

レイアウト情報を保存するとき、ファイル名として画面のタイトル、拡張子として ptn が使用されます。 同じタイトルの画面が複数存在する場合などでは、ファイル名を修正することもできます。 ファイルの保存先はプロジェクト設定時に指定されたフォルダーが使用されます。

#### <u>2-3</u> 売上げ帳画面の作成

[売上げ帳] 画面を作成します。

メニュー画面の作成時と同様に [ファイル]メニュー [新規作成] [親ウィンドウ] を選択し、 タイトルに「売上げ帳」と設定します。

レイアウト編集時に、方眼を表示したり、 拡大・縮小表示を行うことができます。 マウス右ボタンメニュー または [表示] メニューで設定してください。

[売上げ帳] では表形式の画面を作成 しますので、罫線やデータの表示位置を 等間隔に配置することになります。 SimpleMaster では表示位置は方眼に 合わせて調整されます。

方眼の間隔は [表示]メニュー 「方眼 設定] で任意の大きさに設定することが できます。 3482

襖

縦



#### 2-3-1 表の作成

表の作成には 矩形枠 を使用します。 矩形枠 は塗りつぶされた矩形と上下 左右の罫線で構成されます。 矩形枠 は方眼の内側に描画されるので 上下左右に複数個配置することで表を 作成することができます。 矩形枠の内部には、固定/可変文字列、 数値データを配置します。



売上げ帳の表示項目は次のとおりとします。

No.	項目名	データ形式	桁数
1	日付	可変文字列	7
2	品名	可変文字列	24
3	単価	数値データ	8
4	数量	数値データ	8
5	合計	数値データ	10

#### 2-3-2 表示色の設定

売上げ帳では次の表示色を使用します。

表示色番号	項目名
10	ボタン 左・上枠 / スクロールバー 左・上枠
1 1	ボタン 右・下枠 / スクロールバー 右・下枠
12	ボタン 表面   / スクロールバー 内部
13	ボタン 文字   / スクロールバー スライダー
14	ボタン 文字 (操作不可)
2 0	項目名称 文字
2 1	データ 文字
22	項目名称 背景
23	データ 背景

[編集]メニュー [表示色設定]を選択し表示色を設定します。



SimpleMasterをデフォルトの状態で起動すると、表示色番号 0~19には次の目的で使用する表示色が 設定されています。

番号	項目名	-141
0	親ウィンドウ タイトル領域 文字の表	
1	親ウィンドウ タイトル領域 背景色	
2	親ウィンドウ 作業領域 背景色	
3	親ウィンドウ ボタン領域  背景色	
4	子ウィンドウ タイトル領域 文字の表	
5	子ウィンドウ タイトル領域 背景色	
6	子ウィンドウ 作業領域   背景色	
7	子ウィンドウ 枠の表示色	
8	キーボード入力時 文字の表示色	
9	キーボード入力時 文字の背景色	

番号	項目名
10	ボタン/スクロールバーで使用
11	ボタン/スクロールバーで使用
12	ボタン/スクロールバーで使用
13	ボタン/スクロールバーで使用
14	ボタン/スクロールバーで使用
15	メンテナンス画面で使用
16	メンテナンス画面で使用
17	メンテナンス画面で使用
18	メンテナンス画面で使用
19	エラーメッセージで使用

#### <u>2-3-3 表示位置の設定</u>

1行分のデータを画面上に配置し、表示位置を決めます。

7748 ## ## 187	T ALL IN BI			
売上げ帳			2003 4	11月17日(月)10時:51分 🚟
	******			*******
7482 ×	可度文字列		原銀データ	2
18 10 · Frit	位置	48 / 128 1 100	ttitt	516 / 128 F EDE
12 10 + 1'st	文字数	P ====	表示标题	F 0 F 0
表示色 10	2825	M8 3000 24×12 #-AF	2424	MS 1590 24×12 #'-4+'*
スケール ダ メノマ座標	和荣色	20 0	料景色	23 0 0
○ グリッド番号	背景色	20 0	宣景色	20 0
REE REA	湖古县	「「 教景色有助 」「 点滅 の	海古色	○ 皆豊色有効 □ 点漢 ○□
表示する文字フォント		[ 日本語入力		
を 24×12ドットとし	カーソル			C arrangem
白眼の表示問隔を	热理被刑	自らったわたいせった法信・		- 10##
	調整名称	*	カージル	
	備身白ード	HEUNE .	地理權別	白からどかいったージ状は、*
QLE UA 9.		P ±	画器-名样	*
		厂 後で設定	#HD-7	SUURYOU *
		□ 次国からの初期値として保存		<b>○</b> 王
	11 ST 4 42	1 12/10 A 88 - 1 12 1		1 単で担当
				Provide the second state of the

[編集]メニュー [可変文字列] または [数値データ]を選択します。 表示開始位置 (文字列の左上位置を指定)をマウス左ボタンで指定します。 詳細設定ダイアログボックスに次の項目を設定します。

文字数 または 桁数

表示色 (前景色、背景色、背景色有効)

各項目の表示データを画面上に配置します。 表示位置は簡単に修正できますので、ここでは 大まかな位置決めを行います。 表示位置は次のような手順で修正することができます。

マウスを修正しようとするデータの境界部分に移動します。 マウスカーソルの形状が [矢印] から [十字] に変化します。 マウスカーソルの形状が [十字] の時、マウス左ボタンをクリックします。 修正しようとする項目が矩形で囲まれ、詳細設定ダイアログメッセージが表示されます。

選択された描画要素の境界付近にマウス カーソルを移動すると、マウスカーソル形状 が十字から右図のように変化します。

マウスカーソル形状が右図のように変化した ときマウス左ボタンを押したままでマウスを 移動するとカーソルの方向に従って図形が 拡大・縮小されます。 所定の大きさに達したときマウス左ボタンを 離すと図形の大きさが確定します。



のカーソルが表示されているときは、大きさ は変化せず表示位置のみを移動させることができます。

図形の表示位置と大きさは グリッド線の間隔の 1/2 で揃えられます。

固定文字列などでは、 🛟 のカーソルが表示され、表示位置の変更のみが可能です。 表示位置はグリッド線の間隔で揃えられます。

この状態で、移動、図形の大きさの変更、詳細設定の変更 などを行い、マウス右ボタンメニューを開いて登録します。

修正が不要な場合は、マウス右ボタンメニューで [再表示] または [取消] を選択します。 データが表示されていない位置でマウス左ボタンをクリックすることで取消すこともできます。

CRT画面上に配置する図形や文字列は方眼に合わせて表示位置が調整されます。 微妙な位置 合わせを行う場合には方眼の間隔を変更してください。 (表示メニュー 方眼設定)

#### <u>2-3-4 項目名称の配置</u>

矩形枠と固定文字列を使用して項目名称をCRT画面上に配置します。

-14 ##	85 AA	17	1.1			_	
上げ帳						2003年11月1	7日(月)118
日付		*	名		数量	単価 合	計
#######	2 2424	unususa	******	a#a#a	******	******	******
12				×	固定文字列		
ġ.	35 / 9	0 1 10	<u>P</u>		位置	60 / 88	T TOP
¢.	156 / 12	0			文字列	日付	
10.00.000	21	0	6		フォント	Lie Heathout	u Vaatatiil
世界英·約45	24	0	- <u>p</u>			MS 1999 24	0
IR / +- #24篇	24	0	- 12		制放出	23	0
境界色-科幅	24	0	- 6		背景色	0	0
書題	0		1	-		□ 背景色有効	□ 点满
18.55	(35.)		-		消去色	0	
8-645	F				処理種別	(131.)	
7-12	-		• 0	*	面面名称	10007	
	〒 後で終生						
	□ 次回からの	の初期値として世	<b>¥存</b>		M(9)11-1-		
EE ADA	10:14	<b>#表示  推攻</b>	1			P 🚽	
			÷.			□ 後で設定	
く <b>まれのまれ</b> 雨	⋽୲∔ ╆≞	(十些)	こわいん			□ 文字列一覧/	全经
シイデレンイデヤჼ ミ ラ てい ±	囲み 仁炳 オ	/ 11 师 /	て16以71			□ 次回からの初	期値として保護
こんていま	У.						
					登録 前明	除 取消 再表	示 複写

### 2-3-5 データ部の詳細設定

可変文字列、数値データは表示する内容を決めるために識別コードを設定します。 また、マウス左ボタンでクリックされたとき、ロジック部にメッセージを送ることにします。 ロジック部ではクリックされた項目を判定し入力用ウィンドウを開きます。

可是文字列	the second s	N			
0.00	40 / 100 E TOP				
文手教	1 2		売上げ帳の表示、入力の	処理を行うプログラム	
24:04	M8 2540 24×12 N-41		ヘメsッセージを送ります。		
的原色	13 0		<b>一</b> /		
發展色	10 0				
	F BREAK FAR				
消去色	0				
	[二人力可				
	□ 日丰語入力				
7-94		1	識別コード	ロジック部で変数名として使用しま	きす
処理理問	ロウィンドウヘメッセージ送信 王			英数字、""(アンダースコア)で設	定
画图名称	·			してください。	
講習コード	HZUKE			牛面け苗字 最大20文字で設定	1
	P 3				0
	秋で秋定				
	□ 次国からの和期値として保存				
211 MIN	10.6 再表示 疾军				

上げ帳				2003	年11月17日(月)1
日付	品	名	数量	単価	合計
#######	*********	******	. ***. ***	. ***. ***	**. ***. ***
1					
文字列		×	₩ <b>#</b>		
x	48 / 128	TOP	uff 36	/ 120 1	OP
字款	7	25	A 156	/ 160	
9h	MS ゴシック 24×12	* *	18月 20	0	_
景色	23 0	7	· 堆存色· 持幅 24	0	
景色	20 0	*	現得色-持幅 24	0	
	▼ 背景色有効	一点液 专	境界色·科幅 24	0	0 2
去色	0	10	(1) (1) (1) (1) (1) (1) (1) (1) (1) (1)		
	□ 入力可	53	·····································	ンドウヘメッセージ送	
	□ 日本語入力			KE	
-976		- □ -	-	202	
理種別	自ウィンドウヘメッセー	ジ送信 ▼	日次	目からの初期値として	保存
面名称		*	R28   408   1034	再表示   相	12
別コード	HIZUKE	*	and Research Process		
l	0 😤	J			
	□ 後で設定				
	□ 次回からの初期値	RUT 保存			
趣 削除	取消 再表示	複写	政備データ		
			位置	648 /	128 TOP
			表示析数	8	303
文字列と知	三形枠の識別コー	۲	フォント	MS IS:	ウ 24×12 ボールビ
同一の文字	マ列を設定します。	9	前景色	23	0
			背景色	20	0
コードは次	のように設定しま	す。	124.5	▼ 背景色	有効 「点滅
項目名	識別コー	۴	消去色	0	
日付	HIZUKE			□ 入力可	
品名				- UD95	ルスしない
				▼ 桁区切	り使用
合計	GOUKFI	<u>~</u>		16進調	
			処理種別	18040/Kg	コヘメッセージ送信
			国の名称		
		/ /	1-1-02.00	TANKA	
Г	あはご りょうだつ			-	-

再表示 複写

登録 削除 取消

#### レイアウト作成

2-3-6 表の完成

データ部を下方向に複写して表を完成します。 複写には2種類の方法があります。

個別複写

データの1項目のみを選択し マウス右ボタンメニューで [複写]を選択します。 詳細設定ダイアログボックス下端 の [複写] ボタンでも同様に個別 複写を行うことができます。

日付	ELEETI	
225522	10 <b>2</b>	48 / 128 (* 10P
CHILLER	大平数	P = =
-	3458	NES 25/97 24×12 85-81 *
	前景色	23
<b>HH</b>	背景色	20 0
87		F HERMS F GM
#¥	消去数	0
		广大力町
		E 日本語入力
	#~2%	
	治理療的	目からせかみませージます。
	画面 6件	and an and a second sec
	AND-K	HOME *
		p +
		C WORK
		「次国からの初期儀して専作
	transel res	I multiment and
	Dit NO	E EIA AAT EF

#### 一括複写



描画要素を選択していない状態で、複写の対象となる項目を囲む矩形を選択します。 固定文字列 日付 のように一部のみしか矩形領域に含まれていない描画要素は複写の対象とは なりません。 複写以外に、移動、削除を行うことができます。

ここでは、一括複写を使用して表を完成させます。 ポップアップメニューで複写を選択すると、詳細設定ウィンドウが表示されます。

- 複写個数 11行複写し、合計12行表示できる ようにします。
- Y方向ドット数 複写の間隔を40ドットとします。 - xx と設定すると、上方向へ 複写します。
- 識別番号 複写するとき、識別コードの番号部
   を + 1 して行きます。
   1行目のデータは HIZUKE+0
   12行目のデータは HIZUKE+11 と

机三氟基苯基	
複写個数	11 🚊
×方向ドット数	0 🗄
Y方向ドット数	40 💼
識別番号	1 .
複写 取注	ā l

HIZUKE+11 として識別できるようになります。

複写後

日村	品名	数量	単価	合計	
avavat	xavavavavavavavavavavava	. ***. ***	. ***. ***	**. ***. ***	
avavat	xavavavspsoravavavspsora		. ***. ***	**. ***. ***	
asasas	savavavavavavavavavavava			**. ***. ***	
asasas	xavavavavavavavavavavava			**. ***. ***	
ararar	xavavavsososavavsosososa		. ***. ***	**. ***. ***	
ararar	uavavavsosovavavsosova			**. ***. ***	
#######	savavavavavavavavavavava			**, ***, ***	
asasas	xavavavsosavavavsosavs			**. ***. ***	
avavat	xaxaxaxxxxxxxxxxxxxxxxxx			**. ***. ***	
asasas	uavavavararavavavarara			**. ***. ***	
avavav	savavavavavavavavavavava			**. ***. ***	
asasas	xavavavsosovavavsosovava				
	20		671 N.	1	

複写により表がほぼ完成しました。 最下行の罫線の太さなどを調整すれば 完成です。

#### 罫線の太さ調整、総合計の欄を追加して完成

				**, ***, ***
*****	*********************	, ***, ***	. ***. ***	**, ***, ***
*######	**********************	. ***. ***	. ***. ***	**, ***, ***
		1	1. A	

#### レイアウト作成

#### 2-3-7 スクロールバーの配置

SimpleMaster ではスクロールバーを部品として CRT 画面上に配置することができます。

スクロールバーでは、データの一部を CRT 画面上 に表示するときに、表示位置を決める手段として 使用します。

SimpleMaster ではスクロールバーを配置すると スクロールバーを操作されたとき、ロジック側へ メッセージを送ります、ロジック側では表示開始 位置を取得し再表示を行います。

スクロールバー × t 商点 947 / 320 T TOP 947 / 320 ± 10.00 \*\*\* 4 「未早方向 @ 重重方向 \*\*\* 180 F. 559% F 974 \*\*\* 在上· 右下特 10 11. 内部 12 14 \*\*\* 涌去色 21 \*\*\* スライダー 1942 5 \*\*\* 表示/デー与量 30 12 + 1 3 12 3 10 3 移動量 \*\*\* ▼ 使用时 表示量 マウスホイール \*\*\* カージルモー ▼ 使用可 \*\*\* 然理種利 自ウロドウヘメッセージ通信 💌 画图名样 データ量 \*\*\* ٠ 第10-11 SCROLL \* \*\*\* 0 愤 \*\*\* Ŧ 「操作設定 もできます。 ○ 次回約らの初期値として保存 \*\*\* 今回はプログラムで設定します。 with min 再表示 1981

CRT 画面表示部 データ量 ダブルボタン シングルボタン 表を作成したときの値(12行)を 設定します。 データ全体の大きさが固定の場合 には数値を設定できますが、実行 時にプログラム側より設定すること

表全体

表示開始位置

表示量

移動量 左より シングルボタン / ダブル ボタン / ボタン スライダー間を マウス左ボタンでクリックした時の 表示開始位置の移動量を設定 します。

消去色 表示量 > データ量の時 スクロールバーは非表示となり ます。この時に使用する消去色 (作業領域の背景色)を設定 します。 作業領域の背景色はレイアウトの 初期設定で設定しています。

<u>2-3-8 ボタンの配置</u>

L.

次のボタンを配置します。

 [伝票入力]
 入力ウィンドウを開きます。

 [ディスク登録]
 データをハードディスクに保存します。

 [前画面]
 メニュー画面に戻ります。

伝票入力	〕 ディスク保存		前回
97	×	#97	×
18.A.	36 / 720	她点 192 / 720 厂 112	99) -
時点	168 / 760	終点 360 / 760	
#\$2x0文字	伝蒙入力 =	ポタンの文学 ディスケ保存	
オンド	MS 3990 20×10 #-41 -	7+21 MS 35+5 20×10 h-4	F
150/07科幅	P =	#50/074編 2 王	
生上 右下校	10 11 00	左上右下校 10 11	3
約5.40表面	12	市5040表面 12	
的(水)文字	13 14	ボジンの文字 13 <b>14 14</b>	
再去色	0	消去色 0	
假ン文学列	@ 中央 C 指定 [7] 王	ポジン文学列 @ 中央 C 指定  0	*
サングションキー		77595404- [	
心理推用	自ウィンドウヘメッセージ送信 💌	想理権制 自ウィンドウヘメッセージ送信	
和副名称	×	通過名件 	*
取用白一杯	MUURYOKU *	總府口一世 HOZON	*
	E q	E 이	
	□ 禄で說定	「 後で設定	
	□ 文字列一覧へ登録	□ 文字列一覧へ登録	
	「 次回からの初期値として保存	「次国からの初期論として	保存
TER MIRS	取消 再表示 複写	金蜂 网络 取消 再表示 夜	5
97	×		
<b>タン</b> 地点	912 / 720 「 <u>FOR</u>		
的点 转点	× 912 / 720 □ 1009 1006 / 750		
タン 他点 他点 #S2AD文字	× 912 / 720 □ <u>1709</u> 1008 / 750 ■ ■		
参加 総点 統点 析知2の文字 7月ント	× 912 / 720 「10月 1008 / 760 「約第第章 ● 「MS ゴシック 20×10 前1-41 ▼		
地点 地点 地点 地反200文字 24ント ポジンの枠幅	× 912 / 720 「10月 1008 / 750 「新売酒」 ▲ MS ゴシック 20×10 市-AF ▼ 早 一里		
総点 成点 152-00文字 242/ト 152-00科幅 生上・右下枠	× 912 / 720 「10月 1008 / 760 「約第第第 「MS ゴシック 20×10 前1-AH」 「2 王 10 11 11	ここをクリックすると、現在までに登録され	ιτ
総点 総点 ポジンの文字 パジント ポジンの科幅 生上・右下枠 ポジンの表面	× 912 / 720 「10月 1008 / 760 「新商品」 「MS ゴンック 20×10 計'-AH」 「2 」 11 」	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 「メニュー画面」を選択します。	ιτ
総点 総点 地力 地力 地力 地力 地力 地力 地力 地力 地力 地力	× 912 / 720 10月 1006 / 760 MS語题 ・ MS ゴシック 20×10 ポールト 1 2 1 10 11 1 12 1 13 1 14 1	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ
敏点 地点 地点 地点 地点 か文字 オント おひの存編 生上・右下校 おひの文字 肩去色	× 912 / 720 「10月 1006 / 760 「新語語」 「MS ゴシック 20×10 前-AF」 2 日 10 11 1 12 1 13 1 14 1	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ
<ul> <li>第点</li> <li>第点</li> <li>第点</li> <li>第二</li> <li< td=""><td>× 912 / 720 「100円 1006 / 760 「新売売」 ▲ MS ゴシック 20×10 h'-AF ● 2 雪 10 11 1 12 1 13 1 14 1 0 9 6 中央 C 指定 0 雪</td><td>ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。</td><td>ιτ</td></li<></ul>	× 912 / 720 「100円 1006 / 760 「新売売」 ▲ MS ゴシック 20×10 h'-AF ● 2 雪 10 11 1 12 1 13 1 14 1 0 9 6 中央 C 指定 0 雪	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ
総点 総点 構成しめ文字 行りいの文字 行りいの科幅 転しの文字 再去色 行りいの文字 再去 たの文字列 トッグッシュンキー	× 912 / 720 「10月 1008 / 760 「熱毒節」 「MS ゴシック 20×10 市 <sup>1</sup> −AF」 早二日 10 11 12 13 14 13 14 0 ・中央 C 指定 0 王 「一一」	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	וד
<ul> <li>第点</li> <li>第点</li> <li>第点</li> <li>第二</li> <li< td=""><td>× 912 / 720 「10月 1008 / 760 「新商版 ● 「MS ゴンック 20×10 市 - AF ● 2 ① 10 11 ■ 12 □ 13 ■ 14 ■ 0 ● 中央 C 指定 p ⊡ ● 中央 C 指定 p ⊡ ● 二 ●</td><td>ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。</td><td>ιτ</td></li<></ul>	× 912 / 720 「10月 1008 / 760 「新商版 ● 「MS ゴンック 20×10 市 - AF ● 2 ① 10 11 ■ 12 □ 13 ■ 14 ■ 0 ● 中央 C 指定 p ⊡ ● 中央 C 指定 p ⊡ ● 二 ●	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ
地点 約点 約久の文字 カント 約2の科幅 至上・右下枠 約2の次等 消失の文字 消失の文字 消失の文字列 かかっパー 地理種用 画面名称	912 / 720     「市の円       1006 / 760       約馬茄       MS ゴシック 20×10 市 ートト・       2       10       11       12       13       14       0       (* 中央) C 指定) 日       原上位うく) ドウをMK       第       現上位うく) ドウをMK	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ
地点 終点 状況への文字 パジント 状況への科幅 生上・右下校 状況への文字 消費への文字 消費シスマ字利 リッグションキー 思想都名称 動用ロード	× 912 / 720 「10月 1008 / 760 「新売版 ● 「MS ゴシック 20×10 前 - AF ▼ 早 一日 10 11 ■ 12 ■ 13 ■ 14 ■ 0 ● 中央 C 指定 D 団 「一一 ▼ 優上位ウィンドウをMK ▼ 以口2 - 通版 ■	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ
地点 地点 地点 地点 地点 地点 地点 地点 かり、の文字 対シ、の文字 対シ、の大字 対シ、の本 特シ、の大字 消費、の文字 消費、の文字 消費、の文字 消費、の文字 消費、の文字 消費、の文字 引 いいの文字 消費、の文字 引 いいの文字 消費、の文字 引 いいの文字 消費、の文字 引 いいの文字 消費、の文字 引 いいの文字 消費、の文字 引 いいの文字 消費、の文字 引 いいの文字 消費、の文字 引 いいの文字 消費、の文字 引 いいの文字 引 いいの文字 引 いいの文字 引 いいの文字 引 いいの文字 引 いいの文字 引 いいの文字 引 いいの文字 引 し、のの文字 引 し、の文字 引 し、のの文字 引 し、の 、の文字 引 し、の文字 一 、 引 し、の 、 の 、 の 、 の 、 の 、 の し 、 の の の 、 の 、 の の の 、 の の の 、 の の の の の の の の の の の の の	912 / 720     「市の円       1008 / 760       柳雨面面       MS ゴンック 20×10 市 ー・トレッ       2       10       11       12       13       14       0       ※ 中央 C 指定 p 当       風上位からどやをMK       第       第二位からどやをMK       第       第	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ
地点 終点 特別ンの文字 カシト 特別ンの科幅 転し、右下称 特別ンの文字 消費ンの文字 消費ンスの文字 消費ンスの文字 消費ンスの文字 消費ンスの文字 消費ンスの文字 消費の、一手 配面名称 取用ロード	912 / 720       「市の円         1006 / 760       ●         MS ゴシック 20×10 市 ートトッ       ●         2       ①         10       11         12       □         13       14         0       ①         0       ①         ※       中央 C 指定 D 型         ※       ●         ※       ●         ※       ●         ※       ●         ※       ●         ※       ●         ※       ●         ※       ●         ※       ●         ※       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ● <t< td=""><td>ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。</td><td>זד</td></t<>	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	זד
地点 終点 特別ンの文字 フォント 特別ンの文字 特別ンの文字 消費ンの文字 消費ングションキー 思想都名称 動用ロード	912 / 720       「市の円         1006 / 760       ●         MS ゴシック 20×10 市 - AF (1)       ●         2       一         10       11         12       11         13       14         0       ①         ※       中央 C 指定 p 団         ※       ●         ※       ●         13       14         0       □         14       □         15       14         16       □         ※       ●         ※       ●         ※       ●         ※       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ●         ●       ● <t< td=""><td>ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。</td><td>ιτ</td></t<>	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ
地点 地点 地点 地点 地点 地点 地点 地点 地点 地点	912 / 720       「市の円         1006 / 760       第         第二方ック20×10市ーホトッ       第         10       11         12       11         13       14         14       11         15       14         16       11         17       14         18       14         19       14         10       14         11       14         12       14         13       14         14       14         15       14         16       14         17       14         18       14         19       14         10       14         11       14         12       14         13       14         14       14         15       14         16       15         172       16         18       17         19       18         11       14         11       14         11       14         11       14         15 </td <td>ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。</td> <td>ιτ</td>	ここをクリックすると、現在までに登録され いるレイアウトの一覧が表示されます。 [メニュー画面]を選択します。	ιτ

#### <u>2-4 伝票入力画面の作成</u>

[売上げ帳] 画面での 入力画面 を作成します。

#### <u>2-4-1 伝票入力画面の作成</u>

メニュー画面の作成時と同様に [ファイル]メニュー [新規作成] [子ウィンドウ] を選択し、 タイトルに [伝票入力(新規)] と設定します。

セイアウト作品 [SimpleDemo / 伝篇。	レイアウト情報	0 RE (チウィントウ) 🛛 🕺
伝言入力 (新想)	運動の新聞度	· · · · · · · · · · · · · · · · · · ·
	直面の表示位置	☑ 中央 □ 移動不可
		는 글×이 크
	与小小蝉城	高さ 30 三 開城色 5
	タイトル	伝票入力(新規)
	フォント	MS ゴシック 20×10 ホ'ールト"
	表示位置	◎ 空端 ○ 中央 表示色 4
	作業領域 背景色	6 0'97'a1-5a5
	ビットマップ名	気を
	ポジン積差	高さ 0 当 背景色 0
	子ウインドウ	科幅 2 三 特色 7
	マウスカーソル形状	● 矢印 C 十字 C 砂時計
	マウス入力	□ 親54シドウで入力可
	キー人力時	前展色 日 日本 背景色 ヨ
	設定 取消	「約7時面面 「自動更新不要

画面の大きさはレイアウト作成途中で調整しますのでここでは適当な値を設定します。

固定文字列、可変文字列、数値データ、 矩形(キーボード入力域)、ボタン を使用 してレイアウトを作成します。

左右、上下の余白を調整し画面の大きさを 確定します。

レイアウト情報の設定[子ウィントウ]					
画面の解像度	512 =× 440 =				
画面の表示位置	☑ 中央 □ 総動不可				

日付	•• / ••
品名	
単価	. ***. ***
数量	. ***. ***
合計	**.***.

#### レイアウト作成

#### <u>2-4-2 キーボード入力の設定</u>

可変文字列、数値データの詳細設定 ダイアログボックスで[入力可]をセット するだけでその項目のキーボード入力が 可能となります。

キーボード入力は、ロジック側で記述 することにより、継続させることができます。 (例)月の入力が終わると続けて 日の入力を行う。

カーソルキー をセットすれば、右図の場合 [ ]または [ ] 方向のカーソルキー のみを受付け、プログラム側の記述により 入力を継続させることができます。

キーボード入力可能な数値データを囲む 矩形領域に、数値データと同一の識別 コードを割当てることができます。 この場合、矩形領域の内部をマウス左 ボタンでクリックすると同一の識別コード が設定された数値データの入力処理が 開始されます。 (可変文字列も同様です。)



レイアウト作用(LingleDemy / 世界入力(新用))

/ 112

1.1 5 - 2

\*\*\*\*

2408

有颜色

建果色

April.

7-5/16

的理想和

**法国 され** 

請約12-17

144.7 10

P. 田 P MS ゴシャラ 24×4

25.0

0

学人力年

1台道府

HHELKE

10702

BIS ANY RIA TRAT HW

市場の方の日期日にして 保存

25

P 接受色发动 F 他派

ゼロを力。33.年い 和区(的)使用

TELCH

T YOR

10

伝素入力 (新潟)

日付

晶名

単価

数量

合計

世録

.

\*\*\*\*\*\*

\*\*\*

\*\*\*.

\*\* \*\*

INCH .

#### 各項目には次の識別コードを設定します。

項目名称	識別コード	備考
日付 (月)	IN_HIZUKE1	入力可
日付(日)	IN_HIZUKE2	
品名	IN_HINMEI	
単価	IN_TANKA	
数量	IN_SUURYOU	
合計	IN_GOUKEI	入力不可

(合計値はプログラムで計算します。)

#### 識別コードの設定

レイアウト作成時に設定された識別コードはプロジェクト単位で保存されています。 簡単なマウス操作で過去に設定された識別コードより選択することができます。



#### ボタンの詳細設定



\$97	<u>×</u>	#97	×
物点	32 / 380 F (TOP)	推击	144 / 380 F TOP
鮮点	112 / 420	新点	224 / 420
ポジシの文字	( <u>0.10</u> +)	ポシの文字	(E)A +
7424	MS 35-99 20×10 #-48 *	2424	MS 35-99 20×10 #-48
10000种植	p ==	180-0种幅	p ==
在上,在下转	10 11	在上:在下科	10 11
ポないの表面	12	ポタンの表面	12
ポシの文字	13	#900文字	13
消去色	0	消去性	0
ポない文字列	· 中共 C 指定 [5]	ポタン文字列	● 中共 C 接定 反 一只
7129/204-		71395004-	
后细植树	認うにおうへくったージます。*	后细胞的	BO-0809555 *
西亚名称		西亚名称	
満ちつ-ヤ	FTOURCKU.	調整ロード	
	D		0
	C Brill P		F 9789
	T THU-WARM		T THN-WARM
	T attacatter		T attacatters and
	A NUMBER OF THE PARTY		1 WORKSON TORNAL DC 1919
211 AFR	取消 再表示 推荐	210 APA	取消 再表示 推荐

(株) フジテクニカ

#### <u>2-4-3 伝票入力(修正)画面の作成</u>

伝票入力(修正)画面 は 伝票入力(新規)画面 を複写して作成します。



伝票入力(新規) で次の項目を変更します。

画面のタイトルを 伝票入力(修正) に変更します。 (レイアウト情報設定) [取消]ボタンの位置を右に移動し、 [削除]ボタンを追加します。

[ファイル]メニュー [閉じる] または [上書き保存]

でディスクに保存します。

ファイル名はタイトルを修正したので、[伝票入力(修正)] となり、[伝票入力(新規)]はそのまま保存されています。

日付	** / **
品名	
単価	. ***, ***
数量	, ***, ***
合計	**. ***. ***



#### 2-4-4 エラーメッセージ表示

SimplePack では数値データ入力時に上下限値の検査を行い、上下限の範囲外の数値が入力された場合、エラーメッセージを表示することができます。

#### レイアウト側の準備

売上げ帳画面にエラーメッセージを表示する領域を設定します。



エラーメッセージ表示領域として、矩形と可変文字列を定義します。

#### 表示色 / 背景色

通常時は背景色で表示されるように設定します。

エラー発生時には、エラーメッセージを白、背景を赤で表示します。

識別コード

SYSTEM+11 を設定します。 この識別コードはエラーメッセージ表示用に予約されています。

#### 可変文字列の文字数

文字列設定ファイル(次ページ参照)で設定した文字列が表示できる文字数を指定します。

#### Microsoft Excel ファイルの設定

数値データ入力時の入力可能範囲、およびエラーメッセージとして CRT 画面上に表示する 文字列は Microsoft Excel 形式のファイルで設定します。

雛形となる Excel 形式のファイルはプロジェクト作成時に次のフォルダーへ複写されています。 (プロジェクト情報格納先フォルダー) + ¥Excel

No	識別コード			下阳庙	ト阳佑	エラー番号	
NO	名称	番号(1)	番号(2)	비지에게	山水市	下限	上限
1	IN HI7HKE	Λ	Λ	1	12	1	2
2	IN HIZUKE	1	1	1	31	1	2
3	IN SUURYOU	0	0	1	9999	1	2
4	IN TANKA	0	0	1	99999	1	2
5							
6							
7							

識別コードに対応して入力範囲、入力エラー発生時のエラーメッセージ番号を設定します。 メッセージ番号は MesgData.xls 左端の番号に相当します。

#### MesgData.xls 文字列設定ファイル

No	識別コード		世通立字列 (言語 1 )	+ 潘立字列 (主語り)	+ 通立 字列 ( 言語 2 )
NO	名称	番号	共通又子列(言語「 <i>)</i>	共通文子列(言語 2 )	共通又子列(言語 3 <i>)</i>
1			下限エラー		
2			上限エラー		
3					
4	SYSTEM	15	SimpleMaster V1.01		
5	SYSTEM	16	デモ用プログラム		
6	SYSTEM	17			
7	SYSTEM	18	㈱ フジテクニカ		
8					
9					

多国籍言語に対応して文字列を設定することができます。 番号 4~7 はバージョン情報画面で参照しています。

#### <u>2 - 5 帳票レイアウトの作成</u>

売上げ帳を印刷するためのレイアウトを作成します。

#### 2-5-1 レイアウトの新規作成

用紙の大きさ

帳票用レイアウトはCRT画面用レイアウトとほぼ同じ手順で作成することができます。 CRT画面用レイアウトと異なる点は

> 画面の大きさとは異なり、任意の大きさを指定することができます。 縦横比を実際の用紙に合わせておくことで良好な印字結果が得られ ます。

表示色 モノクロプリンターを使用する場合には、白·黒·灰色 のみを使用してくだ さい。 カラープリンターを使用する場合には、レイアウトで設定された色で 印字されます。

#### レイアウトの新規作成

[ファイル]メニュー [レイアウトの新規作成] [その他] を選択します。



#### レイアウト情報の設定

[新規作成] を選択するとレイアウト 情報の詳細設定画面が開きます。

レイアウトの大きさに仮の値 800×600 を設定します。 (適当な数値で可)

レイアウト情報の	夏屋 [その曲]	×
領域の大きさ	800 곳×600 곳	
レイアウト名	極葉売上げ帳	
作業領域 背景色	29	
設定 取消		

レイアウトのファイル名を設定し [登録] します。

(注) レイアウトの大きさをわかりやすくするため背景色を設定しています。

2-5-2 レイアウトの大きさの決定

横方向の大きさを決めるため、項目名称、1行分の印字データを作業領域に配置します。



左右の余白を 3桁(60ドット) とすると 横方向の大きさは 39.5桁(790ドット) となります。

つぎに、縦方向の大きさを決めます。

実際の用紙の大きさA4 の場合297 x 210 mm縦横比210 / 297 = 0.7070縦方向の大きさ790 x 0.7070 = 558.6

[編集]メニュー	[レイアウト	詳細設定	[] により、
レイアウト情報の詳細	設定画面	iを開きま <sup>.</sup>	す。
横方向の大きさ	を 800	790 に	変更します。
縦方向の大きさ	を 600	560 に	変更します。

LAPONEL	RE (tom)	×
領域の大きさ	790 코×560 코	
LAPONE	爆撃、先上げ機	
作業領站 背景包	29	
1877 17:4		
4014	1	

[注] レイアウトの大きさを実際の用紙の縦横比に合わせることで 拡大印字した時良好な印字結果が得られます。

#### <u>2-5-3 左右余白の調整</u>

2 レイアウト作成 [SimpleDemo / 編纂.売上げ編]
7714 編集 表示 ここでマウス左ボタンを押す。 売上げ帳 名 散 量 単 価 合 計 旧付 ÷ 品 anakana | kanakanbukanakanankana 2838383838383 REPRESENT. REFERENCES: 取消 崩隊 多心 板写 ここでマウス左ボタンを離すとポップ アップメニューが表示されます。

作業領域の大きさを設定し、描画要素の一括移動機能を使用して左右の余白を調整します。

詳細設定ダイアログボックスで 移動量 40ドット を設定し 一括移動を行います。

机间接多数		×
×方向ドット数	40	*
Y方向ドット数	0	
移動取得	i l	

移動後の帳票レイアウト



#### 2-5-4 識別コードの設定と複写

日付などの描画要素に識別コードを設定します。

日付	P_HIZUKE
品名	P_HINMEI
数量	P_SUURYOU
単価	P_TANKA
合計	P_GOUKE I

#### 一括複写機能を使用して描画要素を複写します。



一括移動処理時と同様に、複写の対象となる描画要素を矩形で囲み、複写を選択します。 複写の対象となるのは、図形全体が矩形の内部となる描画要素のみで、上図のように、日付 などの固定文字列や罫線は一部が矩形の外部となるため複写の対象とはなりません、

詳細設定ダイアログボックスで、複写個数、間隔を設定します。

識別番号に 1 を設定すると複写時に識別コードの番号部が +1 加算されます。 日付データの場合、

1行目の識別コードは P\_HIZUKE + 0

最下行の識別コードは P\_HIZUKE + 11 と設定されます。



(株) フジテクニカ

最後に、罫線を調整しページ番号などを配置して帳票レイアウトを完成させます。

#### 完成した帳票レイアウト

PAGE ** / **					
日付	8	名	数 量	単価	合 計
444444	********	*********	******	******	*******
444444	*****	*********	******	*******	*****
444444	*******	********	*******	******	******
444444	******	*********	*******	******	******
444444	******	*********	*******	*******	******
444444	******	*********	******	*******	******
444444	******	********	******	*******	******
******	*******	*********	******	*******	******
******	********	*********	*******	*******	*******
444444	********	********	*******	*******	*******
4444444	*****	*********	*******	******	******
*****	*******	*********	******	******	******

#### <u>2-6 制御用画面の作成</u>

制御システム用のレイアウト例として、簡易水道システムの監視画面を作成します。 (ただし、実際の画面からはかなり簡略化しています。)

#### <u>2-6-1 設備図画面の新規作成</u>

メニュー画面の作成時と同様に [ファイル]メニュー [新規作成] [親ウィンドウ]を選択 します。



設備図画面は他の画面とは異なり、背景色全体を暗くします。 また、この時方眼が見えなくなりますので方眼の表示色も白に変更します。 (表示メニュー 方眼設定)





設備図には、直線、 マーク、 棒グラフ、 固定文字列、 数値データ を使用しています。

#### <u>2-6-3 マーク</u>

マーク は 2値のビットマップ で ON / OFF のピクセルに表示色を設定することができます、

ON のビットに 前景色、 OFF のビットに 背景色 が割り付けられます。 設備図では[背景色有効] が設定されていない ので、OFF のビットは作業領域の色となります。

SimpleMaster で CRT 画面上に配置することが できる図形や文字列は2系統の色を設定すること ができます。(プロジェクトの設定により4系統の 設定も可能です。)

設備図では ポンプやバルブの運転状態(入/切) に合わせて表示色を切り替えます。



マークの編集



マークは[編集]メニュー [マークの設定] でも編集することができます。
<u>2-6-4 棒グラフ</u>



数値データを上下限値でスケーリングし 棒グラフとして CRT 画面上に表示します。

設備図では数値データが異常の時 棒グラフの表示色を切り替えるため 2系統の表示色を設定します。

#グラフ	×
始点	64 / 460 TOP
終点	160 / 720
表示方向	●縦 ○ 棟
表示色	31 35
背景色	1
外形色	30
消去色	0
級中區/ 社中中區	
下限值/上限值	0 - 100
処理種別	(なし)
画面名称	8
織別ロード	AI
	3
	□ 後で設定
	□ 次回からの初期値として保存
登録 削除	取消 再表示 複写

# 2-6-5 識別コード設定

下図のように識別コードを設定します。



# 3 ロジック作成

SimpleMaster で作成したレイアウト情報に対してロジック(プログラム)を作成します。 ここで作成するプログラムは、シングルプロセスで構成される小規模なアプリケーションです。

- ロジックを作成するためには次のコンパイラが必要です。 Microsoft Visual C++ V6.0 または Microsoft Visual C++ .Net
  - {注意} SimpleMaster にはコンパイラは含まれていません。 別途準備してください。
    - 本書ではコンパイラの使用方法などの説明は含まれていません。 コンパイラに付属する マニュアルを参照してください。

## <u>3-1 プログラムの構造</u>



小規模なアプリケーションでは、シングルプロセスで構成し、共有データ領域はローカルメモリ上に 確保されます。データ処理部はスレッドで作成し、共有データ領域を介して CRT 画面処理、帳票印字 処理部と情報の受渡しを行います。

中・大規模アプリケーションでは共有データ領域を 共有メモリ 上に確保し、複数のプロセスから参照・ 更新されます。 また、名前付きパイプを使用して複数のパソコンで共有メモリ上に確保されたデータ 領域をアクセスする手法も準備しています。

#### 3-2 共有データ領域

共有データ領域は WORD 型データ、数値データ、文字列データ が格納される配列で構成されます。



共有データ領域には一連の番号が割当てられ、ロジック部はこの番号で参照・更新を行います。

#### 共有データ領域の確保

共有データ領域はレイアウト作成時にデータ数などが決まります。



レイアウト作成時、共有データ領域は 識別コード名称と番号で管理されます。 レイアウト作成プログラム SimpleMaster は [ファイル]メニュー [ロジック作成] を選択した とき、次の手順で共有メモリの一覧表を作成します。

プロジェクトに属するすべてのレイアウト情報を読込み、識別コード名称と同一名称で参照する番号の最大値の一覧表を作成します。 (作成例)

識別コード 名称	データ番号 最大値	データ番号 先頭値
SYSTEM	100	0
AAA	10	100
BBBB	20	110
0000	5	130
DDDD	10	135
	(合計) 145	

このような表を作成し、共有データ領域のデータ量、各識別名称の先頭のデータ番号を決めます。 識別コード名称 SYSTEM(データ量 100) は SimplePack で管理する領域で、レイアウト作成側で 確保する必要はありません。

# ヘッダーファイル SimpleCode.h

レイアウト作成プログラム SimpleMaster で [ファイル]メニュー [ロジック作成] を選択すると 識別コードの一覧表を作成し、ヘッダーファイル SimpleCode.h をロジック領域に作成します。

#define	SYSTEM	0
#define	AAAA	100
#define	BBBB	110
#define	0000	130
#define	DDDD	135

ロジック部ではこのヘッダーファイルを参照してレイアウト部で定義された共有データ領域の識別 コードに該当するデータヘアクセスします。

ロジック部で右図のデータ領域 をアクセスするためには次のように 記述します。

#include "SimpleCode.h"
float data = Get_fData (SUURYOU+10),
<b>Set_fData</b> (SUURYOU+10, data);

数量	LILES	CONTRACTOR OF STREET
	位置	516 / 528 TTOP
. ***, ***	表示有数	
. ***, ***	74525	M8 359524×12 #-41
*** ***	机景色	13 0
*	泉景色	20 0
. ***. ***		戸 背景色有効 「「 糸満
. ***. ***	消去色	0
*** ***		广人力可
	-	「 ゼロサプレスしない
. ***, ***	_	戸 和区470使用
. ***. ***		<b>厂</b> 16通数
*** ***	カーソル	
	均增值则	目ウィンドウヘメッセージ送信 ・
. ***. ***	演曲名称	-
*** ***	國来白-ド	BUURYOU .
. ***. ***		Po ±
		□ 水園からの精粉値として保存
	THE R L L L	free frees free free

識別コードはこのようにコンパイラに 渡される情報となりますのでコンパイラ の「名前付け規則」に従わなければ なりません。

レイアウト作成プログラムでは識別コード

に次の制限を加えています。

使用できる文字 数字(0~9)、英字(大文字、小文字 共に可)、 特殊記号("\_"アンダースコアのみ) 先頭文字 英字のみ 文字数 最大 20文字

レイアウト作成プログラム SimpleMaster で識別コードの追加、データ量の変更を行った場合、 [ロジック作成]メニューを使用してヘッダーファイル SimpleCode.h を作成しなおし、再コンパイルが 必要となります。

## <u>ロジック作成</u>

## 3-3 最小限のアプリケーション

レイアウト作成プログラム SimpleMaster の [ファイル]メニュー [ロジック作成] を選択すると 次の4個のファイルがプロジェクトで設定されたロジック格納先フォルダーに作成されます。

SimpleCode.h	識別コード一覧表	(詳細は前ページ参照)
SampleDemo.h	ヘッダーファイル	(プロジェクト名.h )
SampleDemo.cpp	ソースプログラムファイル	(プロジェクト名.cpp)
SampleDemo.ini	環境設定ファイル	(プロジェクト名.ini)

#### SimpleDemo.h

#includ	e <simplepack.h></simplepack.h>		
#includ #includ	e "SimpleCode.h" e "Uriagecyo.h"		

## SimpleDemo.cpp

#include "SampleDemo.h"	
BOOL UserInit() { return TRUE; }	// 起動時処理
void UserExit () { }	// 終了時処理
void UserTimer () { }	// 100msec タイマー処理
CPtrnDisp* GetPtrnWnd (LPCTSTR {	name) // ウィンドウの生成
、 if (Istrcmp(name, "メニュー	-画面")       == 0) return (new CPtrnDisp());
if (Istrcmp(name, "売上げ軸	<pre>\$ ") == 0) return (new CPtrnDisp()); </pre>
if (lstrcmp(name, "伝票入力	J(新規)") == 0) return (new CPtrnDisp());
if (lstrcmp(name, "伝票入力	](修正)") == 0) return (new CPtrnDisp());
If (Istrcmp(name, "設備図"	) == 0) return (new CPtrnDisp());
(注) IT (ISTICMP(name, "帳票_元	上17吨) == 0) return (new CPtrnDisp());
}	

関数 UserInit アプリケーション固有の初期化処理を組込むために使用します。
 自動生成された時点では return TRUE; が記述されます。
 戻り値を FALSE とするとアプリケーションプログラムの起動は中止されます。
 関数 UserExit アプリケーション固有の終了時処理を組込むために使用します。
 関数 UserTimer 100msec 周期で呼び出されます。
 アプリケーションプログラム側で任意の機能を組込むことができます。

関数 GetPtrnWnd CRT画面とその画面を処理するクラスを関連付けし、クラスを生成します。

if (lstrcmp(name, "伝票入力(修正)") == 0) return (new CPtrnDisp());





(注) 帳票用レイアウトは関数 GetPtrnWnd で記述する必要はありませんが、自動生成 では区別できないため作成されます。(削除可、残しておいても影響はありません。)

#### ロジック作成

<u>3-4 コンパイラの設定</u>

最小限のアプリケーションプログラムを実行できるようにするため、コンパイル/リンクを行います。 こではコンパイラに Microsoft Visual C++ V6.0 を使用します。 SimpleMaster はデフォルトの設定でインストールされたものとして説明しています。 (インストール先フォルダー名 C:¥Program Files¥SimpleMaster)

# 3-4-1 プロジェクトの新規作成

Microsoft Visual C++ を起動し、 [ファイル]

[新規作成]を選択します。





Win32 Applocation	
SimpleDemo	SimplrMaster でレイアウト作成時に設定した
	フロジェクト名
C:¥Program Files¥	SimpleMaster¥Sample¥Demo¥Prog
	SimplrMaster でレイアウト作成時に設定した
	ロジック格納先フォルダー名
、ックスで OK を押	します。
	Win32 Applocation SimpleDemo C:¥Program Files¥ 、 、ックスで OK を押



のアプリケーションが作成される	£1.	
ロジェクトに対して作成またはは	自加されるファイルはありません。	



3-4-2 プロジェクトを構成するファイルの設定



[すべての構成] を選択し MFCのスタティクライブラリ を使用 を選択します。



[C/C++] を選択し プリプロセッサ、 インクルードファイルの パスを設定します。



ブロジェクトの設定 91 × [リンク]を選択し (後定の対象(法): すべての構成 一般 F'A'9が D/D++ 切り カース Midi アミュー オブジェクト/ライブラリ . B Simple Demo モジュールに (Y)VETH 地外国 ٠ Source Files imm32.lib Header Files 因力7+14名00 SimplePack.lib E Resource Files オブジェクトノライブラリ もうシールし) を設定します。 imm32.Rb Simple Pack Ib デムシジ情報を生き サフォルトライブリをすべて素積の imm32.lib は日本語IME 7177-14を行う(1) スペースで区切ります。 を直接制御するために MAP 77-(A 长生成才 必要です。 夫達オブション(①) Inver32. Ib Simple Pack Ib / no logo / outpsystem windows / machine 1388 16 -OK. 年的地质

SimplePack.lib へのパス を設定します。

Release と Debug モード でパスの最後が異なります。





[ビルドメニュー]、[リビルド]または[バッチビルド]でコンパイル、リンクを行います。



## 3-5 アプリケーションの実行

アプリケーションの実行には環境設定ファイル SimpleDemo.ini が必要となります。 このファイルは、レイアウト作成プログラム SimpleMaster で [ファイル]メニュー [ロジック作成]を 選択すると 自動生成されます。

コンパイラで実行プログラムは Debug および Release サブフォルダーに作成されます。 実行プログラム SimpleDemo.exe と環境設定ファイル SimpleDemo.ini は同一のフォルダーに格納 する必要があるので、ここでは手作業で環境設定ファイルをそれぞれのサブフォルダーに複写して ください。

以上で準備は終了し、プログラムを実行することができます。

#### 3 5-1 メンテナンス画面の追加

最小限のアプリケーション では、メニュー選択、前画面、終了ボタンの操作を処理することができます、 メンテナンス画面を追加すると、データを注入し画面に表示することができるようになります。 メンテナンス画面はプロジェクトを新規作成するとバージョン情報画面と共に自動的に複写されてい ます。

メンテナンス画面を使用できるようにするためにプログラムの追加/修正は必要ありませんが、メンテナン ス画面を呼び出すためにレイアウトの変更が必要です。

#### レイアウトの修正

CRT 画面上の操作でメンテナンス画面、およびバージョン情報画面を使用できるようにするため レイアウトの修正を行います。

バージョン情報画面、メンテナンス画面は親ウィンドウ/タイトル領域の左端に配置されたロゴ マーク(ビットマップ)を

マウス左ボタンでクリック	バージョン情報画面	
マウス右ボタンでクリック	メンテナンス画面	を開くように設定することができます。

プロジェクトの修正

2011年11日本 2011年11日本 2011年11日本 2011日 2011	ビットフップファイト
	広張子 .bmp は不
クトの梯正	
わつどう 子ウインドウ その地	
	- 幕告 (40 一) 和景色 0 育景色 1
	(デパージョン情報画面
	〒 デバッグ画面
	〒 ハードコピーオタン
	3イトル 又子Jオント MAS ゴシック 24×12 お'-みド
	表示証明 (* 在場 (* 平央
	The second www.item.Hot B (B) that miny
	ATTAN MAS 1990 TEXE + - AF
	作業領域
親ウィンドウ	背景色 2
	100/修缮
2003年12月17日(水)12時:45分	著 あち 190 王 前果色 3 1000
2003年12月17日(水)12時145分	西 あさ 50 王 背景色 3

ロゴマークは 参照 ボタンを押し、ハードディスク上で探してください。 ここで選択されたビット マップファイルは、レイアウト情報が格納されたフォルダーへ複写されます。 [既存のレイアウト情報をすべて変更] をセットして登録してください。 3-5-2 メンテナンス画面からのデータ注入

アプリケーションプログラム実行中にロゴマークをマウス右ボタンでクリックするとメンテナンス画面 が開きます。

一売上げ帳				200	(908月03日(米)	100\$30分 🚆
日 日 1 01 / 20 01 / 21 01 / 25 01 / 25		をマウス右ボタン 	vでクリック 12	単価 380	合計 4.560 区 6.400 素 3.000	
01 / 25	HILANE HILANE HOON IN GOURE IN HILANE IN HILANE IN HILANE IN SOURYOU T	2 C 0000 3 C 0000 4 0000 5 C 0000 6 C 0000 7 C 0000 8 C 0000 8 C 0000 9 C 0000	0,00 01 / 0,00 01 / 0,00 01 / 0,00 00 0,00 0 0,00 0 0,00 0		8.000	
					19, 960	
伝幕入力	ディスク保存	(DB)				前音面

売上げ帳 日付の欄は 識別コード HIZUKE+0,1,2 ~ 11 の文字列情報を表示するように レイアウト情報で設定しています。

メンテナンス画面で識別コード HIZUKE をマウス左ボタンで選択します。

STRING 欄でデータを注入したい行をマウス左ボタンでクリックします。 キーボードより注入したいデータを入力します。(Enter キーで入力終了)

メンテナンス画面で設定されたデータは直ちに画面に表示されます。

#### 売上げ帳画面処理プログラム 3 - 6

売上げ帳画面を処理するためのクラス CPtrnUriagecyo を作成します。 一般のクラスと同様に 定義部 と 実装部 に分けて作成します。

#### 3-6-1 CPtrnUriagecyo クラスの定義部 (ヘッダーファイル: Uriagecyo.h)

class CPtrnUri	agecyo :	public CPtrnDis	р	// 派生クラスの定義
r protected virt virt	: ual void ual void	PtrnlnitFunc PtrnMesgFunc	: (); : (int);	// 表示開始処理 // マウス入力時処理
private:	int int int DENPYO	DispLine; TotalLine; SelctLine; Denpyo[RECORD_D	ENPY0];	// 表示開始行番号 // 伝票設定済み行数 // 選択された行番号 // 伝票データ領域
	void void void	SetDispData SetDenpyo GetDenpyo	(int); (int); (int);	// 伝票データ表示 // 入力データ設定 // 入力データ取得
};	void	PrintUriagecyo	();	// 帳票印刷

クラス CPtrnUriagecyo を宣言します。

SimpleMaster / SimplePack を使用するアプリケーションプログラムでは、CRT 画面ごとに 基本クラス CPtrnDisp より派生したクラスを定義します。

メニュー画面のように基本クラス CPtrnDisp の機能で画面を処理できる場合は不要です。

売上げ帳画面 独自の処理を記述するために仮想関数をオーバーライドします。

PtrnInitFunc	CRT画面が開かれる前に実行されます。 表示データの初期化、
	スクロールバーの初期化などの処理を行います。
DtroMoogEupo	フウフナボタン撮佐などで注これるメッセージに対応した如理を

マウス左ボタン操作などで送られるメッセーシに対応した処理を PtrnMesgFunc 記述します。

売上げ帳画面 のみで使用する作業領域、関数を定義します。

画面に表示する 伝票データ はディスクファイルに保存されており、売上げ帳画面を表示する 時にのみ CPtrnUriagrcyo のデータメンバ領域 Denpyo に読込まれます。 伝票データの構造はこのヘッダーファイルで次のように定義します。

const int RECORD_DENPYO = 500;	// データ容量
<pre>struct DENPYO {     int date1, date2;     int tanka;     int suuryou;     int goukei;     char hinmei[22]; }</pre>	// 伝票フォーマット // 月・日 // 単価 // 数量 // 合計 // 品名

## <u>3-6-2 CPtrnUriagecyo クラスの実装部</u>

(ソースファイル: Uriagecyo.cpp)

## 仮想関数 PtrnInitFunc

```
void CPtrnUriagecyo :: PtrnInitFunc ()
                                                           // 表示開始時処理
 {
  FileRead(ProjPath, "URIAGECYO", Denpyo, sizeof(Denpyo));
                                                          // ファイル読込み
  TotalLine = SelctLine = 0;
                                                          // 行番号初期化
   for (int i=0; i<RECORD_DENPYO; ++i) {</pre>
                                                          // 登録済みレコード数
       if (Denpyo[i].date1 != 0) ++TotalLine;
       else
                              break:
  PtrnSetScroll(SCROLL, TotalLine, 0, S_FILE, S_LINE);
                                                         // スクロールバー初期化
 SetDispData(0);
                                                          // 初期画面表示
  ShowWindow(SW_SHOW);
}
```

```
売上げ帳データが格納されたファイルを読込みます。

関数 FileRead は SimplePack で定義されています。

作業領域の初期化と有効なデータ数を取得します。

日付情報が設定されていれば有効なデータとみなします。

スクロールバーの初期設定を行います。

CRT 画面の表示情報を共有データ領域へ設定します。

CRT 画面を表示します。 SimplePack では CRT 画面を非表示の状態で作成します。

仮想関数 PtrnInitFunc を実装した場合には必ずこの行の記述が必要です。
```

```
仮想関数 PtrnMesgFunc
```

```
void CPtrnUriagecyo :: PtrnMesgFunc (int code)
                                                               // マウス入力時処理
{
    if (code == NYUURYOKU) {
                                                               // 伝票入力(新規)
         SetDenpyo(TotalLine);
         PtrnDispChild(1, "伝票入力(新規)", S_FILE, S_LINE);
    }
    else if (code == HOZON ) FileWrite(ProjPath, "URIAGECYO", Denpyo,
                                                                   // ファイル書込み
                                                 sizeof(Denpyo));
    else if (code == PRINT ) PrintUriagecyo();
                                                               // 帳票印刷
    else if (code == SCROLL) SetDispData(Get_iData(SCROLL));
                                                               // スクロールバー
    else if (code == TOUROKU) {
                                                               // 登録(新規)
        PtrnCloseWnd(1);
                                                               // 入力画面を閉じる
                                                               // 入力結果取得
         GetDenpyo(TotalLine);
         ++TotalLine;
         PtrnSetScroll(SCROLL, TotalLine, ++DispLine, S_FILE, S_LINE); // スクロールバー再設定
         SetDispData(Get_iData(SCROLL));
                                                               // 再表示
    }
    else if (code == TOUROKU+1) {
                                                               // 登録(修正)
         PtrnCloseWnd(1);
                                                               // 入力画面を閉じる
         GetDenpyo(SelctLine);
                                                               // 入力結果取得
         SetDispData(DispLine);
                                                               // 再表示
    }
    else if (code == SAKUJO) {
                                                               // 削除
         PtrnCloseWnd(1);
                                                               // 入力画面を閉じる
         for (int i=SelctLine; i<=TotalLine; ++i)</pre>
             Denpyo[i] = Denpyo[i+1];
         --Totall ine:
         PtrnSetScroll(SCROLL, TotalLine, DispLine, S_FILE, S_LINE); // スクロールバー再設定
         SetDispData(Get_iData(SCROLL));
                                                               // 再表示
    }
```

```
// 行選択
else {
     int line = -1;
     i f
             ((code >= HIZUKE ) && (code < (HIZUKE +12))) line = code - HIZUKE;
     else if ((code >= HINMEI ) && (code < (HINMEI +12))) line = code - HINMEI;
     else if ((code >= TANKA ) && (code < (TANKA +12))) line = code - TANKA;
     else if ((code >= SUURYOU) && (code < (SUURYOU+12))) line = code - SUURYOU;
     else if ((code >= GOUKEI ) && (code < (GOUKEI +12))) line = code - GOUKEI;
     if (line != -1) {
          if ((DispLine + line) < TotalLine) {</pre>
                                                                // 修正
               SetDenpyo(SelctLine = DispLine + line);
              PtrnDispChild(1, "伝票入力(修正)", S_FILE, S_LINE);
         }
    }
}
```

仮想関数 PtrnMesgFunc は画面上に配置された表示要素をマウスでクリックした時、処理種別 コードに[メッセージ送信] が設定されている場合に呼び出されます。 引数には 識別コード が設定されます。

売上げ帳画面で[伝票入力]ボタンをマウス左ボタンでクリックされた時の処理を記述 します。 入力用データ領域を初期化し、伝票入力画面[新規]を子ウィンドウ1に表示 します。

売上げ帳画面で [登録] ボタンをクリックされた時の処理で、売上げ帳データをディスク に保存します。

帳票印刷を行います。 (3 - 7参照)

スクロールバーを操作された時の処理を記述します。 スクロールバーは Simple Pack 側で処理されています。(マウスホイール、キーボード にも対応しています。)

スクロールバーが操作されると、操作された内容や設定値(表示量、データ量、移動量) に従って表示開始位置を計算し、識別コードで設定された共有データ領域に設定し 仮想関数を呼び出します。

アプリケーションプログラム側では共有データ領域より表示開始位置を取得し、1ページ分の再表示を行います。

、 伝票入力画面で 登録、削除」ボタンを操作された時、子ウィンドウからメッセージが送られます。 親ウィンドウ(売上げ帳画面)では、子ウィンドウの消去 (PtrnCloseWnd)、
 入力結果の取り込みやデータの削除を行い画面を再表示します。
 子ウィンドウの 取消 ボタンは処理種別コードで [自ウィンドウを閉じる] と設定されているのでアプリケーションプログラムで処理する必要はありません。

売上げ帳の表内部をマウス左ボタンでクリックされた時の処理を記述します。 引数で渡された識別コードより行番号を計算し、その行に有効なデータが表示されていれば 修正画面を開きます。

void CPtrnUriagecyo :: SetDispData (int	t line)	// 表示データ設定
DispLine = line;		
Clear_AllData(HIZUKE ); Clear_AllData(HINMEI ); Clear_AllData(TANKA ); Clear_AllData(SUURYOU); Clear_AllData(GOUKEI ); for (int i=0; i<12; ++i) { int datano = DispLine + i; if (datano < TotalLine) { CString date:		
date.Format("%02d / %02d",	Denpyo[datano].date1, Denpyo[datano].date2);	// 日付変換
Set_sData(HIZUKE +i, date, Set_sData(HINMEI +i, Denpyo Set_iData(TANKA +i, Denpyo Set_iData(SUURYOU+i, Denpyo Set_iData(GOUKEI +i, Denpyo }	S_FILE, S_LINE); b[datano].hinmei, S_FILE, S b[datano].tanka, S_FILE, S b[datano].suuryou, S_FILE, S b[datano].goukei, S_FILE, S	// 表示データ設定 _LINE); _LINE); _LINE); _LINE);
else { PtrnSetDisable(TANKA +i, ] PtrnSetDisable(SUURYOU+i, ] PtrnSetDisable(GOUKEI +i, ] }	TRUE, FALSE); TRUE, FALSE); TRUE, FALSE);	// 非表示設定
, int goukei = 0; for (i=0; i <totalline; ++i)<br="">goukei += Denpyo[i].goukei; Set_iData(SOUGOUKEI, goukei, S_FILE,</totalline;>	S_LINE);	// 総合計
J		

1ページ分のCRT画面表示情報を共有データ領域へ設定します。 引数には表示開始行番号が渡されます。 SimpleMasterを使用したアプリケーションではCRT画面への表示は専用の関数を使用した 共有データ領域へのデータ設定で行われます。

CRT 画面表示に使用する共有メモリ領域を初期化します。

WORD 型データ領域には 0x0000,数値データ領域には 0.0、文字列データ領域には NULL 文字が設定されます。

1ページの行数分繰り返します。

関数 Set\_xData を使用してクラス内の作業領域より CRT 画面表示用の共有データ領域へ 表示データを設定します。 関数 Set\_xData の引数 S\_FILE, S\_LINE はコンパイル時にソースファイル名と行番号が 展開されます。共有データ領域への設定エラーが発生した場合やメンテンンス画面でモニタ が指定されているとログファイルに設定値とともに設定元のソースファイル名 / 行番号が出力 されます。

CRT 画面の途中で有効なデータの表示が終了した場合の処理です。 文字列データの場合は NULL 文字で初期化しますので()画面上には何も表示されま せんが、数値データは 0 が表示されます。 関数 Set\_Disable を使用して表示禁止に設定 します。

総合計値を計算し共有データ領域へ設定します。

ローカル関数 SetDenpyo, GetDenpyo

void CP	trnUriagecyo :: SetDe	enpyo (int line)		// 伝票入力データ設定
Set Set Set Set Set	_iData(IN_HIZUKE, _iData(IN_HIZUKE+1, _sData(IN_HINMEI, _iData(IN_TANKA, _iData(IN_SUURYOU, _iData(IN_GOUKEI,	Denpyo[line].date1, Denpyo[line].date2, Denpyo[line].hinmei, Denpyo[line].tanka, Denpyo[line].suuryou, Denpyo[line].goukei,	<pre>S_FILE, S_LINE); S_FILE, S_LINE); S_FILE, S_LINE); S_FILE, S_LINE); S_FILE, S_LINE); S_FILE, S_LINE);</pre>	// 月 // 日 // 品名 // 単価 // 数量 // 合計
void CP	trnUriagecyo :: GetDe	enpyo (int line)		// 伝票入力データ取得
Den Den Den Den Den Ist	pyo[line].date1 = 0 pyo[line].date2 = 0 pyo[line].tanka = 0 pyo[line].suuryou = 0 pyo[line].goukei = 0 rcpy(Denpyo[line].hir	Get_iData(IN_HIZUKE, Get_iData(IN_HIZUKE+1, Get_iData(IN_TANKA, Get_iData(IN_SUURYOU, Get_iData(IN_GOUKEI, nmei, Get_sData(IN_HIN)	S_FILE, S_LINE); S_FILE, S_LINE); S_FILE, S_LINE); S_FILE, S_LINE); S_FILE, S_LINE); MEI, S_FILE, S_LINE;	// 月 // 日 // 単価 // 数量 // 合計 )); // 品名

関数 SetDenpyo 伝票入力画面(新規/修正)を開く前に表示データを設定します。 関数 GetDenpyo 伝票入力画面(新規/修正)で 登録 ボタンが押され親ウィンドウ(売上げ 帳画面)でメッセージを受け取った時、修正結果を取得します。

## 3-7 伝票入力画面処理プログラム

売上げ帳1行分のデータをキーボードより入力するための画面処理プログラムを作成します。 ここで作成するプログラムでは 新規入力画面 と 修正入力画面 の処理を行います。

<u>3-7-1 派生クラスの定義部</u> (ヘッダーファイル: Uriagecyo.h)

cla {	ass CPtrnDen	ιργο : ρι	ublic CPtrnDisp		//	伝票入力画面
	protected virtual virtual	l: void void	PtrnlnitFunc PtrnKeyFunc	(); (int, int, PTRNDATA*);	 	表示開始処理 キー入力処理
};	private:	void	SetButton	();	//	ボタン操作可否設定

#### 派生クラス CPtrnDenpyo を定義します。

仮想関数 PtrnKeyFunc はキーボード入力が終了した時に呼び出され、入力を継続するための 処理と、入力完了項目を判定し、 登録 ボタンの操作の可否の設定、合計値の計算を行っています。

<u>3 - 7 - 2 派生クラスの実装部</u> (ソースファイル : Uriagecyo.cpp)

仮想関数 PtrnInitFunc は画面を表示する前に呼び出されます。

void CPtrnDenpyo :: PtrnInitFunc ()	// 表示開始時処理
<pre>SetButton();</pre>	ここに KeyInputStart(IN_HIZUKE); と記述
ShowWindow(SW_SHOW);	すれば画面を開いた時点でキーボード
}	入力を開始することができます。

この仮想関数の最後に CRT 画面を表示するために次の行が必要です。 ShowWindow(SW\_SHOW);

仮想関数 PtrnKeyFunc はキーボード入力が終了した時に呼び出されます。 SimplePack ではキーボードの入力は次の手順で行います。

入力項目の選択

[入力可]が設定された表示項目をマウスでクリックするか、プログラム側より KeyInputStart 関数を実行することでキーボード入力処理を開始します。

入力開始の通知 仮想関数 PtrnKeyStart が呼び出されます。 ガイダンスメッセージの表示などを行う場合に実装します。

キーボード入力処理 入力種別(文字列または数値データ)に対応して入力処理を行います。 入力終了キーが入力されエラーチェックが終わるまでの処理は SimplePack 側で処理され ます。

キーボード入力終了通知 仮想関数 PtrnKeyFunc を呼び出します。

void CPtrnDenpyo :: PtrnKeyFunc (int Key, int Shift, PTRNDA	TA* Ptrn) // キー入力終了時処理
SetButton(); int code = Ptrn->DataNo; if ((Key == VK RETURN)    (Key == VK TAB)) {	キーボード入力が行われ た表示項目の識別コード l理
if (code == IN_HIZUKE+0) code = IN_HIZUKE+1; else if (code == IN_HIZUKE+1) code = IN_HINMEI;	
else if (code == IN_HINMEI ) code = IN_TANKA; else if (code == IN_TANKA ) code = IN_SUURYOU; else if (code == IN_SUURYOU ) code = IN_HIZUKE+0;	
KeyInputStart(code); }	
<pre>else if (Key == VK_DOWN) {     if (code == IN_HIZUKE+0) code = IN_HINMEI;     else if (code == IN_HIZUKE+1) code = IN_HINMEI;     else if (code == IN_HINMEI ) code = IN_TANKA;     else if (code == IN_TANKA ) code = IN_SUURYOU;     KeyInputStart(code);</pre>	
}	
以下省略	

仮想関数 PtrnKeyFunc では入力終了時のキーコードと入力された表示項目より次に入力を継続 すべき項目を選択します。

入力終了時のキーコードには次の7種類が渡されます。

No.	キーコード	プログラムでのコード	備考
1	Enter	VK RETURN	
2	Tab	VK_TAB	
3		VK_UP	有効なキーを個別に指定する
4		VK_DOWN	ことができます。
5		VK_LEFT	
6		VK_RIGHT	
7	Esc	VK_ESCAPE	入力処理の終了とします。

カーソル移動キーはレイアウト作成時に描画要素の詳細設定画面で個別に指定することが できます。

	☑ 入力可				
	□ 日本語入力				
カーソル					
処理種別	(なし)				
画面名称	*				

入力終了キーコードと入力処理が行われた表示項目の識別コードより次に入力する項目を選び その識別コードを引数として KeyInputStart を実行すればキーボード入力処理を再開すること ができます。

TAB キーが入力された場合、引数で渡される シフトキーの状態を判定して次のキー入力を行う 識別コードを選択することができます。 (Shift = 1 : Shift キー押下中) 登録 ボタンの操作の可否と合計値の計算

void CPtrnDenpyo :: SetButton ()	// 登録ボタン
BOOL flag = TRUE; if (Get_iData(IN_HIZUKE+0) == 0) flag = FALSE; if (Get_iData(IN_HIZUKE+1) == 0) flag = FALSE; if (Get_iData(IN_TANKA ) == 0) flag = FALSE; if (Get_iData(IN_SUURYOU ) == 0) flag = FALSE; CString hinmei = Get_sData(IN_HINMEI); if (hinmei.GetLength() == 0) flag = FALSE;	// 月 // 日 // 単価 // 数量 // 品名
if (flag == TRUE) OFF_wData(TOUROKU, 0); else ON_wData (TOUROKU, 0);	// 操作可 // 操作不可
<pre>int tanka = Get_iData(IN_TANKA ); int suuryou = Get_iData(IN_SUURYOU); int goukei = tanka * suuryou; Set_iData(IN_GOUKEI, goukei); }</pre>	// 合計 算出

すべての入力項目が設定されている場合に 登録 ボタンの操作を可能とします。 ボタンに設定された識別コードの WORD 型データ、最下位ビットを ON に設定すればボタンの 入力は不可となります。 ボタンの文字も灰色表示となります。

# <u>3-8 帳票印字処理プログラム</u>

レイアウト情報を使用した帳票印刷は、CRT 画面表示と同様に共有データ領域へ印字データを設定する ことで行います。

売上げ帳の印刷は 関数 PrintUriagecyo で行います。この関数は売上げ帳画面に配置されたボタンを マウス左ボタンでクリックされた時に実行されます。

## 3-8-1 帳票印刷プログラムの構造

帳票印刷プログラムは 売上げ帳 以外の帳票の場合でも同一の構造で作成することができます。



関数 PrintOpen, PrintClose, PrintNewPage は印刷用デバイスコンテキスト CPrintDraw のメンバ関数です。
関数 PtrnDispData はレイアウトデータ管理クラス CPtrnData のメンバ関数です。 <u>3-8-2</u> 売上げ帳 帳票印刷プログラム

```
const
       char*
               PRINT_PTRN = "帳票_売上げ帳";
                                                                       // 帳票名称
               PRINT_LINE = 12;
                                                                       // 印字行数
       int
const
void CPtrnUriagecyo :: PrintUriagecyo ()
                                                                       // 帳票印刷
{
    CPrintDraw DC(this, "URIAGECYO");
                                                                       // プリンターデバイス
    CPtrnData PtrnData(PRINT_PTRN);
                                                                       // レイアウト情報読込み
    if (DC.PrintOpen("売上げ帳", PtrnData.PtrnHedr.Parm1[1],
                                                                       // 印字開始処理
                                PtrnData.PtrnHedr.Parm1[2]) == TRUE) {
         int Page = 1 + (TotalLine-1) / PRINT_LINE;
                                                                       // 総ページ数
         for (int page=0; page<Page; ++page) {</pre>
              Set_iData(P_PAGE1, page+1);
                                                                       // ページ番号
              Set_iData(P_PAGE2, Page );
              Clear AllData(P HIZUKE );
                                                                       // 共有データ領域消去
              Clear_AllData(P_HINMEL );
              Clear_AllData(P_TANKA );
              Clear_AllData(P_SUURYOU);
              Clear_AllData(P_GOUKEI);
              for (int line=0; line<PRINT_LINE; ++line) {</pre>
                   int datano = PRINT_LINE * page + line;
                   if (datano < TotalLine) {</pre>
                       CString date;
                       date.Format("%02d / %02d", Denpyo[datano].date1, // 日付変換
                                                 Denpyo[datano].date2);
                                                                       // 印字データ設定
                       Set_sData(P_HIZUKE +line, date);
                       Set_sData(P_HINMEI +line, Denpyo[datano].hinmei );
                       Set_iData(P_TANKA +line, Denpyo[datano].tanka );
                       Set_iData(P_SUURYOU+line, Denpyo[datano].suuryou);
                       Set_iData(P_GOUKEI +line, Denpyo[datano].goukei );
                   }
                   else {
                       Set_Disable(P_TANKA +line, TRUE, FALSE);
                                                                       // 非印字設定
                       Set_Disable(P_SUURYOU+line, TRUE, FALSE);
                       Set_Disable(P_GOUKEI +line, TRUE, FALSE);
                   }
              .
PtrnData.DispPtrnData(&DC);
                                                                       // 1ページ印刷
                                                                       // 最終ページ
              if (Page == (page+1)) break;
              else DC.PrintNewPage();
                                                                       // 改ページ
         DC.PrintClose();
                                                                       // 印刷終了
    }
}
```

印字用デバイスコンテキストを作成します。 引数 "URIAGECYO" より環境設定ファイルを参照し印字に使用する用紙などの情報を取得します。

[PRINTER] URIAGECY0=2,1,0,10,10 ; 売上げ帳 印字

設定情報の数値は左側より次の内容を設定します。

用紙サイズ	1:A 3 / 2:A 4 / 3:A 5 / 4:B 4 / 5:B 5
印字方向	0:縦 / 1:横
給紙装置	0:自動選択 / 1:上用紙トレイ / 2:中用紙トレイ / 3:下用紙トレイ
	4:用紙カセット / 5:手差しフィーダ / 6:フロントトレイ
左右方向余白	印刷する用紙の大きさに対する比率(%)で設定します。
上下方向余白	

用紙サイズに 0 を設定すると、プリンター使用開始時に [プリンター設定] のダイアログ ボックスが表示され、プリンターの選択、用紙、印字方向などを設定することができます。

レイアウト処理クラス CPtrnData を作成します。引数で指定されたレイアウトファイルが読込まれます。

印字開始処理を行います。

1番目の引数は帳票名称でプリンタースプーラーに渡されます。 2、3番目の引数でレイアウトの大きさ (横×縦 単位:ドット数)を設定します。 印字用デバイスコンテキストでは用紙の大きさとレイアウトの大きさを比較し拡大 / 縮小印字を行い ます。

ページ数分繰り返します。

1ページ分の印字データ領域を初期化します。 WORD、数値データ領域には 0、文字列データ領域には NULL 文字が設定されます。

1ページの行数分繰り返します。

共有データ領域へ印字データを設定します。

印字するデータが存在しない行では、文字列は空白で印字されますが、数値データは "0" が印字され ます。 数値データも空白で印字するために非印字の設定を行います。

印字用デバイスコンテキストへ出力を行います。

改ページ処理を行います。

印字終了処理を行います。この時最終ページが印字されます。

#### 3-9 制御用データ処理プログラム

制御データは画面処理部の外で作成します。 (ソースファイル名 : SampleDemo.cpp)

<u>3-9-1 スレッドの起動</u>

制御データを作成するためスレッドを起動します。

スレッドの起動はアプリケーションプログラムの起動時に SimplePack より呼ばれる 起動時処理関数 UserInit で行います。

static UINT	SetsubiDataThread	(LPVOID);	// 設備データ入力スレッド
BOOL UserInit	()		// 初期化処理
۲ AfxBeginTh return TRU	read(SetsubiDataThro JE;	ead, 0);	// データ処理スレッド起動
}			

#### <u>3-9-2</u> データ作成スレッド

データとして デジタル入力信号 7点、アナログ入力信号 10点 を模擬的に作成します。

static UINT SetsubiDataThread (LPVOID)	// 設備データ入力
{	// AI初期値設定
Sleep(1000);	// 起動待ち(1秒)
for (;;) { Sleep(100); if (SystemStop != 0) break;	// 100msec 周期 // スレッド停止要求
if (++timr >= 10) {	// 1秒周期
int dich = rand() % 7; EOR_wData(DI+dich, 0x0001);	// DI擬似信号設定
for (int j=0; j<10; ++j) {	// AI擬似信号設定
if (Al_Data[j] >= 80) Set_Color(Al+j, 1); else	
}	
} return (0);	
· ·	

作業領域の初期化

画面処理が開始するまで待ちます。(特別な意味はありません。)

100msec周期でシステム停止要求フラグを判定します。

CRT画面上で 終了 ボタンが押されると、この変数 SystemStop に1が設定されます。 1秒周期でデジタル、アナログ信号の作成を行います。

デジタル信号は乱数を使用して 0~6 の数値を作成し、1秒毎に1個の信号を反転させ ます。 WORD 型データの下位2ビットは表示色を決めるために使用されています。 制御用画面ではデジタル信号を表示するマークに2系統の表示色を設定していますから 最下位ビットを反転することで表示色が切り替ります。

アナログ信号10点の数値を 0~99 の範囲で設定します。

アナログ信号が 80~99 の時 WORD 型データ最下位ビットを ON に設定することで 表示色を切り替えています。

#### <u>3-9-3 設備図画面</u>

設備図画面で表示するデータはすべてデータ処理部(制御データ作成スレッド)で作成しています。 したがって、設備図画面を処理するクラスは作成せずに、基本クラス CPtrnDisp で画面処理を 行うことができます。

```
CPtrnDisp* GetPtrnWnd (LPCTSTR name) // ウィンドウの生成
{

CPtrnDisp* PtrnWnd = NULL;

if (Istrcmp(name, "メニュー画面") == 0) PtrnWnd = new CPtrnDisp();

else if (Istrcmp(name, "売上げ帳") == 0) PtrnWnd = new CPtrnUriagecyo();

else if (Istrcmp(name, "伝票入力(新規)") == 0) PtrnWnd = new CPtrnDenpyo();

else if (Istrcmp(name, "伝票入力(修正)") == 0) PtrnWnd = new CPtrnDenpyo();

else if (Istrcmp(name, "設備図") == 0) PtrnWnd = new CPtrnDisp();

return PtrnWnd;

}
```

メニュー画面、設備図画面は画面処理用のクラスを派生させずに基本クラスで処理することができます。 派生クラスの作成が必要かどうかは次の要因で決まります。

処理種別コード

CRT 画面上に配置された表示項目に設定される処理種別コードにより 派生クラスが必要になります。

No.	処理種別コード	派生クラス	備考
1	自ウィンドウへメッセージ送信	必要	
2	子ウィンドウ1ヘメッセージ送信	不要	他のウィンドウより
3	子ウィンドウ2へメッセージ送信	不要	メッセージが送られる
4	親ウィンドウヘメッセージ送信	不要	場合には必要
5	子ウィンドウ1を開く	不要	
6	子ウィンドウ2を開く	不要	
7	システムウィンドウを開く	不要	
8	最上位ウィンドウを開く	不要	
9	自ウィンドウを閉じる	不要	
10	子ウィンドウ1を閉じる	不要	
11	子ウィンドウ2を閉じる	不要	
12	システムウィンドウを閉じる	不要	
13	バージョン情報画面を開く	不要	
14	デバッグ画面を開く	不要	
15	ハードコピー	不要	
16	システム終了	不要	

CRT 画面へ表示するデータの作成が必要な場合

売上げ帳画面のようにスクロールバーに連動して表示情報を共有データ領域に設定 する必要がある場合には派生クラスの作成が必要です。

キーボード入力を継続する場合

伝票入力画面ではキーボード入力を継続することができるようにするために派生クラス を作成しています。 基本クラスでもキーボード入力処理、入力範囲のエラーチェック、 ボタン操作の親ウィンドウへの通知は可能です。

## <u>4. デモ用プログラム</u>

デモ用プログラムを完成させるためにソースプログラムの修正、再コンパイルを行います。

<u>4 - 1 ソースプログラム修正</u>

売上げ帳画面、伝票入力画面を追加します。

ヘッダーファイル名 : SimpleDemo.h

```
#include <SimplePack.h>
#include "SimpleCode.h"
#include "Uriagecyo.h"
```

ソースファイル名 : SimpleDemo.cpp

```
static UINT
                 SetsubiDataThread (LPVOID);
                                                                 // 設備データ入力スレッド
BOOL UserInit ()
                                                                 // 初期化処理
{
     AfxBeginThread(SetsubiDataThread, 0);
                                                                 // データ処理スレッド起動
     return TRUE;
}
void UserExit () { }
                                                                 // 終了時処理
void UserTimer () { }
                                                                 // 100msec タイマー処理
                                                                // ウィンドウの生成
CPtrnDisp* GetPtrnWnd (LPCTSTR name)
{
     CPtrnDisp* PtrnWnd = NULL;
     if (lstrcmp(name, "メニュー画面") == 0) PtrnWnd = new CPtrnDisp();
else if (lstrcmp(name, "売上げ帳") == 0) PtrnWnd = new CPtrnUriagecyo
else if (lstrcmp(name, "伝票入力(新規)") == 0) PtrnWnd = new CPtrnDenpyo();
                                                 ) == 0) PtrnWnd = new CPtrnUriagecyo();
     else if (lstrcmp(name, "伝票入力(修正)") == 0) PtrnWnd = new CPtrnDenpyo();
     else if (lstrcmp(name, "設備図"
                                                 ) == 0) PtrnWnd = new CPtrnDisp();
     return PtrnWnd;
}
```

売上げ帳、伝票入力画面処理プログラムの宣言部 設備画面用データ作成スレッドのプロトタイプ宣言 設備画面用データ作成スレッドの起動 売上げ帳、伝票入力画面処理クラスの関連付け

# <u>4-2 コンパイラの設定</u>

プロジェクト SimpleDemo へ2本のソースプログラム Uriagecyo.h, Uriagecyo.cpp を追加します。



# <u>4-3</u>デモ用プログラムの実行

ソースプログラムを追加し再コンパイルを行うとデモ用プログラムが完成します。 実行前に環境設定ファイルに帳票印刷用の設定が必要です。

SimpleDemo.ini

[PRINTER] URIAGECYO=2,1,0,0,0

;売上げ帳

(設定値の詳細は 3 - 7 - 2 を参照してください。)

# <u>5 スプレッドシート</u>

# 5-1 概要

SimpleMaster では次のようなスプレッドシートを作成し、親ウィンドウの指定された位置に表示することができます。

235日	o. S#SD11	333				納入	先	71/7387	Α				スプレッドシート
作香	112					52N	<u>1.</u>	2					
<b>a</b> X)	11												
					14 - AU		di.						
行番	因有释重	綿径	綿板	10	回路	記線	-	日報子	9/1'	释長	U-F No.		
1	A022A	2.0	B	E I	HC	inter.	YI	1	MS3	200 -01	A02		
2	A0228	2.0	8		HC	-	Y1	2	MS3	200 4	A02		
3	A022C	2.0	8		HC		YL	3	MS3	300 🖨	A02	10	
4	AD14A	2.0	8		HC		YI.	4	84	100 🖨	TB		
5	A0138	2.0	8		HC		YI	5	84	200 🜲	TB		
6	B05A2A	1.25	B		HC.	-	YI	6	144	100 🔹	TB		
7	B05A2B	1.25	B		HC		YI	7	M4	300 👲	B05A		
8	B05A3A	1.25	B		HC.		11	8	84	100 🚭	TB		
. 9	B05A3B	1.25	B		HC.	-	Y1	9	144	200 🚭	B05A		
10	BOGA3A	1.25	8		HC		¥1	10	84	300 🏚	806A		
11	B06A7A	1.25	8		HC		YI.	11	144	100 축	806A		
12	B06A7B	1.25	8		HC.		YL	12	84	200 축	B06A		
13	P1601	1.25	8		HC:		YI.	13	84	200 축	803		
14	N1801	1.25	8		HC		YI.	14	84	100 🚖	B18	王	
	10.000								100	253700	C. Market		
			_			10				for the second second second			

スプレッドシート用レイアウト

C.W.	1124-10.36	10.07	10.00	Æ	回路	<b>N</b> 249	5	Ĥ		2		相		千	線長	シート	
0.	EN TINK B	**12	****	-	区分	和定	座標	端子	9-17.	処理	庄稼	端子	3.12.	処理	(mm)	No.	
1111	********	tttt	4444		4466	6111	4466	1111	1111	derate.	6111	4468	\$555	Extere.	11100 Å	rancer :	*
****	*******	****	\$544		\$444	****	4465	\$544	2222	******	6888	****	1111	******	****** 🍯	FRACES 3	
1334	********	1111	\$111		1111	6644	4468	\$544	1114	eeraat.	6644	4468	\$\$54	ranter	11100 츂	<b>EXACLE</b>	
****	********	****	\$444		\$444	****	4468	****	2222	******	****	4468	1111	******	****** 출	FRARET	
****	<b>EXAMPLES</b>	1111	1111		4444	6684	4468	1111	6884	derate.	6644	4468	1111	ranter	111111 查	TALLET	
****	********	****	\$344		\$442	2222	4468	\$544	****	******	2222	4468	\$544	FRAder	111111 着	FRATER	
****	<b>SUGGERSS</b>	4668	8888		6666	6666	4468	\$844	6684	******	6666	4468	\$844	reacce	****** 싶	raaaee .	2
****	********	****	4444		\$448	****	4468	\$544	****	erraad	****	4468	\$\$55	******	111111 🛓	ERAFEE .	Ē
1111	enterne .	1111	1000		11113	11111	4466	1000	1111	111111	1111	4466	1000	111111	111111 -	ERAFEE.	
	********	****	\$544		1111	****	4448	1111	****	******	****	4444	1111	estatee	****** <b>4</b>	******	
****	********	tter	1111		1111	6644	1111	1111	<b>FFFF</b>	******	6644	1111	1111	estater.	111111 A	FRATER	
1111	********	ttet	\$546		\$444	2222	4448	\$544	2225	eerste.	2222	4448	\$\$56	states.	111111 🛓	FRANCE	
****	********	1111	4444		\$444	1111	tter	\$555	2222	******	2224	4468	11111	FRAME.	****** 축	ranner .	÷
	anneeran	tter	1111		4444	2222	4468	1111	2225	derate.	****	4468	1111	annee.	11100 至	states 3	Ŧ
														合計	******		
														<b>F</b> H	6		

スプレッドシートには次の情報を表示することができます。

可変文字列 キーボード入力可 数値データ キーボード入力可、スピンボタンの付加が可能 チェックボックス マーク (2値のビットマップ)

スプレッドシート上には通常のレイアウトを配置することができます。 上記の例では、1/2行目のタイトル、下部の合計値、罫線は通常のレイアウト情報で描画 しています。

スプレッドシートでは表の一部をスクロール禁止に設定することができます。 上記の例では、上部2行、下部1行、左側1列、右側2列 をスクロール禁止に設定しています。 スプレッドシート上では次の処理を行うことができます。

(アプリケーションプログラム側での記述は不要です。)

- 1. 可変文字列、数値データなどの表示、定周期での表示更新
- 可変文字列、数値データに対するキーボード入力 マウスで入力を行うセルを選択(画面を開いた時に自動起動することも可能です。) 入力終了キー(Enter / TAB / Shift+TAB / カーソルキー)に従って次に入力を行うセルを 選択します。この時、セルは非表示の状態であれば自動スクロールします。 入力を自動継続しない、または自動で選択するセルとは別のセルで入力を継続することも 可能です。(アプリケーションプログラム側での記述が必要)
- 3. 上下・左右方向へのスクロール (マウスホィールも使用可能)
- アプリケーションプログラム側では次の処理の記述が必要となります。
  - 1. 親ウィンドウ側で表示位置を指定してスプレッドシートを子ウィンドウとして表示します。
  - 2. 実際に表示を行うデータ行数を指定します。
  - 3. スプレッドシートに表示するデータを共有データ領域に書込み。
  - 4. キーボード入力結果を共有メモリ領域から読込み。

スプレッドシートからは、マウス操作時、キーボード入力完了時にメッセージを出すことができます。 前ページの例では 線長の合計値の算出はキーボード入力完了、およびスピンボタンの操作を メッセージとして受け取り、アプリケーションプログラム側で再計算しています。

# <u>5-2 レイアウト作成</u>

スプレッドシートのレイアウトは、レイアウト作成プログラム SimpleMaster で作成します。

#### <u>5-2-1 レイアウトを開く</u>

## 新規作成時

[ファイル]メニュー [新規作成] [スプレッドシート] と選択することで詳細設定ダイアログボックス が開きます。

#### 修正時

通常のレイアウトと同様に

[ファイルメニュー] [レイアウトを開く] によりスプレッドシート用のレイアウトを開きます。 Microsoft Explorer 上でレイアウトファイルを ダブルクリックすることで開くこともできます。



[注意] スプレッドシートのレイアウト情報は通常のレイアウト情報と同一種類のファイル (ファイル名拡張子 ptn)に格納されます。

#### <u>5-2-2 詳細設定ダイアログボックス</u>

スプレッドシートはダイアログボックスへの設定でレイアウトが自動生成されます。

詳細設定ボックスの各項目 に値を設定することで表の 大きさ、色などが決まります。

[データ列設定] ボタン を押すと、個々の列に表示 する情報を指定することが できます。

[スクロールバー] ボタン を押すことでスクロールバー の大きさや色などを設定する ことができます。

奇域の大きさ 99	0 × 362		
リアか名 スラレ	ポシート1		
行の設定		列の設定	
1行の高さ 20	*	1 桁加 9編	8 .
データ行数 17	*	データ列数	17 -
上额国定行数 2	20	左側固定列数	1 =
下部国定行数	- 20	右側固定列数	2 -
第月ロード 14	-	表示桁数	R5 -
清去色 2	霎線表示色 1	外存表示的	A. 2
未設定積塔 21	11 M 10	· 카취4월	P =
表示デー挑散定方式	C -18 0	18881	
フォント	MS ゴシック16	×8 #'-#F'	*
マウスカーソル形状	@ 矢印 (	十字 ① 砂時計	
キー入力時	転要色 8	1 背景色	9

#### 列の設定

1桁の幅をドット数で指定します。
表の大きさの管理は 行 / 桁数で行います。 データを表示する
列はそれぞれ異なる桁数を設定することができます。
データとして表示する行数を指定します。
表の左右にスクロールを行わない列を指定することができます。

- 識別コード データ列に表示する可変文字列などに割り当てる識別コードの総数を指定します。 表示データ設定方式により設定内容が異なります。(4 表示データ設定 を参照)
- 表示桁数 スプレッドシートを表示する時、親ウィンドウ上の座標を指定する方法と、矩形領域 を領域を指定する方法があります。開始座標のみを指定する場合に表示桁数が 有効となります。(5 - 3 スプレッドシート表示 を参照)

#### 消去色 / 外枠表示色



**罫線表示色/罫線幅** 表の内部に罫線を表示する場合に設定します。 罫線幅に 0 以外の数値が 設定されている場合に有効となります。

**文字フォント** データ列に表示される可変文字列、数値データに使用する文字フォントを 指定します。(文字フォントは個別に指定することはできません。)

**表示データ設定方式** 表示データ設定方式 一括 または 個別 を設定します。 (5-4 表示データ設定 を参照)

# <u>スプレッドシート</u>

# <u>5-2-3</u>データ列設定

レイアウト詳細設定ダイアログ ボックスで[データ列設定] ボタンを押すと右の画面が 表示されます。

データ列は最大 100項目まで 設定することができますが、詳細 設定ダイアログボックス / データ 列数 で指定された列数のみが 有効となります。

R	建富希益		. 4	故	歳10-7					
9	教徒データ	٠	F	-	LINENO		0	1746	14.A. MIDE	
2	可意文字列	٠	F1	-	HIVO,DATAL		0	<b>FF HE</b>	14A   HIP	
8	可充大子州	•	F	+	HIVO,DATA2	*	0	11.00	147. ARE	
4	可能大学列	٠	F	*	HV0.0ATAS		0	(THE)	10A ARE	
5	71900923		4	+	HIND, DATAN	+	p	(File)	#入 許除	
6	可爱文字列		F	*	HYO, DATAS		10	算紙	14.7. MIRE	
T	可意文学列	٠	F	+	INTAG, DVH		0	IT HE	102 ABE	
8	可充文字列	-	F	+	HVO,DATA7		þ	19.62	143 ARE	
Ø.)	可要大学列	٠	1	-22	HV0,04748		0	11.6	14X ABE	
10	可充文字列		F	+	PHIO, DAT AR		0	(Fat	183 AR	

設定項目 右端のボタンをマウス左ボタンで クリックすると選択画面が開きます。 可変文字列、数値データ、チェックボックス マーク より選択することができます。 未設定を選択した場合、桁数が設定され ていれば表として表示することができます。



- 桁数 データ列の桁数を設定します。 最大 99桁 まで指定することができます。
- 識別コード データとして表示する可変文字列などには識別コードを割当てる必要があります。
   識別コード文字列と番号を指定します。
   通常のレイアウト作成時と同様に右端のボタンをクリックすると登録済みの識別コード
   ドー覧表が表示され、一覧表より選択することができます。
- **詳細 ボタン** 可変文字列、数値データなどの設定項目に対応した詳細設定画面が開きます。 (次ページ参照)
- **挿入 ボタン** 選択されている列の直前に 未設定のデータ列 を挿入します。
- 削除 ボタン 選択されているデータ列を削除します。

可変文字列 詳細設定

文字数(最大99文字)、表示色などを設定します。

表示色 プロジェクトで 4系統 が指定されて いる場合、4系統となります。



# 数値データ 詳細設定

総桁数、小数点以下桁数、表示色などを設定 します。

スピンボタン付き 数値データの右側にスピン ボタンを表示します。 スピンボタンの大きさ(横×縦 ドット数)を設定します。 数値データは入力不可でも スピンボタンから入力すること は可能です。

メッセージ出力 スピンボタンが操作された とき、メッセージを出力すること ができます。

# チェックボックス 詳細設定

チェックボックスの大きさ、色を設定します。





#### マーク 詳細設定

マークの大きさ、種類、色を設定します。


## 5-2-4 スクロールバー 詳細設定

レイアウト詳細設定ダイアログ ボックスで [スクロールバー] ボタンを押すと右の画面が表示 されます。

水平、垂直方向スクロールバー の表示方式、制御方式などを 設定します。

	水平方面		重直方向	
(150)	F 5:27%	FF-\$774	F 325%	F 176
在上· 右下种	10	10	10	10
内部	12	14	12	14
日前量	4 표 32	표 10 표	h = hs	王王
キーロ4月	H.SCROLL	* 0 =	V.SOROLL	* 0 =
大きさ	A 20 -	19 2	H 10 21	49

<u>5-3 スプレッドシートの表示</u>

スプレッドシートは親ウィンドウ上で指定された位置に表示されます。

親ウィンドウ

🥭 テス ト	・(スプレッドシート	1)	2004年0	6月29日	(火)	10時27分
ΫλテムNo.	*******	納入先	***********			
作番	***********	盤No.	**********			
コメント	*******************	*********	*********			
_						_
		諳	別コード HYO_RECT			
登録						閉じる

スプレッドシートを表示するためにはアプリケーションプログラム側で次のようなロジックの記述が必要です。 この処理は 親ウィンドウ側 に記述します。

例 1

<pre>void CSpreadSheet1 :: PtrnlnitFunc ()</pre>	// ウィンドウの初期化
{ PTRNDATA Ptrn = {32, 180}; PtrnDispGrid(1, "スプレッドシート1", Point);	// 表示開始位置 // スプレッドシート表示

例1 ではスプレッドシートの左上座標を指定して表示します。

スプレッドシー	トの大きさは次	のよ	うになります。		
縦	(1行の高さ)	<b>x</b> (	(データ行数)	)	
横	(1桁の幅)	×	(表示桁数)		
			データ行数、	、表示桁数はレイアウト詳細設定ダイアログボックス	ζ
			で設定します	す。	

例1 の場合、上図で矩形領域を準備する必要はありませんが、スプレッドシートの表示開始位置を 矩形領域の左上座標から取得することもできます。 例 2

```
// ウィンドウの初期化
void CSpreadSheet2 :: PtrnInitFunc ()
{
  PTRNDATA Ptrn;
                                          // 矩形領域表示情報
  PtrnReadData(HYO_RECT, PTRN_RECT, &Ptrn);
  PtrnDispGrid(1, "スプレッドシート2", Ptrn.Rect);
例2 ではスプレッドシートの左上座標を指定して表示します。
  スプレッドシートの大きさは次のようになります。
     縦
           (1行の高さ) × (データ行数)
     横
           (1桁の幅) × (表示可能桁数)
                      表示可能桁数 = (矩形領域の幅) / (1桁の幅)
                                  (余りは切上げ)
  この方式では表の大きさは指定された矩形領域に収まるように自動的に調整されます。
```

(1桁分大きくなる場合があります。)

#### 5-4 表示データの設定

スプレッドシートに表示するデータの設定方式には、一括と個別の2種類の方式が準備されています。

#### <u>5-4-1 一括設定方式</u>

ー括設定方式では、スプレッドシートの表示開始前に 表示するデータをすべて共有データ領域に設定して おきます。

1 (11) (11) (11) (11) (11) (11) (11) (1	크_25
歳別ロード 150	
(道主会 2)	<b>罪迫表示负 1</b>
AGC 2	新線線
表示データ設定方式	○ 一括 ○ 個別

共有データ領域	琙	
	表	示対象データ

上記の例では 識別コード として 150個 確保され ます。 表示可能なデータ列は 150行以下 となり ます。

アプリケーションプログラムからは次のプログラムで実際に表示する行数を指定することができます。

void CHaisenhyo1 :: PtrnInitFunc ()	// ウィンドウの初期化
{ GridSetScroll(100, 1);	
}	

上記の場合、データ行数は 100行 となります。また、2番目の引数に 1 が指定されているのでスプレッドシート表示後、ただちにキーボード入力が開始されます。

データ行数を設定しなければ、レイアウト詳細設定画面で指定された 識別コード定義数 (上記の場合 150) が表示対象データ行数となります。

識別コード定義数よりも大きい値を指定しようとするとエラーとなり、ログへエラーメッセージが 出力されます。 スプレッドシートは識別コード定義数をデータ行数に設定して処理を行います。



識別コード定義数に次の値が設定されます。

(識別コード定義数) = (データ列数) (上部 / 下部固定行数)

アプリケーションプログラムからは 次のプログラムで実際に表示する行数を指定します。

```
void CHaisenhyo1 :: PtrnInitFunc () // ウィンドウの初期化
{
GridSetScroll(100, 1);
}
```

上記の場合、データ行数は 100行 となります。

個別設定方式では垂直方向スクロールバーの変化を検出して表示データを入れ替える必要が あります。

現在表示中のデータを共有データ領域より取り出し表示対象データ領域に保存します。 キーボード入力可能なデータがある場合にのみ必要です。 スクロールバーの表示開始行番号を取得し、1ページ分の表示対象データを共有データ 領域へ設定します。

75

<u>5 - 5 サンプルプログラム</u>

テストプログラムには2種類のスプレッド シートの例が準備されています。

- テスト画面6 スプレッドシート(1) 表示位置は 左上座標 で指定 します。 表示データの設定は 個別方式 で行います。 右側2列をスクロール禁止領域に 設定しています。
- テスト画面7 スプレッドシート(2) 表示位置は 親ウィンドウの矩形 領域で指定します。 表示データの設定は 一括方式 で行います。

オテスト (メニュー)	RIR)	2004368779108121 (34) 12090659
テスト画面1	図形	1200
テスト画面2	文字列・ボタン	
テスト直面3	キー入力	
テスト画面4	グラフ	
テスト画面5	マウス操作	the second
テスト裏面6	スプレッドシート (	1)
テスト直面7	スプレッドシート ()	2)
117 - PAL	and the second second	AT

- ソースプログラムは haisenhyo.h / haisenhyo.cpp に格納されています。
- スプレッドシートを処理するために次のようは専用の関数が準備されています。

No.	クラス名	関数名	処理内容	備考
1	CPtrnDisp	PtrnDispGrid	スプレッドシートの表示	親ウィンドウで実行
2		GeidSetScroll	実際に表示する行数の設定、	スプレッドシートを
			キー入力の自動開始	処理する子ウィンドウ
3		GridGetScroll	表示開始行番号の取得	で実行
4		GridReadParm	スプレッドシートの表示処理パラメータ	
			を取得 (下記参照)	
5		GridReadData	スプレッドシート データ列設定情報の	
			取得	

キーボード入力開始 / 強制終了 などは通常のレイアウト処理関数を使用することができます。

関数 GridReadOarm スプレッドシートの表示処理パラメータを取得

レイアウト設定画面で設定した内容などを取得することができます。

Parm = GridReadParm (data);

int	Parm;	取得したパラメータ情報
int	data;	取得したい情報

引数 data には次の定数を指定します。 (ヘッダーファイル Ptrndata.h に定義されています。)

No.	定数	取得できるパラメータ情報	備考
1	MODE	表示データ設定方式	0:一括 / 1:個別
1	HEIGHT	1行の高さ	ドット数
2	WIDTH	1桁の幅	ドット数

No.	定数	取得できるパラメータ情報	備考
3	DATALINE	データ行数	固定行を含む
4	FIXLINE1	上部固定行数	
5	FIXLINE2	下部固定行数	
6	DATACULM	データ列数	固定列を含む
7	FIXCULE1	左側固定列数	
8	FIXCULM2	右側固定列数	
9	TOTALSIZE	総桁数	
10	DISPLINE	表示データ行数	固定行を含まない
11	DISPSIZE	表示桁数	固定列を含む
12	DATACOLOR	未設定データ領域 背景色	
13	FIXCOLOR1	上部固定行 背景色	
14	FIXCOLOR2	下部固定行 背景色	
15	CLEARCOLOR	消去色番号	
16	GRIDCOLOR	罫線表示色番号	
17	BODERCOLOR	外枠表示色番号	
18	GRIDWIDTH	罫線 線幅	
19	BODERWIDTH	外枠 線幅	
20	DATACOUNT	表示対象データ数	
21	FONT	文字フォント番号	
22	HSCROLL	水平スクロールバー	0:表示 / 1:非表示
23	VSCROLL	垂直スクロールバー	0:表示 / 1:非表示

### <u>5-5-1 テスト画面6 スプレッドシート(1)</u>

```
親ウィンドウ側
```

/

画面を開く時、閉じる時、マウス入力時の処理を記述します。

void CSpreadSheet1 :: Pt	nInitFunc ()	//	ウィンドウの初期化
CFileRead HaisenHyo(F HaisenHyo.ReadBlock(&H	rojPath, "Haisenhyo"); aisenHedr, sizeof(HAISENHEDR));	 	配線表ファイル読込み ヘッダー部
if (HaisenData != NULL) delete [] HaisenData; HaisenData = new HAISENDATA [HaisenHedr.datacount]; for (int i=0; i <haisenhedr.datacount; ++i)<br="">HaisenHyo.ReadBlock(&amp;HaisenData[i], sizeof(HAISENDAT</haisenhedr.datacount;>		// // );	作業領域破棄 データ部
Set_sData(SYSTEMNO, Set_sData(NOUNYUUSAKI, Set_sData(SAKUBAN, Set_sData(BANNO, Set_sData(COMMENT,	HaisenHedr.systemno); HaisenHedr.nousaki ); HaisenHedr.sakuban ); HaisenHedr.banno ); HaisenHedr.comment );	//	初期表示
POINT Point = {32, 18 PtrnDispGrid(1, "スプ  }	0}; ィッドシート1", Point);	//	表示開始位置

配線表データが格納されたファイルを開きデータを読込みます。 共通表示項目を共有データ領域に設定します。

ウインドウ左上座標を指定してスプレッドシートを子ウィンドウとして表示します。

```
void CSpreadSheet1 :: PtrnExitFunc() // ウィンドウの終了処理
{
    if (HaisenData != NULL) delete [] HaisenData; // 作業領域破棄
    HaisenData = NULL;
}
```

画面と開く時に確保していた領域を開放します。

<pre>void CSpreadSheet1 :: PtrnMesgFunc (int </pre>	code) //	マウス操作による通知
if (code == TOUROKU) { Istrcpy(HaisenHedr.systemno, (LP Istrcpy(HaisenHedr.nousaki, (LP Istrcpy(HaisenHedr.sakuban, (LP Istrcpy(HaisenHedr.banno, (LP Istrcpy(HaisenHedr.comment, (LP	// CTSTR)Get_sData(SYSTEMNO CTSTR)Get_sData(NOUNYUUSA CTSTR)Get_sData(SAKUBAN CTSTR)Get_sData(BANNO CTSTR)Get_sData(COMMENT	登録 ボタン )); // 入力結果の取得 (I)); )); )); ));
CFileWrite HaisenHyo(ProjPath, HaisenHyo.WriteBlock(&HaisenHedr for (int i=0; i <haisenhedr.datac HaisenHyo.WriteBlock(&amp;Haise } }</haisenhedr.datac 	"Haisenhyo"); // , sizeof(HAISENHEDR)); ount; ++i) nData[i], sizeof(HAISENDA <sup>-</sup>	配線表ファイル書込み TA));

登録 ボタンを押された時の処理を記述します。 共通表示項目を共有データ領域より読込みます。 配線表データをディスクファイルに保存します。 スプレッドシート(子ウィンドウ側) 画面を開く時、マウス入力時、キーボード入力終了時の処理を記述します。

void CHaisenhyo1 :: PtrnInitFunc () {	// ウィンドウの初期化
DispDataSet(0); GridSetScroll(HaisenHedr.datacount, 1);	// 先頭ページ 表示データ設定 // 表示対象データ数設定
DispDataSum();	// 合計値計算}

先頭ページの表示データを設定します。 引数で表示開始行番号を指定します。 スプレッドシートの表示対象データ数を設定します。 2番目の引数に 1 を指定 するとスプレッドシート先頭部の入力可能セルよりキーボード入力処理を開始します。 線長データの合計値を計算します。

void CHaisenhyo1 :: PtrnMesgFunc (int code)	// マウス操作による通知
DispDataGet();	// 表示データ取得
<pre>int cont = GridReadParm(DISPLINE); if (code == V_SCROLL)     DispDataSet(GridGetScroll());</pre>	// 表示行数 // スクロールバー(縦)
<pre>else if ((code &gt;= HY0_DATA15)     &amp;&amp; (code &lt; (HY0_DATA15+cont))) DispDataSum(); }</pre>	// スピンボタン // 合計値計算

表示中の情報を共有データ領域から配線表データ領域へ読込みます。 スプレッドシート1ページ当たりの表示行数を取得します。

垂直方向スクロールバーが操作された時、スクロールバーの現在位置情報(表示開始 行番号)を引数に設定してスプレッドシート1ページ分の表示データを共有データ領域に 設定します。

線長データ部のスピンボタンが操作された時、合計値の再計算を行います。

```
BOOL CHaisenhyo1 :: PtrnKeyFunc (int code) // キー入力終了処理
{
DispDataGet(); // 表示データ取得
int cont = GridReadParm(DISPLINE); // 表示行数
if ((code >= HYO_DATA15)
&& (code < (HYO_DATA15+cont))) DispDataSum();
return TRUE;
}
```

キーボード入力終了時、表示中データを共有データ領域より配線表データ領域へ読込み ます。 次項の線長合計値再計算、親ウィンドウで 登録 ボタンがいつ押されても良い ように、表示中データに変化があれば読込みを行います。 線長データに対しキーボード入力が行われた時、合計値の再計算を行います。 TRUE を返すとパッケージプログラム SimpleMaster 側でキーボード入力の継続を行います。 スプレッドシート(子ウィンドウ)側では、表示データの設定 / 取得、線長合計値算出のため関数 を定義しています。

```
void CHaisenhyo1 :: DispDataSet (int dispno) // 表示データ設定
{
    DispDtno = dispno;
    int cont = GridReadParm(DISPLINE); // 表示行数
    for (int i=0; i<cont; ++i) {
        int j = DispDtno + i;
        Set_iData(LINEN0 + i, DispDtno+i+1);
        Set_sData(HYO_DATA1 + i, HaisenData[j].senban);
        (一部省略)
    }
}</pre>
```

引数で渡された表示開始行番号を保存します。 スプレッドシート1ページ当たりの表示行数を取得し、 表示開始行番号より1ページ分の配線表データを共有データ領域に設定します。

```
void CHaisenhyo1 :: DispDataGet () // 表示データ取得
{
    int cont = GridReadParm(DISPLINE); // 表示行数
    for (int i=0; i<cont; ++i) {
        int j = DispDtno + i;
        HaisenData[j].length = Get_iData(HY0_DATA15+i);
        Istrcpy(HaisenData[j].senban, (LPCTSTR)Get_sData(HY0_DATA1 +i));
        Istrcpy(HaisenData[j].senkei, (LPCTSTR)Get_sData(HY0_DATA2 +i));
        Istrcpy(HaisenData[j].sensyu, (LPCTSTR)Get_sData(HY0_DATA3 +i));
        (一部省略)
    }
}</pre>
```

スプレッドシート1ページ当たりの表示行数を取得し、 表示開始行番号より1ページ分の配線表データを共有データ領域より取得します。

```
void CHaisenhyo1 :: DispDataSum () // 線長データ 合計値再計算
{
    int goukei = 0;
    for (int i=0; i<HaisenHedr.datacount; ++i) goukei += HaisenData[i].length;
    Set_iData(GOUKE1, goukei);
    }
```

配線表 線長データの合計値を計算し、共有データ領域へ設定します。

### <u>5-5-2 テスト画面7 スプレッドシート(2)</u>

```
親ウィンドウ側
```

画面を開く時、閉じる時、マウス入力時の処理を記述します。

voi	d CSpreadSheet2 :: PtrnInitFunc ()	// ウィンドウの初期化	
١	CFileRead HaisenHyo(ProjPath, "Haisenhyo"); HaisenHyo.ReadBlock(&HaisenHedr, sizeof(HAISENHEDR));	// 配線表ファイル読込み // ヘッダー部	
	<pre>if (HaisenData != NULL) delete [] HaisenData; HaisenData = new HAISENDATA [HaisenHedr.datacount]; for (int i=0; i<haisenhedr.datacount; ++i)="" {<br="">HaisenHyo.ReadBlock(&amp;HaisenData[i], sizeof(HAISENDATA</haisenhedr.datacount;></pre>	// 作業領域破棄 // データ部 A));	
	Set_iData(LINENO +i, i+1); Set_sData(HYO_DATA1 +i, HaisenData[i].senban); Set_sData(HYO_DATA2 +i, HaisenData[i].senkei); (一部省略) }	// 表データ設定 // (一括方式)	
	Set_sData(SYSTEMNO, Set_sData(NOUNYUUSAKI, HaisenHedr.nousaki);HaisenHedr.nousaki); HaisenHedr.sakuban);Set_sData(SAKUBAN, Set_sData(BANNO, HaisenHedr.banno);HaisenHedr.comment); HaisenHedr.comment);ShowWindow(SW_SHOW);HaisenHedr.comment);	// 初期表示	
}	PTRNDATA Ptrn; PtrnReadData(HYO_RECT, PTRN_RECT, &Ptrn); PtrnDispGrid(1, "スプレッドシート2", Ptrn.Rect);	// 矩形領域表示情報	

配線表データが格納されたファイルを開きデータを読込みます。 配線表データを読込む領域を確保します。 配線表データを読込みながら、共有データ領域に設定します。 共通表示項目を共有データ領域に設定します。 親ウインドウ上の矩形領域の座標を指定してスプレッドシートを子ウィンドウとして表示します。

```
void CSpreadSheet2 :: PtrnExitFunc () // ウィンドウの終了処理
{
    if (HaisenData != NULL) delete [] HaisenData; // 作業領域破棄
    HaisenData = NULL;
}
```

画面と開く時に確保していた領域を開放します。

```
void CSpreadSheet2 :: PtrnMesgFunc (int code)
                                                                   // マウス操作による通知
{
   if (code == TOUROKU) {
                                                                   // 登録 ボタン
        \verb|Istrcpy(HaisenHedr.systemno, (LPCTSTR)Get\_sData(SYSTEMNO)| \\
                                                                  ));
        Istrcpy(HaisenHedr.nousaki, (LPCTSTR)Get_sData(NOUNYUUSAKI));
        \verb|Istrcpy(HaisenHedr.sakuban, (LPCTSTR)Get\_sData(SAKUBAN)|
                                                                  ));
                                    (LPCTSTR)Get_sData(BANNO
        lstrcpy(HaisenHedr.banno,
                                                                  )):
        Istrcpy(HaisenHedr.comment, (LPCTSTR)Get_sData(COMMENT
                                                                  ));
       CFileWrite HaisenHyo(ProjPath, "Haisenhyo");
                                                                   // 配線表ファイル書込み
       HaisenHyo.WriteBlock(&HaisenHedr, sizeof(HAISENHEDR));
        for (int i=0; i<HaisenHedr.datacount; ++i) {</pre>
            HaisenData[i].length = Get_iData(HY0_DATA15+i);
                                                                   // 表データ取得
             Istrcpy(HaisenData[i].senban, (LPCTSTR)Get_sData(HYO_DATA1 +i)); // (一括方式)
             Istrcpy(HaisenData[i].senkei, (LPCTSTR)Get_sData(HY0_DATA2 +i));
             Istrcpy(HaisenData[i].sensyu, (LPCTSTR)Get_sData(HYO_DATA3 +i));
             (一部省略)
             if (Get_iData(HYO_DATA4+i) == 0) Istrcpy(HaisenData[i].kyoku, "0");
                                             Istrcpy(HaisenData[i].kyoku, "1");
            else
            HaisenHyo.WriteBlock(&HaisenData[i], sizeof(HAISENDATA));
       }
  }
```

登録 ボタンを押された時の処理を記述します。 共通表示項目を共有データ領域より読込みます。 配線表ファイルをオープンし 共有データ領域より配線表データを読込みながら ファイルへ書込みます。

#### スプレッドシート(子ウィンドウ側)

画面を開く時、マウス入力時、キーボード入力終了時の処理を記述します。

void CHaisenhyo1 :: PtrnInitFunc ()	// ウィンドウの初期化
<pre>GridSetScroll(HaisenHedr.datacount, 0); DispDataSum();</pre>	// 表示対象データ数設定 // 合計値計算
}	

スプレッドシートの表示対象データ数を設定します。2番目の引数に 0 を指定 するとキーボード入力処理は開始されません。 線長データの合計値を計算します。



線長データ部のスピンボタンが操作された時、合計値の再計算を行います。

```
BOOL CHaisenhyo1 :: PtrnKeyFunc (int code) // キー入力終了処理
{
    if ((code >= HYO_DATA15) // 線長入力
    && (code < (HYO_DATA15+ HaisenHedr.datacount))) DispDataSum(); // 合計値計算
}
```

線長データに対しキーボード入力が行われた時、合計値の再計算を行います。

スプレッドシート(子ウィンドウ)側では、線長合計値算出のため関数を定義しています。

```
void CHaisenhyo1 :: DispDataSum () // 線長データ 合計値再計算
{
    int goukei = 0;
    for (int i=0; i<HaisenHedr.datacount; ++i) goukei += Get_iData(HYO_DATA15+i);
    Set_iData(GOUKEI, goukei);
}
```

共有データ領域の線長データの合計値を計算し、共有データ領域へ設定します。

# SimpleMaster へのお問合せは下記へお願いします。 株式会社 住所 〒812-0013 福岡市博多区博多駅東2丁目9番25号 TEL 092-431-0800 FAX 092-431-2018 URL http://www.fuji-technica.jp/ E-Mail master@fuji-technica.jp