



[プライマネージ]

PRIMANAGE

CSS Open Package Series for Enterprise Resource Planning

*PRIMANAGE
MEISTER*

ソフトウェア部品システムの印刷

Central Soft Service Co.,Ltd.

株式会社セントラルソフトサービス

目 次

1. 概 要	3
2 ソフトウェア部品システムの印刷の仕組み	4
2.1 印刷を行う部品の動作	4
2.2 印刷に特有な作業	6
2.3 印刷ファイルの命名規則と取得方法	7
3 ソフトウェア部品システム の 印刷の実行過程	8
3.1 印刷出力の形態(用紙の種類)	8
3.2 “ストックフォーム印刷”と“A4印刷”の動作過程.....	9
3.2.1 印刷形態の指定	9
3.2.2 印刷出力の形態の指定(具体例).....	9
3.2.3 印刷ファイルのコピーとプリンタへの出力.....	11
3.3 “帳票印刷”の動作過程	13
3.3.1 帳票印刷と他の印刷形態(ストックフォーム印刷 と A4 印刷)との関係.....	13
3.3.2 印刷情報の生成とプリンタへの供給	13
3.3.3 “帳票印刷”の動作過程	14
3.4 印刷の要約	15
4. 印刷動作のトラブルシュート.....	16

1. 概要

全社の業務を処理するソフトウェア・システムでは、プリンタへの出力を行うこと必要になりますが、その出力の形態を分類すると、おおよそ次の3種類のいずれかになります。

- (1) スtockフォームへの印刷 (一覧表の出力で用いられる)
- (2) A4用紙への印刷 (最も一般的な出力形態。例えば、受注報告書)
- (3) 特定用紙への印刷^{〔*1〕} (源泉徴収票のように、印字位置と寸法が決まっている用紙への印刷)

ソフトウェア部品システムでは、部品から、この(1)～(3)の印刷を行うことができます。

〔*1〕ソフトウェア部品システムでは、この、「特定用紙への印刷」を、“帳票印刷”と呼んでいます。

このドキュメントでは、上記の(1)～(3)の印刷の種類について説明して、それぞれの印刷を行うために、部品のプログラムでは、何をすればよいのか、を記します。

2 ソフトウェア部品システムの印刷の仕組み

ソフトウェア部品システム の印刷方式は、独特の“仕組み”を採用しており、この仕組みについての知識が、印刷を行う部品を作成するときに必要なになります。この仕組みの中でも、一番の特徴は、

ソフトウェア部品システムの、標準のシステム^[*2]では、

- ◆ 印刷すべきデータは、すべて、第 2 相〔業務処理プログラム(COBOL)側〕で作成^[*3]し、
- ◆ 印刷の作業そのものは、すべて、第 1 相〔GUI 側〕から^[*4]行っている、
という点にあります。

^[*2] ユーザー殿が独自に作成された部品では、ここに記した方式以外の方式で、印刷が行われているかもしれません。ここでは、ソフトウェア部品システムの、標準のシステムでの印刷について記しています。
なお、以下では、“ソフトウェア部品システムの”を省略して、単に“標準システム”と呼びます。

^[*3] 第 2 相では、印刷すべきデータを集めるだけでなく、そのデータを、印刷フォーマット に合わせて配置する作業も行います。この配置の作業を行うときに、COBOL の“レポート機能”を用います。

^[*4] これにより、クライアント側で、印刷の制御を行うことができます。(印刷を行うプリンタは、ネットワーク・プリンタでも、クライアントに ローカルな プリンタ でも、差し支えありません。)

2.1 印刷を行う部品の動作

(上記のように、)印刷の作業では、データを集める作業を行う相と、印刷の実行を行う相が異なっています。(第 2 相と第 1 相。)

相の間で、どのような(作業の)連携を行っているか、を説明するために、部品が起動されてから、その部品の動作が終了するまでの過程(シーケンス)を、(印刷に関連する部分だけ抜き出して)具体的に追ってみることにします。

図 1(次ページ)をご参照下さい。

- ① クライアント側(第 1 相)から部品を起動します。
- ② クライアント側からのリクエストにより、第 2 相の業務処理プログラム が起動されます。^[*5]

^[*5] 業務処理プログラムの中では、印刷の機能が占める役割は、ごく一部ですが、ここでは印刷の動作を説明することに重点を置いているので、印刷のために第 2 相のプログラムが呼ばれたような記述になっています。

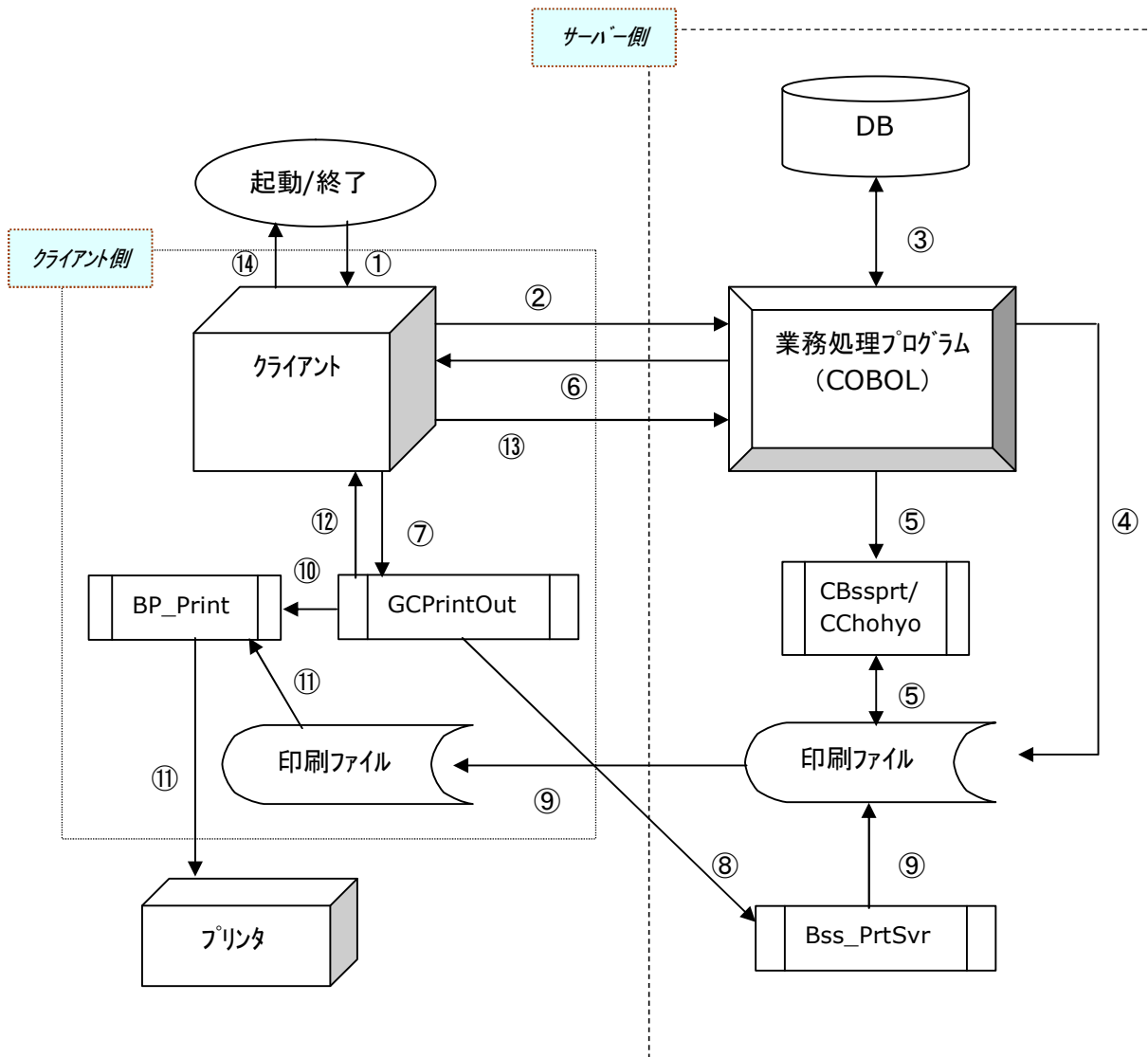


図 1 印刷動作の詳細

- ③ 業務処理プログラムは、DB にアクセスして、印刷の対象となるデータを集めます。
 - ④ データが集まったら、(その部品の)印刷フォーマットに従って、各データを順次、配置します。^[*6]
配置を終えた部分を、順次、ファイルに書き込みます。^[*7]
ファイルに書き込まれた内容は、この段階で既に、印刷のイメージ通りになっています。(つまり、ヘッダやタイトルが書き込まれ、ページが割り振られ、縦に揃うべきところは、揃っています。)^[*8]
- [*6] COBOL のレポート機能で、この作業を行います。
- [*7] 業務処理プログラム が、このファイルを作成します。作成したファイルを、サーバー側にインストールされているソフトウェア部品システムの、ホームディレクトリの下にある、wrk ディレクトリ(%BSS_HOME%\wrk)の下に保存します。作成の時には、そのファイル名を、wrk ディレクトリの下に既に存在している他のファイルとは異なるようにします。
業務処理プログラム は、終了するときに、このファイルを削除します。
なお、このファイルの名前の特徴については、後述の 2.3 項を参照してください。
- [*8] 3.3 項で述べますが、“帳票印刷” の場合は、まだ印刷時のイメージになっていません。
帳票印刷では、BP_Print という実行モジュールが、プリンタに出力を行う直前に、印刷イメージを作成します。また、改ページのコードはまだ入っていません。

- ⑤ 項目④ で作成した、印刷用のファイルに、改ページコードを挿入します。
 スtockフォーム印刷や、帳票印刷の場合は、用紙が連続帳票になっているので、次の印刷位置の頭出しをするために、プリンタに改ページコード(^L)を送る必要がありますが、④ で作成したファイルには、このコードが入っていないため、ここで挿入します。(A4 印刷でも、念のために挿入します。)
- なお、stockフォームとA4 では、一般的に66行で改ページして差し支えありませんが、帳票印刷の場合、帳票によっては、用紙の縦の長さが短いため、用紙1枚分の位置で改ページをする必要があります。
 関数 CBssprt を用いると、66行で改ページを挿入します。これに対して、関数 CChohyo は、パラメータで指定した行数の位置に、改ページコードを入れることができます。
- この2つの関数は、3.1項の“印刷形態”に対応させて、使い分けます。(stockフォーム印刷、A4印刷、では、CBssprt を実行し、帳票印刷では、CChohyo を実行します。)
- ⑥ サーバ側で、印刷ファイルの準備が整うと、クライアント側に制御を戻します
 ⑦ クライアント側は、GCPrintOut という関数を呼び出します。
 ⑧ GCPrintOut は、印刷のサーバ・モジュール: Bss_PrtSvr.exe を起動します。
 ⑨ Bss_PrtSvr により、クライアント側は、サーバ側にある印刷ファイルを、クライアント側にコピーすることができます。
 クライアント側では、コピーした印刷ファイルを、クライアント側の、ソフトウェア部品システム のホームディレクトリの下にある、wrk ディレクトリ (%BSS_HOME%¥wrk) の下に保存します。このとき、クライアント側では、このファイルに独自の名前を与えます。
- ⑩ 印刷ファイルを、サーバ側から受け取ると、GCPrintOut は、BP_Print という名前の印刷モジュール を起動します。
 ⑪ BP_Print は、クライアント側にある印刷ファイルを開き、内容を読み込んで、プリンタに送信します。
 プリンタ(正確には Windows API を介して、プリンタドライバ)と直接やりとりを行うのは、この BP_Print であり、他のモジュール が、このやりとりをすることはありません。
 (つまり、実際に印刷を制御する機能は、BP_Print に集約されています。)
- ⑫ 印刷が終わると、制御は GCPrintOut に戻り、ここで GCPrintOut も動作を終了します。
 ⑬ クライアント側は、サーバ側に終了のメッセージを送ります。
 (サーバ側は、印刷の一時ファイルを削除して、終了します。)
- ⑭ クライアント側も、印刷の一時ファイルを削除して、終了します。
 (なお、Bss_PrtSvr は、自動的に終了します。)

2.2 印刷に特有な作業

上図の要約として、印刷に特有な作業を取り出すと以下のようになります。

- (a) サーバ側
- ◆ (COBOL の)レポート機能を用いた印刷イメージの作成 ^[*9] (④)
 - ◆ 改行コードの差し込み (⑤)
- (b) クライアント側
- ◆ GCPrintOut の呼び出し (⑧)
 (呼び出したあと、印刷動作は自動的に実行されます。印刷動作の実行後、GCPrintOut に制御が戻されます。)

[*9] ソフトウェア部品システム は Unix システム でも稼働しますので、この Unix システム に接続されている プリンタ に対して印刷出力を行うことがあります。
 この場合は、プリンタとしては、ラインプリンタ あるいは、その同等品が使われていることがあり、しかも インテリジェンス をあまり期待できない場合があります。そこで、印刷を、「文字コード」と「(7bit の ASCII テーブルに定義されている)プリンタ の制御コード」で実行するようになっています。

2.3 印刷ファイルの命名規則と取得方法

サーバー側で作成した印刷ファイルは、保存を行うときに名前が必要になります。この名前は、印刷の都度異なるようにして、仮に以前の印刷で生成された印刷ファイルが(削除されずに)残っていても、上書きしないようにします。

(ソフトウェア部品システムの) 標準システム では、次のようにして名前を生成しています。

- (1) 名前の生成は、業務処理プログラムの中で行います。
- (2) 名前の先頭は、その部品の ID に続いて、アンダースコアを 1 つ置き、その後ろに文字列 PRT をつけたものとします。

例: 「担当者テーブル一覧」(SYS9840) の場合、この文字列は、“SYS9840_PRT” となります。

- (3) 印刷を実行する都度、項目(2)の文字列の後ろに、8 桁の数字文字列を付けて、その文字列をファイル名とします。(拡張子はつけません。)

例: 「担当者テーブル一覧」(SYS9840) の場合で、(印刷の都度、ファイル名は異なりますが、)例えば、“SYS9840_PRT00000025” となります。

- (4) 標準システム では、この数字文字列を含んだ新しいファイル名を取得するために、CSseqwk を用いています。^[*10]

この関数の第 1 パラメータ に、項目(2)の文字列をセットして呼び出すと、第 2 パラメータに、(先頭に項目(2)の文字列を持ち、重複のない数字文字列が後ろについた、) ファイル名がセットされて戻されてきます。

(CSseqwk は、シーケンシャルなワークファイル名を自動採番より取得するので、戻されてくる文字列は、それ以前に生成された文字列とは、一致することがありません。)

^[*10] 開発をする人は、この関数を用いれば、重複のない新しい名前を取得するための ルーチン を作成する必要はありません。

3 ソフトウェア部品システムの印刷の実行過程

3.1 印刷出力の形態(用紙の種類)

「ソフトウェア部品システムの印刷の仕組み」の特徴をまとめると、次のようになります。

- ◆ 印刷する内容を、第 2 相(業務処理プログラム)で作成します。
- ◆ 印刷内容の作成には、COBOL のレポート機能を用います。
- ◆ この印刷内容は、サーバー側に、ファイル^[*11]として保存されます。

[*11] この ファイル を、以降は“印刷ファイル”と呼びます。

- ◆ このファイルの内容は、(7bit の ASCII テーブルに定義されている プリンタ の制御コード を除き) すべて、キャラクタ(文字コード)で構成されています。(日本語は、Shift-JIS コード を用いています。)
- ◆ 印刷は第 1 相で行われます。

印刷ファイルの内容は、基本的には、

「用紙に印刷される“出力イメージ”が構成されていて、あとは単に用紙の上に、そのイメージを現すだけ」の状態になっています。

(既に [*8] で述べていますが、)帳票印刷の場合は例外で、まだイメージ通りにはなってはいません。しかし、イメージを構成するための情報は、印刷ファイルの中に、全て含まれています。(3.3 項)

次に、この“印刷ファイル”の内容を、プリンタに打ち出しますが、部品の目的に応じて、3 種類の形態の用紙のいずれかに印刷を行います。

この形態とは、「ストックフォーム」、「A4」、および「特定の用紙(複数種類あります)」、で、それぞれの形態の用紙への印刷を、“ストックフォーム印刷”、“A4 印刷”、“帳票印刷”、と呼んでいます。^[*12]

[*12] 以降、この 3 つの印刷の種類を、まとめて“印刷形態”と呼びます。

[印刷形態の相互関連]

部品を開発する立場から、“印刷形態”を分類すると、次のようになります。

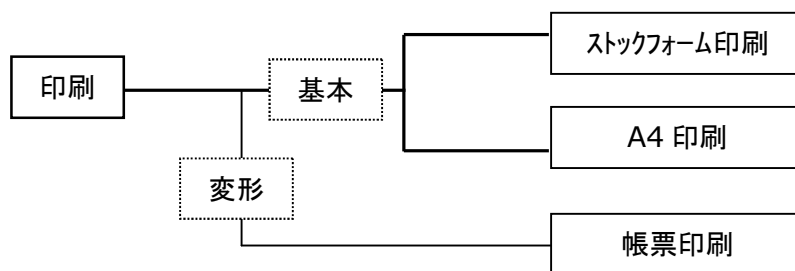


図 2 印刷形態の関連

- (1) “ストックフォーム印刷”と“A4 印刷”が基本になっている。
- (2) “帳票印刷”は、“ストックフォーム印刷”と“A4 印刷”の変形バージョンである。

なお、印刷ファイルの内容を見ても、“ストックフォーム印刷を実行しようとしているのか”、あるいは、“A4 印刷を実行しようとしているのか”、区別ができません。つまり、“ストックフォーム印刷”と“A4 印刷”の違いは、印刷ファイルとは別のところで決められています。

次項で、印刷の動作過程を説明しますが、“ストックフォーム印刷”、“A4 印刷”について、まず説明します。

3.2 “ストックフォーム印刷”と“A4印刷”の動作過程

“ストックフォーム印刷”と“A4印刷”の場合は、“印刷ファイル”の内容は、印刷されるイメージそのものになっていますので、印刷は、このファイルの内容を読み込んで、そのままプリンタに転送する動作をします。

その印刷ファイルで、“ストックフォーム印刷”として出力するのか、あるいは“A4印刷”として出力するのかは、業務処理プログラムで指定します。

3.2.1 印刷形態の指定

具体的には、業務処理プログラムが、印刷ファイル^[*13]を作成して、クライアント側に制御を戻すときに、制御を戻す関数(SSMProcPut)を呼ぶ前に、そのパラメータに、印刷ファイルの名前と、印刷出力の形態をセットします。

印刷出力の形態は、業務処理プログラムに、PRINT_INFO.cpyを読み込み、このコピー句に定義されている、2つの変数^[*14](PS-STOCKFORM または、PS-AYON-TATE)のうちのいずれかを指定します。

[*13] このファイルは、改行コードが埋め込まれたあとの、最終形態となった印刷ファイルです。

[*14] PRINT_INFO.cpyでは、この2つの変数に文字列の値を与えています。
値はそれぞれ、“stockForm”(ストックフォーム印刷)、“a4Portlate”(A4印刷)となっています。

クライアント側は、印刷データをサーバー側で集めるために、SCMProcStartを呼び、これによって、サーバー側の業務処理プログラムが起動されます。サーバー側の業務処理プログラムがデータの収集(と演算)を実行している間、クライアント側は、この状態のまま待機しています。

サーバー側から応答があり、制御が戻されてきたとき、SCMProcStartの最後のパラメータに、印刷ファイルの名前と、印刷出力の形態がセットされています。

SCMProcStartの最後のパラメータは、Print_Info構造体へのポインタですが、この構造体の、styleフィールド(charの配列)に、印刷出力の形態がセットされています。(当然、“stockForm”、“a4Portlate”のいずれかになっています。)

3.2.2 印刷出力の形態の指定(具体例)

上記の説明を、対応する実際のコードで示すと、以下のようになります。
(なお、コードはSYS9840「担当者テーブル一覧」から引用しています。)

(1) 印刷データの収集

クライアント側では、印刷データの収集を行うために、サーバー側の業務処理プログラム(MSYS9840.cbl)を、次の関数で起動します。

```
Rtn_m_code = SCMProcStart ( &Prn_db_id, &Proc_inf_tbl,  
                           (char *)&SendCob, (char *)&RecvCob );
```

図3 SYS9840のクライアント側のコード(第2相への依頼)

4番目のパラメータは、(呼び出しのときはキャストされていますが、)次のように定義されています。

```
static Print_Info RecvCob; /* 受信データ */
```

図4 SYS9840のRecvCobの定義(クライアント側のコード)

このPrint_Infoの型は、p_com.hで次のように定義されています。

```

typedef struct {
    char fname[PFILE_LEN];          /* 印刷用一時ファイル名 */
    char style[PSTYLE_LEN];        /* 用紙スタイル */
} Print_Info;                      /* 合計 178 */

```

図 5 型 Print_Info の定義

- (2) 業務処理プログラムは、印刷を行うときは、PRINT_INFO.cpy を読み込みます。このコピー句は次のようになっています。(部分)

```

01 PRINT-INFO.
*                               * 印刷用一時ファイル名
05 PRINT-FILE PIC X(128).
*                               * 用紙スタイル
05 PRINT-STYLE PIC X(50).
*                               * 印刷スタイル(ストックフォーム)
01 PS-STOCKFORM PIC X(50) VALUE "stockForm".
*                               * 印刷スタイル(A 4 縦)
01 PS-AYON-TATE PIC X(50) VALUE "a4Portlate".

```

図 6 第 2 相で、印刷に使用するコピー句 (PRINT_INFO.cpy) (部分)

- (3) 印刷ファイルの作成が終わり、業務処理プログラムが、クライアント側に応答を返す(制御を返す)ときは、次の関数を呼びます。

```

CALL "SSMProcPut" USING SOCKET-ID RTN-STATUS
                        DATA-SIZE PUT-REC.

```

図 7 第 1 相に制御を戻すコード

この呼び出しを行う直前に、PUT-REC に値を次のようにセットします。

```

MOVE PRT-FILE-NAME TO PRINT-FILE
MOVE PS-STOCKFORM TO PRINT-STYLE
*
MOVE PRINT-INFO TO PUT-REC
MOVE 178 TO DATA-SIZE

```

図 8 第 1 相に制御を戻す前のパラメータのセット

この例では、印刷出力の形態として、ストックフォームが指定されています。
(図 6 から、PRINT-INFO が、PRINT-FILE と PRINT-STYLE を合わせた領域を指していることがわかります。この PRINT-INFO の内容が、PUT-REC にコピーされています。)

- (4) 項目(3)の CALL ステートメントが実行されると、図 3 の SCMProcStart に制御が返ります。このとき、SCMProcStart の、RecvCob には、図 8 の、PUT-REC の値がセットされています。

3.2.3 印刷ファイルのコピーとプリンタへの出力

サーバー側にある、印刷ファイルの名前と、印刷出力の形態はわかりましたが、肝心の印刷ファイルは、まだクライアント側には来ていません。

そこで、サーバー側にある印刷ファイルを、クライアント側にコピーする作業を行います。

そのために、クライアント側が行うのは、GCPrintOut 関数を呼び出すことだけです。^[*15]

^[*15] 通常は、ユーザーが印刷を希望しているかどうかを、まず尋ねます。(ユーザーが、間違った部品を起動してしまった場合などのため) ユーザーからの確認を得たあと、GCPrintOut を呼び出します。

GCPrintOut 関数へは、SCMProcStart 関数の 4 番目のパラメータ(RecvCob)のアドレスを渡すだけの、極めて簡素なインタフェースとなっています。

例: Rtn_m_code = GCPrintOut(&RecvCob);

【GCPrintOut 関数】

この関数(GCPrintOut) は、サーバー側にあるファイルをクライアント側にコピーして、プリンタへの印刷を行う作業を、すべて自動的に行います。

GCPrintOut 関数を呼び出してからは、次の動作が実行されます。

- ① Bss_PrtSvr.exe の起動
- ② サーバー側にある印刷ファイルの(クライアント側への)コピー
- ③ 印刷実行関数: BP_Print.exe の起動

Bss_PrtSvr.exe は、サーバー側にあるモジュールで、このモジュールの働きで、サーバー側にある印刷ファイルを、クライアント側にコピーすることができます。

クライアント側では、サーバー側から受け取ったデータを、%BSS_HOME%\wrk の下にファイルとして保存します。このときのファイル名は、クライアント側で独自に作成するもので、wrk ティレクトリの下にあるファイルと重複しない名前が自動的に選ばれます。通常は、数字 1 つのファイル名が選ばれます。(拡張子はありません。)

【印刷実行モジュール BP_Print.exe】

このモジュールは、印刷の実作業を行います。

このモジュールには、印刷の機能の多くが実装されています。例えば、

- (1) プレビュー画面の表示
- (2) プリンタの切り替え
- (3) プリンタへの出力
- (4) 直接印刷の実行
- (5) フォントの自動変換と用紙の向きの自動変換

【印刷形態 と BP_Print.exe】

業務処理プログラムは、印刷ファイルを作成して、制御をクライアント側に戻すときに、印刷の形態^[*16]を指定しましたが、その指定に対応する処理は、BP_Print で行われます。^[*17]

^[*16] 印刷出力の形態(“stockForm”あるいは“a4Portlate”)のことです。

^[*17] この値は、SCMProcStart → GCPrintOut と経由して、BP_Print に、起動時のパラメータとして渡されます。

なお、BP_Print は、実際の印刷を行う前に、プリンタのドライバを通じて、プリンタにセットされている用紙のサイズを取得する作業を行います。

指定された“印刷の形態”に対応して、BP_Print は、次のような動作をします。

(a) 指定が“ストックフォーム印刷”の場合

BP_Print は、印刷形態の指定が、“ストックフォーム印刷”であることを知ると、

- 1) プリンタにセットされている用紙が A4 の場合は、用紙の向きを横向きにします。(=印刷の向きを、Portrait から Landscape に変更します。)
- 2) 用紙の横幅の中に、180 文字(半角)を印刷することができるかどうかを調べます。
(プログラムでは、最大の印字文字数は 176 文字に制限されています。)
- 3) もし、印刷できない場合は、フォントのサイズを 1 ポイント小さくして、2) を繰り返します。
- 4) 180 文字印刷できるフォントのサイズが確定したら、このフォント・サイズで印刷を行うことにします。

これにより、ストックフォーム印刷では、15" × 11" の用紙ばかりでなく、A4 用紙に印刷を行うことができます。A4 用紙に印刷が行われる場合は、横方向に印刷が行われます。(フォントは 15" × 11" の場合と較べると小さくなっています。)

この用紙の向きを変える機能は、業務処理プログラムで、印刷のスタイルに、“PS-STOCKFORM” (= “stockForm”) と指定したときだけ、働きます。

(b) 指定が“A4 印刷”の場合

この場合、BP_Print は、用紙の横幅の中に、96 文字(半角)を印刷できるかどうかを調べます。

もし、印刷できないとわかると、(a) の場合と同様に、96 文字(半角)印刷ができるようになるまで、フォントのサイズを、1 ポイントずつ小さくしていきます。

96 文字印刷できるフォントのサイズが確定したら、このフォント・サイズで印刷を行うことにします。

(c) “帳票印刷”

帳票印刷については、この後(3.3 項)で説明いたしますが、要点を述べると、印刷ファイルには、印字される文字列だけではなく、

- ① 文字列の印字開始位置と、印字のフォントのサイズについての指定、
 - ② 印刷用紙の幅と長さ、の指定、
- が同時に埋め込まれています。

帳票印刷では、このファイルの内容を読み込み、① と ② の“印字の制御情報”と、印字すべき文字列を分離して、“印字の制御情報”に従って印刷を行うものです。

この作業は、BP_Print によって行われます。

(a)、(b)、(c)、の各項で、“印刷を行う”と述べていますが、その“印刷”とは、Windows API を通じてプリンタ・ドライバに対して(印字の位置を含めて)データを送る、ことを指します。

BP_Print が、この“印刷”を行っていますが、実際の印刷動作は、OS の管理下で行われます。

プリンタ・ドライバに 1 ページ分のデータを送り、「1 ページ分のデータを送った」と連絡を行います。それから先の印刷動作の制御は BP_Print の手を離れます。

プリンタ・ドライバに対してデータを送ったあとは、印刷の形態が“ストックフォーム印刷”、“A4 印刷”、“帳票印刷”のいずれであっても、動作は同じになります。

3.3 “帳票印刷”の動作過程

3.3.1 帳票印刷と他の印刷形態（ストックフォーム印刷 と A4 印刷）との関係

“帳票印刷”は、“ストックフォーム印刷”と“A4印刷”の変形バージョンである、と説明いたしましたが（3.1.2項、および図2）、変形バージョンが必要になるのは、

“ストックフォーム印刷”と“A4印刷”では、行単位で印刷する方式で、目的の印刷を行うことができるが、業務処理の出力の中には、この方式では、目的としている印刷ができないものがあるためです。

具体的には、“給与支払報告書（個人別明細書）”、“給与所得の源泉徴収票”のような所定様式に印刷する場合、あるいは、“被保険者報酬月額基礎届・変更届”、“所得税 源泉徴収簿兼賃金台帳”のように一般的に使用されている用紙に印刷する場合です。

このような、印刷を行う用紙が決まっているときの印刷では、文字列の印字位置が決まっているほかに、印刷の結果が所定の欄内に収まるようにする必要があります。

このため、印字位置の制御が必要となるほか、場合によっては、所定の欄内に印刷結果が収まるように、フォントサイズ¹⁸の選択を行うことが必要になります。

このような印刷は、行単位の印刷方式では、実現できません。

ソフトウェア部品システム の印刷では、この印刷用紙が決まっている場合の印刷を行うために、印字位置の制御と、フォントの制御、および印刷用紙サイズ¹⁸の指定^[*18]、を行うことができるようになっています。

（以降、“印字位置の制御を行う情報”と、“フォントの制御を行う情報”、および“印刷用紙のサイズの情報”、を合わせて、“印刷情報”と呼びます。）

[*18] BP_Print で必要としている情報のため。

3.3.2 印刷情報の生成とプリンタへの供給

この印刷情報を生成するのに最も適しているのは、業務処理プログラム であると判断されますが、この情報が実際に使用されるのは、BP_Print です。

印刷情報が、第2相から、第1相のBP_Printに届けられるように、以下のような方法が使われています。

- ① 業務処理プログラムは、データを生成すると、「そのデータ」と、「そのデータの印刷情報（印字されるべき位置とフォントサイズ）」を、対（ペア）にして、印刷ファイルに書き込む。
- ② 印刷情報は、（印刷ファイルに書き込むので）印刷可能な文字コード¹⁹で構成する。（つまり、バイナリコードは用いない。）
- ③ 印刷情報は、*メ*文字の機能を導入して、印刷される文字と識別できるようにする。
- ④ *メ*文字化のために、“~c”（*チルダ*と半角小文字 c）と“~L”（*チルダ*と半角大文字の L）という文字の組み合わせを導入する。それぞれの*メ*文字がカバーする範囲（その範囲は印刷情報を記述している範囲であり、印字の対象外である、という範囲）は、固定長とする。

(a) “~c” は、印字位置の指定を行うもので、“~cxxxxyyyypppFPP” の形式をとる。

xxxx は、印字を開始する位置（横方向の位置）を、用紙左端からの距離によって表したものの。単位は、1/100 インチ。^[*19]（4 バイト）

yyyy は、印字を開始する位置（縦方向の位置）を、用紙上端からの距離によって表したものの。単位は、1/100 インチ。^[*19]（4 バイト）

ppp は、印字する文字の大きさ（フォント・サイズ）を、1/10 ポイントを単位として計り、その値を、数字の文字列として表現したものの。（3 バイト）

F は、文字のピッチフラグを指定するもの。（1 バイト）

PP は、文字のピッチサイズを指定するもの。（2 バイト）

現在、このうちの F と PP の指定は実装されていないため、実際の指定を行うときは、この 2 つのパラメータ の位置には、数字の文字列 “000” を入れる。

印字位置の指定は、(“~c”を含めて)16バイトの固定長とする。

(b) “~L”は、用紙サイズの指定を行うもので、“~Lxxxxyyyy”の形式をとる。

xxxx は、用紙の横幅を指定する数字文字列で、横幅を 1/10 mm 単位^[*19]で表したものの。

yyyy は、用紙の横長さを指定する数字文字列で、横幅を 1/10 mm 単位^[*19]で表したものの。

用紙長の指定は、(~Lを含めて)10バイトの固定長とする。

[*19] 印字位置の指定と、用紙サイズの指定の単位系は異なっていますのでご注意ください。(印字位置は、インチの単位系、用紙サイズは、MKS(mm)の単位系を用いています。)

⑤ 印刷情報と用紙サイズの情報の使い方は、以下のようにする。

(a) “印字位置の指定”は、文字列を単位として行き、“文字列の最初の文字”の印字位置を指定する。

(b) 文字列自身と、その文字列の印刷情報の対を明らかにするために、文字列の直前に、その文字列の印刷情報を記述する。(つまり、“~c”で始まる16バイトに続いて、印字される文字列が続いている、という形になります。)

(c) 用紙の幅と長さの情報は、印刷ファイルの先頭に記述する。(これは、BP_Printとのインタフェースの約束となっています。)

この方式を採用して、印刷ファイルの中に、印刷情報を組み込むと、サーバー側からクライアント側に印刷ファイルがコピーされ、印刷を行うモジュールのBP_Printに渡されるまで、印刷情報は(単なる文字列であるため)OSや他のモジュールによって解釈されることがありません。

印刷情報を解釈することができるのは、BP_Printだけです。

3.3.3 “帳票印刷”の動作過程(ストックフォーム印刷、A4印刷の動作過程との比較)

印刷情報を、印刷可能な文字列として印刷ファイルに組み込むこととしたので、(帳票印刷用の印刷ファイルを作成する、という点を除けば、)“帳票印刷”の動作過程は、“ストックフォーム印刷”、“A4印刷”の動作過程と、同じになります。

つまり、第2相から第1相に制御を戻すところから、GCPrintOutを呼び出すところまで、“印刷形態によって動作が異なる”というところはありません。

第2相から第1相に制御を戻すときでも、帳票印刷では、印刷用紙のスタイルとして、“PS-STOCKFORM”、“PS-AYON-TATE”、のどちらかをセットします。

(BP_Printは、帳票印刷の場合は、“PS-STOCKFORM、PS-AYON-TATE”の情報を使っていませんので、どちらかがセットされていれば差し支えありません。)

また、GCPrintOutが呼び出されてからも、動作は全く同じとなります。

帳票印刷と、他の印刷の差が出てくるのは、BP_Printの動作に入ってからです。

これは、帳票印刷の印刷ファイルの内容を解釈して、各文字を印刷の制御コードである、と認識することができるのは、BP_Printだけであり、BP_Print以外のモジュールやプロセスでは、印刷ファイルの内容を解釈しないためです。

GCPrintOutから、帳票印刷の印刷ファイルを渡されて起動された場合、BP_Printは、その印刷ファイルから、実際に所定の用紙に印刷を行うために、以下のような動作を行います。

① ファイルの先頭を読んで、印刷用紙の幅と長さを受け取ると、帳票印刷であることを知ります。^[*20]

[*20] これにより、フォントの自動変換は行われなくなります。

次いで、基本的には^[*21]、次の動作を繰り返し、ファイルの内容を全部印刷します。

② ファイルの各行を読んで、印字位置とフォントサイズの指定があると、その指定の部分と、後続する(印刷するべき)文字列の部分とを切り離します。

③ フォントを指定のサイズに変え、指定の位置から文字列をプリンタに出力します。

[*21] 印刷が完了するまでには、詳しく見ると、ページの制御等がありますが、このドキュメントでは省略いたします。

なお、BP_Print の実装には、Windows API を用いていますので、BP_Print からプリンタに出力しても、そのまま直ちにプリンタが動作するわけではありません。

Windows API では、印刷の管理をページ単位(と、ドキュメント単位)で行っているため、1 ページ分のデータを送り終えたという信号を送るまでは、プリンタには出力されません。

これにより、ページの下の方に印刷した後、ページの上の方に印刷しても、実際の印刷結果は、あたかも上から順番に印刷したように見えます。

3.4 印刷の要約

以上から、ソフトウェア部品システム で印刷を行う場合は、作業としてはあまり多くないことがおわかりのことと思います。新規部品で印刷を行う場合、実施しなければならない作業を要約すると次のようになります。

- (1) 新部品の第 2 相(業務処理プログラム)で、COBOL のレポート機能を使って、印刷ファイルを作成する。
印刷ファイルは、
 - ◆ “ストックフォーム印刷”、“A4 印刷” では、既に印刷イメージの通りになっているようにする。
 - ◆ “帳票印刷” では、“~L” で始まる用紙情報の行を先頭に持ち、印刷されるべき文字列ごとに、その直前に、“~c” で始まる印字位置とフォントサイズの情報を持っているようにする。
- (2) 印刷ファイルに改行コードを埋め込む。
 - ◆ “ストックフォーム印刷”、“A4 印刷” では、CBssprt を実行する。
 - ◆ “帳票印刷” では、CChohyo を実行する。(この場合は、改行コードを埋め込む行の値をパラメータで指定する。)
- (3) 第 2 相から戻るときに、印刷ファイル名と、印刷のスタイルをセットする。
 - ◆ “ストックフォーム印刷” では、(PRINT_INFO.cpy の定義で)PS-STOCKFORM、
 - ◆ “A4 印刷” では、PS-AYON-TATE、
 - ◆ “帳票印刷” では、PS-STOCKFORM、または、PS-AYON-TATE を指定する。(BP_Print では使用しないのですが、未定義の値が SSMProcPut のパラメータにセットされないようにします。)
- (4) 第 1 相では、第 2 相から制御が戻ってきたら、ユーザー(オペレータ)に印刷を実行するかどうかを確認する。
- (5) 印刷をするとの確認を得たら、GCPrintOut を呼び出す。

この(5)項以後、GCPrintOut の動作が終了するまで、開発者が行うことはありません。(GCPrintOut の動作が正常に終了すれば、そのときには、印刷が完了しています。)

4. 印刷動作のトラブルシューティング

新しい部品を作成して、その部品から印刷を行う場合、思ったようには印刷できないことがあります。印刷機能が、うまく動作しないときには、印刷の過程の、どちらの相に問題があるかを判断します。もし、印刷がうまくいかない場合は、例えば、つぎのようなアプローチを試みてください。

① BP_Print のプレビュー画面は表示されましたか？

もし表示されていれば、印刷ファイルは、BP_Print に渡されています。

(このような場合は、印刷内容が意図通りでない、というケースが多いようです。)

もし表示内容が正しくない場合は、

1) プレビュー画面を表示したままで、(印刷の動作をサスペンドさせたままで、)

2) サーバー側の印刷ファイルを探し、エディタか Notepad で、その内容を見てください。

◆ ファイルのあるフォルダ は、サーバー側の %BSS_HOME%wrk です。

◆ ファイル名は、XXXPRTO00000nn の形式をしていて、日付と時刻は、その部品を起動した日付と時刻にほとんど一致しています。(XXX は、その部品が属しているサブシステムのコードです。)

内容が印刷の意図通りであれば、業務部品プログラムは、正しい動作をしたので、次項 3) の、クライアント側の印刷ファイルの中を見ます。

意図の通りでなければ、業務処理プログラムが、印刷ファイルを作成する過程をチェックしてください。

3) クライアント側の印刷ファイルを探して、その内容を見ます。

◆ ファイルのあるフォルダ は、クライアント側の %BSS_HOME%wrk^[*22] です。

◆ ファイル名は、通常は数字の文字 1 文字です。そのファイルの日付と時刻は、その部品を起動した日付と時刻にほとんど一致しています。

内容が印刷の意図通りであれば、BP_Print に渡された印刷ファイルは正しいので、BP_Print の内部での動作を調べる必要があります。

なお、帳票印刷の場合は、プレビュー画面も見てください。印字されるべき文字に欠けや、余分な文字がついていることがないことを確認してください。(このような現象が起こっていたら、印刷情報の指定(特に“~c”)の記述に誤りがないかどうか確認をしてください。)

[*22] 1 台のマシンの中に、サーバー側とクライアント側がインストールされている場合は、1 つのソフトウェア部品システムのディレクトリ構造を共有します。(つまり、サーバー側もクライアント側も、同じディレクトリ構造にインストールされます。

この結果、ホーム(%BSS_HOME%)も同じになります。また、印刷ファイルが保存されるフォルダ(%BSS_HOME%¥wrk)も同じになります。

サーバー側の印刷ファイルと、クライアント側の印刷ファイルが、同じフォルダにできるので、印刷ファイルの名前が一致しないようにします。

② プレビュー画面が表示されなければ、

1) 印刷ファイルが作成されているかどうか、調べます。

2) その印刷ファイルの内容は、意図通りになっているかどうかを調べます。

印刷が、意図通りに印刷できない場合、いろいろのケース^[*23] があります。このドキュメントでは、とりあえず手がかりを得るところまでの記述に留めることとします。

[*23] 大きいところでも、プリンタの問題、プリンタのドライバの問題、ポートの設定の問題、ハードディスクの空き容量の不足の問題、などがあります。

この“ソフトウェア部品システムの印刷”に関するお問い合わせは、下記へお願いいたします。

ご注意

- ◎ 本書は著作権法で保護されている著作物です。
- ◎ 本書の内容の一部または全部をソフトウェア部品の普及の目的に限り複製することを認めます。
- ◎ 本書の内容は予告なく変更される場合があります。
- ◎ 本書は内容について万全を期して制作いたしましたが、万一、ご不審な点や誤り、記載漏れなどお気づきのことがありましたら、ご連絡下さい。

ソフトウェア部品システムの印刷

発行者 清川茂満
発行所 株式会社セントラルソフトサービス Q&A センター
〒 444-0831 岡崎市羽根北町 2 丁目 1 番 8
TEL:0564-59-3221 e-mail : QAcenter@css-snet.co.jp
発行日 2008 年 11 月 30 日

Copyright © 2008 CSS Co.,Ltd. Allrights reserved.
