解　説

文書記述言語の標準化動向—IV

標準ページ記述言語（SPDL）とその動向†

高　橋　　亨††

1. はじめに

ISO/IEC JTC 1/SC 18/WG 8（以下，WG 8 と省略）では，電子出版環境下における文書情報の交換を可能にするため，文書記述言語およびフォントに関する一連の規格を開発している．本シリーズ解説では，WG 8 における国際標準化活動の概要と，ここで開発されている主要な規格群について紹介してきた．今回は，フォーマット済みの印刷・表示用文書のレベルで文書情報を交換するための規格である，標準ページ記述言語 SPDL (Standard Page Description Language)[1] を取りあげ，その特徴と最近の動向について解説する．

2. 背　景

レーザビームプリンタを初めとするページプリンタの発達によって，さまざまな書体やサイズのフォントを駆使した文字テキスト，各種の幾何図形，画像などが混在する複雑なページを手軽に印刷できる可能性が開かれてきた．しかし，これらのプリンタで従来提供されてきたインタフェースは，メーカや機種によって異なっていたうえ，シリアルプリンタ以来の命令体系を引きずっていたり，解像度や内蔵フォントなどのハードウェア特性に依存していたりして，ページプリンタのもつ能力を十分に引き出せるものとは言い難かった．このため，文書処理ソフトウェアで各種のプリンタをサポートしようとすると，それぞれの命令体系や機能水準に合わせた出力を生成するためのドライバを多数用意しなければならず，多大な労力を必要とするだけでなく，ソフトウェアの移植・普及を妨げる要因ともなっていた．このような問題を解決するため，特定の出力装置やシステムに依存せずにフォーマット済みの文書データを正確に記述することができる，高水準の共通インタフェースが強く求められてきた．

この要求に対する有力な回答として現れてきたのが，**ページ記述言語** (Page Description Language: PDL) と呼ばれる一群の言語である．PDL は，その名前が示すとおり，印刷や表示の対象となる文書ページをプログラミング言語的な形式に従って記述する．この際，たとえばページ上における位置や大きさの指定に解像度に依存しない実寸単位 (inch や mm) を用いるというように，特定の出力装置に依存しない形でページの内容を記述することができる．

PDL を解釈実行する処理系は，このように装置に依存しない形で表現されたページ記述を入力として，それを出力装置の能力が許す範囲内で最大限に近似するようなページ画像を生成する．したがって，各種のプリンタやディスプレイが同じ PDL の処理系を備えるようになれば，共通のページ記述データを用いて，さまざまな出力装置上で実質的に等価な出力を得ることができるようになる．

このような形で印刷・表示用のデータ形式が統一されることからは大きな利益が得られる．まず，出力形式が一つで済むようになるため，前記のような多数のドライバを用意する必要がなくなり，文書処理ソフトウェアの生産性や移植性が大きく向上する．また，作成中の文書ページを印刷結果と同様な体裁で画面上に表示し，結果を確認しながら編集作業を行う WYSIWYG (What You See Is What You Get) 型の DTP (Desktop Publishing) システムなども容易に実現できるようになる．ユーザ側からみると，品質とコストとのバランスを考慮して多様な出力装置の中から出力先を

† The Standard Page Description Language (SPDL) and it's trends by Toru TAKAHASHI (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所

自由に選択することが可能になる．たとえば，校正段階でのゲラ刷りは安価な低解像度のレーザプリンタに出力しておき，最終的に版面が完成した段階で高解像度の電算写植機から印刷用フィルムに出力するといった方法がとれるようになる．このほか，PDL は実質的に一般のプログラミング言語と同等の記述能力をもっているので，直接には提供されていない機能であっても手続きなどの形で定義することができ，応用プログラムからの多彩な要求に柔軟に対処することができる．

ページ記述データの流通性を高めるためには PDL そのものの標準化が不可欠であり，これが SPDL 開発の主要な動機となっている．

3. SPDL の特徴

SPDL は，既存のページ記述言語である PostScript[2]* および Interpress[3]** をベースに開発されており，これらの言語から多くの特徴を受け継いでいる．PostScript については最近大幅なバージョンアップが行われ，その仕様が PostScript レベル 2[4] として公開されている．SPDL は，レベル 2 で追加された諸機能（拡張された色表現，複合フォント，パターン，フォーム，フィルタなど）もほとんど取り入れている．ただし，SPDL はこれら二つの言語によく似てはいるが，これらのいずれとも完全な互換性を有するわけではないので注意が必要である．

ここで，SPDL の主な特徴を簡単にまとめてみると，以下のようになる．

（1） 階層化された文書構造の概念をもつ．SPDL 文書を構成するデータは，この**構造**を表すものと，具体的なページ記述である**内容**とに分けられる．文書構造を利用することで，内容の解釈に立ち入ることなく，外部からのページ選択やリソースの効率的な管理などが行えるようになる．

（2） 用紙の選択や印刷済みページに対する仕上げ処理の指定などを行い，文書の出力過程全体を制御する一群の文書プロダクション命令（Document Production Instructions : DPI）を備えている．

（3） 内容部を記述する構文はきわめて単純で，カッコを使わずに後置記法で演算順序を指定するスタック型言語になっている．演算や描画処理を行う基本機能はそれぞれオペレータとして提供されている．

（4） インク，マスクおよびクリッピング領域という概念を用いて定義される高水準のイメージングモデルにより，文字テキスト，幾何図形およびラスタ図形（画像）を統一的に取り扱う．これらのいずれについても自由に移動・拡大縮小・回転などの座標変換を行ってページ上に描画することができる．曲線の表現には3次ベジエ曲線を用いる．また，出力装置の種類や目的に応じた各種の色表現（RGB, CMYK, CIE 表色系など）を扱うことができる．

（5） フォントリソースに関する規定が別規格（ISO/IEC 9541[5)~7)]）として言語から分離されている．（フォントリソースのユーザは SPDL だけに限定されないため．）この規格に準拠したフォントであれば文書中から自由に利用することができる．

（6） 上記の規格（9541）に準拠したフォントは複数の組み方向モード用のメトリックスデータをもつことができる．これを利用して，通常の横書き（左から右へ）のほかに，縦書きや右から左への順で文字列を出力することができ，日本語を含む各国語の表記法に対応できる．また，特定の文字符号系に依存せずに出力すべき文字列を指定するためのマッピング機構を備えている．

（7） 処理効率の良いバイナリ形式と，通常の文字だけを用いて表現し，人間が直接読み書きできるクリアテキスト形式の二つの交換形式をもつ．

4. 文書処理モデル

4.1 SPDL の位置付け

図-1 は，著述から可視化された文書の完成にいたる文書処理過程を概念的に示したモデルである．このモデルによれば，文書はまず編集プロセスによって，著述内容そのものと，その論理的な構成を示す論理構造とを備えた編集可能形式文書として生成される．SGML[8),9)] でマークアップされた電子化原稿などがこの例にあたる．

編集可能形式文書を入力として，組版レイアウトプロセスが文書のフォーマティングを行い，文書の空間的構成（レイアウト構造）を決める．こ

* PostScript は米国 Adobe Systems 社の登録商標です．
** Interpress は米国 Xerox 社の登録商標です．

の組版レイアウトプロセスに対して文書の体裁を指示する言語が DSSSL[10] である．組版レイアウトプロセスにより，文書内容のページへの分割や各ページ内における構成要素の配置などが決定された最終形式文書が生成される．SPDL は，この最終形式文書を記述するための言語として位置付けられる．

最後に，表示プロセスが最終形式文書を受け取り，ページ画像の集合を生成して媒体上に出力し，文書を目に見える形に可視化する．（ここでは「表示」という言葉を，紙への印刷などをも含んだ意味で用いている．）

SPDL 文書はこのように組版レイアウトプロセスからの出力として生成されるだけでなく，他の最終形式文書（たとえば ODA[11] のフォーマット済み形式文書）からの変換によって作られたり，グラフィックエディタなどのツールを用いて直接生成される場合もある．

4.2 表示プロセスモデル

SPDL 文書の可視化を行う SPDL 表示プロセスは，文書構造を処理する構造プロセッサと，文書内容を処理する内容プロセッサとの組合せとしてモデル化される．可視化すべき文書が与えられると，まず構造プロセッサが構造解析を行い，文書を各構造要素に分解する．次に，構造に沿って現れる各種の宣言を参照して内容処理に必要となる**解釈環境** (Context of Interpretation) を生成し，内容プロセッサを呼び出す．内容プロセッサは与えられた環境のもとで文書内容を解釈実行して一連の描画処理を行い，ページ画像（またはその一部）を生成する．最後に，構造プロセッサが生成されたページ画像を媒体上に出力する．この過程を繰り返すことで，最終的に文書全体が可視化される．

ここで，入力（文書内容）に対応して内容プロセッサが行う処理は，一種の仮想機械（SPDL 仮想機械）を用いて定義されている．内容プロセッサに与えられる解釈環境は，処理の開始時点での SPDL 仮想機械の初期状態と，処理中に参照される各種のリソースとから構成される．

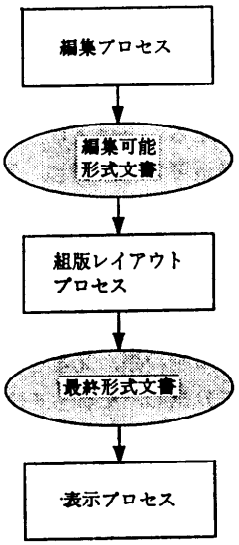

図-1 SPDL の文書処理モデル

5. 文書構造

5.1 構造要素

SPDL 文書の基本的な構造は，ページセット，ピクチャ，およびトークンシーケンスという 3 種類の構造要素を用いた木構造によって表現される．ページセットは連続したページの集合を表すもので，他のページセットまたはピクチャを従属要素として含むことができる．ピクチャは単一のページまたはその一部を表すもので，他のピクチャやトークンシーケンスを従属要素として含むことができる．そして，最下位の構造要素であるトークンシーケンスが，実際に描画すべき内容を保持する．ここで，他のピクチャに従属していない独立したピクチャをページ，階層構造全体の最

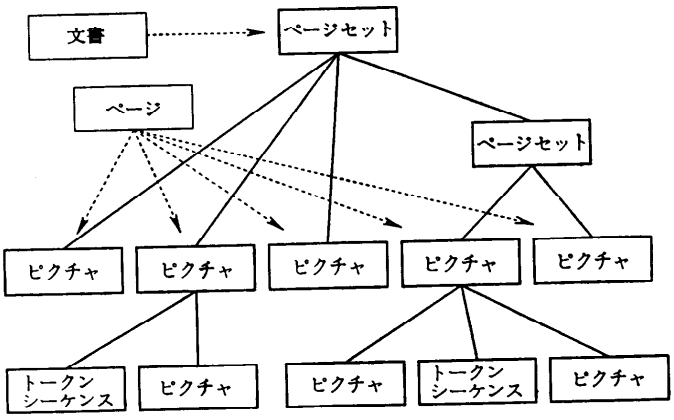

図-2 文書構造の例

上位に位置するページセットまたはページを**文書**と呼ぶ．**図-2** に，このような文書構造の例を示す．

文書構造を備えることにより，SPDL 文書を段階的に小さな構造要素に分割し，外部からその任意の要素を選択して処理することが可能になる．すなわち，内容の詳細に立ち入らずに効率よく文書の各部分を識別し，その一部を取り出したり，部分ごとに異なる処理を施すことができるようになる．たとえば印刷の際，文書を構造に沿っていくつかの部分に分割し，それらを複数のプリンタで並列に処理することで印刷所要時間を短縮できる．また，文書中に一枚だけカラーの図版が含まれていたような場合に，その図版を含むページだけをカラープリンタに出力し，残りのページは印刷コストの安いモノクロプリンタに出力するといった選別が可能になる．文書中の特定の構造要素を取り出して，他の文書を構成するフォーマット済み要素（図版や表）として再利用する処理なども同様の例と言える．

5.2 ブロックとプロローグ

前述した構造要素のうち，ページセットとピクチャとを総称して**ブロック**と呼ぶ．各ブロックはそれぞれ独立した環境を構成しており，あるブロックの内部で行われた処理が後続する他のブロックに副作用を与えないよう，構造プロセッサによる制御が行われる．たとえば，SPDL 仮想機械の状態（描画処理に影響を与える各種グラフィックパラメータの値など）は，ブロックの処理終了時に，その開始時点と同じ状態に復元される．特にそのブロックがページだった場合には，媒体上への出力後にページ画像の初期化も行われる．このように，ブロック境界に関する処理が自動的に行われるため，SPDL のユーザ（フォーマタなど）の側ではページ境界などの前後における処理を意識する必要がない．

各ブロックは，先頭にプロローグと呼ぶ構造要素をもち，ここにそのブロックおよび従属する下位ブロックに共通に適用すべき各種の宣言や手続きを記述しておくことができる．プロローグ内に記述できるものとしては，(1)ファイルなどに格納された外部実体を文書内に取り込むための外部宣言，(2)描画処理時に参照される各種の**リソース**（フォントオブジェクトやカラー空間など）にアクセスするためのリソース宣言，(3)内容処理中に使用すべき一連の定義を含んだ辞書を生成する辞書生成子，(4)初期設定用のセットアップ手続き，(5)文書プロダクション命令を指定する DPI 宣言，などがある．

5.3 文書プロダクション命令

文書プロダクション命令 (DPI) は，ページ画像を媒体上に出力して可視化する文書処理の最終段階を制御するものである．その主な役割は，文書内容中には記述されない（というより，本来記述すべきでない）ような側面について文書の仕上りを指定することと，出力のたびごとに変化しうるようなパラメータを制御することにある．

SPDL が規定する DPI には，寸法や地色，紙質などを指定して媒体を選択するもの，印刷部数や出力ページ範囲を制御するもの，穴あけやステープル処理などの仕上げ処理を指定するもの，などがある．ただし，DPI の実装は必須ではない．たとえば文書のソフトコピーを画面上に表示するような処理系では，ほとんどの DPI は意味をもたない．

DPI は，DPI 宣言の形で文書内に記述される場合（文書内 DPI）と，文書とは別に供給される場合（補足 DPI）とがある．**図-3** は，SPDL 文書の

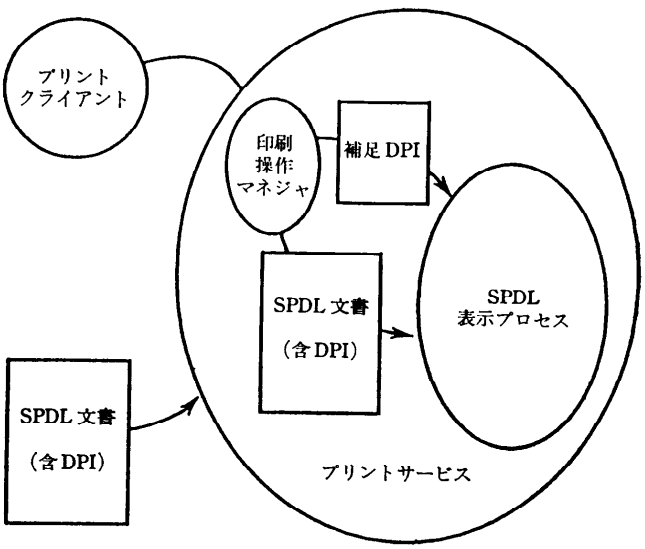

図-3 プリントサービス

印刷を行うプリントサービスと，文書および DPI との関係を示している．このようなプリントサービスは，SPDL 表示プロセスと，印刷ジョブの制御を行う印刷操作マネジャとを構成要素として含み，プリントクライアントからの依頼に従って文書の印刷を行う．プリントサービスの機能に関する標準としては，DPA (Document Printing Applications)[12] が提案されている．

6. 文書内容とその処理

6.1 SPDL 仮想機械

前記のように，内容プロセッサの仕様は SPDL 仮想機械と呼ぶスタック型の仮想機械モデルを用いて定義されている．この仮想機械は以下のような要素から構成される（図-4 参照）．

（1） インタプリタ

インタプリタは，入力トークンシーケンスを解析して個々のデータ要素（トークン）に分解し，各トークンを逐次的に解釈実行する．

トークンには大別してリテラル（数値，文字列など）と，エグゼキュータブル（変数やオペレータに対応付けられた名前オブジェクト）とがある．リテラルはそのままオペランドスタックにプッシュされるが，エグゼキュータブルは対応する値が取り出されて，その種別に応じた処理が行われる．

（2） オペランドスタック

オペランドスタックは，オペレータや手続きに与えるパラメータや，それらの結果として返された値を一時的に保持するために用いられる．

（3） 辞 書 集 合

SPDL では，**辞書**と呼ぶデータ構造を用いて一連の名前とその値との対応関係を記憶し，これを利用して一般のプログラミング言語における変数や名前付きの手続きなどに相当する機能を実現している．ここで，各種の辞書群と文脈スタック（辞書のアクセス順序を規定するためのスタック）とを含む記憶領域全体を辞書集合と呼ぶ．文脈スタックの底にはオペレータの名前と動作定義との対応を示すシステム辞書，その上には大域的な変数や手続きを定義するためのユーザ辞書が置かれている．

（4） 状 態 変 数

状態変数は，SPDL 仮想機械の状態を規定する一群の大域的なデータであり，特に，各種のグラフィック操作を制御する暗黙のパラメータとして用いられるものをイメージング変数と呼ぶ．文字テキストの出力に用いるフォントオブジェクトを指定する CurrentFont，線の太さを指定する CurrentStrokeWidth，クリッピング領域を示す CurrentClipRegion などがある．

（5） イメージャ

イメージャは，インタプリタからの指示に従って描画処理オペレータを実行し，上記のイメージング変数群を参照しつつページ画像を書き換えていく．

6.2 簡単なプログラム例

ここでは，簡単なプログラム例を用いて，SPDL 仮想機械の動作を追ってみることにする．たとえば，次のようなトークンの列が与えられた場合を考える．

2 50 60 Add Multiply

インタプリタは，最初のトークン（数値の 2）に出会うと，これをオペランドスタックにプッシュする．続く 50 と 60 についても同じである．次に **Add** が現れると，文脈スタックをサーチしてその定義を捜す．実はこれはオペレータなので，システム辞書でその定義が見つかり，対応する動作が実行される．オペレータ **Add** は，オペラン

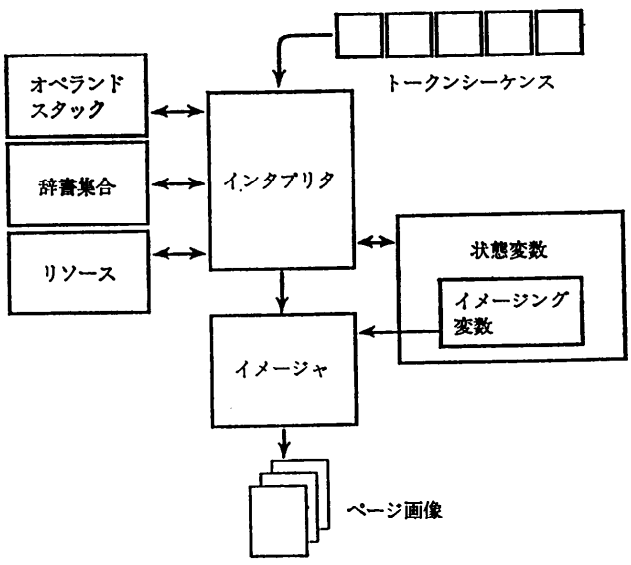

図-4 SPDL 仮想機械

ドスタックから二つのオペランドを取り出し，その和を求めてこれをオペランドスタックにプッシュする．次の **Multiply** もオペレータであり，二つのオペランドを取り出してその積を求め，これをオペランドスタックにプッシュする．結局，(50+60)*2 という演算が行われたことになり，結果の 220 がオペランドスタックに残される．**図-5** に，このプログラムの実行過程におけるオペランドスタックの状態変化を示す．

紙面の都合上詳しい説明は省略するが，このような単純なスタック型の言語構造をとることにより，SPDL では条件分岐や繰り返し，手続き呼出しなどの実行制御機能もすべてオペレータの形で実現されている．

上記のような演算・制御オペレータのほか，SPDL のオペレータには文字や図形，画像の描画処理を行うもの，座標変換やグラフィックパラメータの設定を行うもの，出力装置の特性に合わせた細かなレンダリング制御を行うもの，パターンやフィルタ，フォームなどを扱うもの，エラー処理を行うものなどがある．SPDL が規定するオペレータは合計 190 個ほどで，PostScript（レベル 2 機能を含めると約 380 個のオペレータをもつ）と比べるとかなり整理された仕様になっている．

6.3 イメージングモデル

各種の描画処理オペレータが実行するイメージング操作は，ページ画像の上にインク，マスク，およびクリッピング領域によって決まる**イメージ要素**を次々に塗り重ねていくというモデルによって統一的に表現される．**図-6** に，このイメージングモデルの概要を示す．イメージングモデルを構成する各要素は，それぞれ次のような役割を果たしている．

（1） イ　ン　ク

インクは，イメージ要素の描画に用いる色を指定する．この色は，色の表現方法を規定する**カラー空間**と，カラー空間内での座標という形で特定の色を選択する**色指定**とによって決められる．インクは常に不透明なものとして扱われるので，新たにイメージ要素が描かれると，重複する位置にそれまで描かれていたものは覆い隠される．

一般的には全体が均一な色からなる単純なインクが用いられるが，各画素がそれぞれ固有の色をもつ画像データ全体をインクとして用いたり，小さな図形の集まりであるパターンを定義してこれをインクとすることもできる．パターンをインクに指定して領域の塗潰しを行った場合，領域内にはそのパターンを構成する図形群がタイル状に繰り返し描かれる．

ここで，SPDL で利用できるカラー空間には，大別して装置カラー空間，CIE 関連カラー空間，特殊カラー空間の 3 種類がある．装置カラー空間は，われわれにもなじみ深いグレースケールや RGB，および印刷系での色表現（減法混色）に対応した CMYK などの総称である．CIE 関連カラー空間は，出力装置の特性に依存しない正確な色再現を実現するためのもので，CIE（国際照明委員会）の定めた各種の表色系標準に準拠した形で色を表現する．特殊カラー空間は，ルックアップテーブルを介した間接的な色指定やカラーセパレーションなどを扱うためのものである．

（2） マ　ス　ク

マスクは，描画すべきイメージ要素の形状（インクが塗られる範囲）を規定する．幾何図形の場合，線分や円弧，ベジエ曲線などを組み合わせて構築される**パス**（path）をベースとしてマスクが決定される．ラスタ図形（画像）では，画像全体を包む矩形をマスクとして扱う場合と，特定の値をもつ画素の集合によって表される図形をマスクとする場合とがある．文字テキストの描画では，各文字の形状を規定するアウトラインデータなど

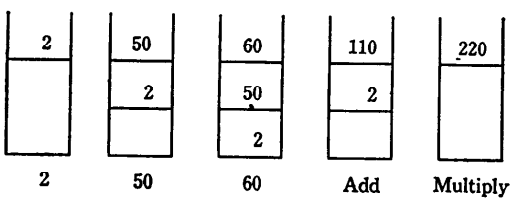

図-5　オペランドスタックの変化

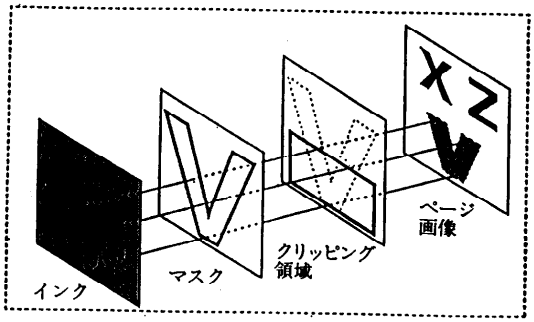

図-6　イメージングモデル

がマスクとして用いられる.

(3) クリッピング領域

クリッピング領域は，ページ画像上で描画処理の影響を受ける範囲を制限するのに用いられる．マスクによって指定される形状のうち，クリッピング領域の内側に入る部分だけが実際に書き換えの対象となる.

6.4 座標系

SPDL で用いる座標系には，参照座標系 (Reference Coordinate System : RCS) とユーザ座標系 (User Coordinate System : UCS) の二つがある．RCS は，ページ上における位置や大きさをあいまいさなく表現するためのもので，ISO 規格という性格を反映してミリメートル単位の座標系になっている．原点は媒体の左下隅に置かれ，右向きに X 軸，上向きに Y 軸をとる直交座標系である.

これに対して，描画処理時のパラメータ指定に使用される座標系が UCS である．デフォルトでは UCS は RCS と一致しているが，この UCS は，座標変換を行う一連のオペレータを用いて，座標系自身を移動したり，拡大縮小したり，回転させたりすることができる．したがって，ユーザは自由に自分に都合の良い座標系を設定して使用することができる．**図-7** にこの両者間の関係を示す.

6.5 文字列のマッピング

テキストデータの交換に用いられる文字符号は，本来各文字の意味概念を表現するものであって，文字の視覚的な形状を直接規定するものではない[13]．しかし，具体的／物理的なページ記述を取り扱う SPDL や，フォント情報交換の際には，

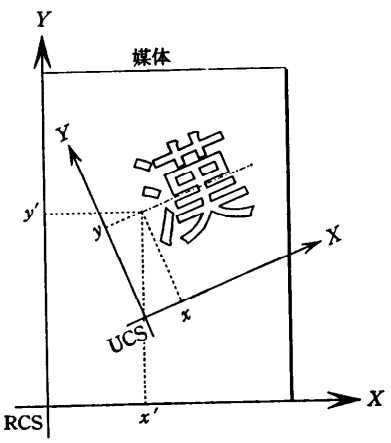

図-7 UCS と RCS

符号化方式や意匠デザインとは独立に個々のグラフィックシンボルを識別するための手段が必要になる．このため WG 8 では，抽象化された（デザインに依存した特徴を取り除いた）グラフィックシンボルである**グリフ** (Glyph) の概念を導入している[14]．一つの文字概念にいくつものグリフが対応する場合もあれば，逆に複数個の文字からなる文字列が単一のグリフで表される場合もある．時代や環境の変化に応じて現れるさまざまな異体字などが前者の例であり，欧文印刷時に特定の文字列 (fi, ffl など) を単一の字形として扱う合字などが後者の例である．数学記号やロゴマークのように，文字概念とは対応しないグリフもある．無限集合的なグリフの性格を考慮して，グリフそのものは規格化の対象とはされず，その登録手続き[15]だけが定められている．各グリフは登録機関である AFII (Association for Font Information Interchange) に登録することでグリフ識別子 (Glyph Identifier) を割り当てられ，一意に指定できるようになる.

SPDL は内部的にはこのグリフ識別子を用いて出力すべき文字を指定する．たとえば，グリフ識別子 afii : 41 を指定してオペレータ ShowGlyph を実行することで，ラテン文字の大文字Aが出力される．とはいえ，個々の文字を逐一グリフ識別子で指定していくのでは効率が悪いので，一連のオクテット（バイト）の列を 0 から 255 までの整数の並びとして解釈し，各整数値またはその組合せからグリフ識別子を求めるマッピング機構を備えている．この対応を示す表をグリフインデックスマップ (Glyph Index Map) と呼ぶ．グリフインデックスマップは入れ換え可能であり，各種の符号化方式に対応することができる.

SPDL で取り扱う個々のフォント（基本フォント）は，それぞれ最大 256 個までのグリフに対応したデータしか保持することができない．そこで，漢字のように多数のグリフを含む文字集合を扱う際には，複数の基本フォントを階層的に組み合わせた複合フォントを使用する．複合フォントからグリフを選択するマッピング方式には，8/8 マッピング，1/7 マッピング，9/7 マッピング，インターバルマッピング，エスケープマッピングなどがある．たとえば 8/8 マッピングでは，2 バイトを単位とし，その最初の 1 バイト（フォントイ

ンデックス）を用いて複合フォントを構成する下位フォント群の中から一つの基本フォントを選択し，次の1バイト（グリフインデックス）でこの基本フォント中に含まれる特定のグリフを指定する．適切なマッピング方式を用いることで，JIS，シフト JIS，EUC などで符号化された文書を出力することができるようになる．

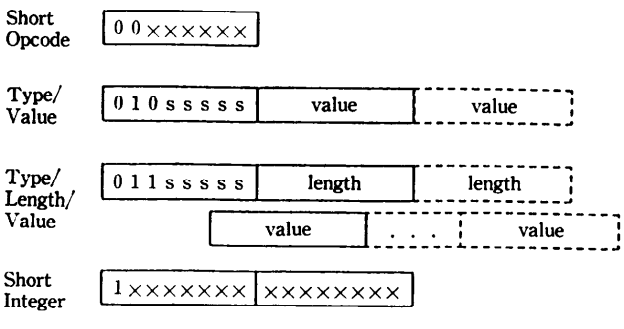

図-8 バイナリ形式でのトークン構造

7. 交換形式

SPDL 文書の交換形式には，処理効率の良いバイナリ形式と，通常の文字だけを用いて表現し，人間が直接読み書きできるクリアテキスト形式の二種類がある．文書構造は，バイナリ形式では ASN. 1[16),17)]，クリアテキスト形式では SGML を用いて表現する．文書内容については，それぞれの形式ごとに最適化された表現方法が定められている．

バイナリ形式では，文書内容を構成する各トークンは図-8 に示す Short Opcode, Type/Value, Type/Length/Value, Short Integer のいずれかの形で表現される．クリアテキスト形式では，英数字や空白などからなる文字列の形でトークンの列を表現する．ちなみに，**6.2** で用いたプログラム例を両方の形式で表現してみると，以下のようになる．

バイナリ形式：

9002 9032 903c 31 4083

クリアテキスト形式：

2 50 60 add mul

バイナリ形式については，各トークンを 16 進表記で示している．また，上の例から分かるように，クリアテキスト形式でのオペレータ表記には，実際には短縮形の名前が用いられる．

8. 審議状況と動向

SPDL については，91 年 10 月を期限として DIS (Draft International Standard) 投票が行われ，賛成多数で成功裏に終了した．コメントは多数提出されているが，今後の作業日程を大きく遅らせるような問題点は見いだされていない．順調にいけば，92 年 9 月ごろには IS となる予定である．また，SPDL はページ記述言語というすでに実績のある分野に関する標準化であるため，処理系の実現に関しても未知の問題が発生することはないと予想される．むしろ SPDL 普及のカギは，フォント情報交換規格に準拠した高品質なフォントが豊富かつ安価に供給されるようになるかどうかにかかっている．

ともあれ，SPDL によって，ページ記述レベルでの文書データの互換性がほぼ確立されることになる．今後に残された課題としては，マルチウィンドウなどに対応した拡張表示機能の導入があげられる．もちろん画面上に文書のソフトコピーを表示する分には現在の仕様で十分であるが，対話型のグラフィックアプリケーションを本格的にサポートするためには，表示内容のスクロールやインクリメンタルな更新，マウスやキーボードからの入力の取り込みなどを行う機能が必要になる．これらの機能は HyTime[18)] (Hypermedia/Time-based Structuring Language) などを用いて記述されたハイパメディア文書の表示を行うためにも必要である．

SPDL の見通しがほぼ固まってきたのにともない，その JIS 化作業も開始されている．現在 DIS 文書をベースとして素案作成を行っており，IS 完成を受けて 92 年度中に原案を作成することを目指している．

謝辞 日頃より有益な検討とご助言をいただいている，日本事務機械工業会 SC 18/WG 8 国内委員会委員の方々に感謝いたします．

参考文献

1) ISO/IEC DIS 10180, Standard Page Description Languge (SPDL) (1991).
2) Adobe Systems Inc. (石田晴久監訳)：ページ記

述言語 PostScript リファレンス・マニュアル，アスキー出版局 (1988).
3) Harrington, S. J. and Buckley, R. R. (富士ゼロックス訳)：インタプレス，丸善 (1989).
4) Adobe Systems Inc.: PostScript Language Reference Manual (SECOND EDITION), Addison Wesley (1990).
5) ISO/IEC 9541-1, Font Information Interchange—Part 1: Architecture (1991).
6) ISO/IEC 9541-2, Font Information Interchange—Part 2: Interchange Format (1991).
7) ISO/IEC CD 9541-3, Font Information Interchange—Part 3: Glyph Shape Representation (1990).
8) ISO 8879, Standard Generalized Markup Language (SGML) (1986).
9) Bryan, M. (山崎俊一監訳)：SGML 入門，アスキー出版局 (1991).
10) ISO/IEC DIS 10179, Document Style Semantics and Specification Language (DSSSL) (1991).
11) ISO 8613, Office Document Architecture (ODA) and Interchange Format, Parts 1-8 (1988).
12) ISO/IEC DP 10175, Document Printing Applications (DPA) (1989).
13) 田嶋一夫：JIS 漢字補助集合案の設定と今後の課題：情報処理学会研究報告，情報学基礎 12-1 (1989).
14) 小町祐史：EP 環境への対応を迫られる文字，字体，字形，フォント：SuperASCII, Vol. 1, #1-2 (1990).
15) ISO/IEC DIS 10036, Procedure for registration of glyph and glyph collection identifiers (1990).
16) ISO 8824, Specification of Abstract Syntax Notation One (ASN. 1) (1990).
17) ISO 8825, Basic encoding rules for Abstract Syntax Notation One (ASN. 1) (1990).
18) ISO/IEC CD 10744, Hypermedia/Time-based Structuring Language (HyTime) (1991).

(平成3年9月12日受付)

高橋　亨 (正会員)

1959 年生．1981 年山形大学工学部精密工学科卒業．1983 年東北大学大学院工学研究科精密工学専攻修士課程修了．同年(株)日立製作所入社．以来，同社システム開発研究所に所属し，ウィンドウ管理システム，文書作成システム，マルチメディアプレゼンテーションシステムなどの研究に従事．ISO/IEC SC 18/WG 8 国内委員会委員．日本機械学会会員．