ソフトウェア開発支援用ライブラリTUIS_LIB

井関 文一*

近年、日本で生産されているほとんどのパーソナルコンピュータは同一のOSであるMS-DOSを採用しているにもかかわらず、グラフィックなどの機能は全くと言っていいほど互換性が存在しない。

今回、富士通のFMRシリーズとNECのPC9801シリーズで同様に作動する、グラフィック機能を主体としたソフトウェア開発支援用の共用ライブラリTUIS_LIBを作成した。そこで本稿ではその考えかた、特徴、構造また機能などについて述べるものである。

1．背 景

現在東京情報大学では、学生のコンピュータ実習用に富士通のFMRシリーズを使用しているが、学生個人が購入する機種はNECのPC9801シリーズ（もしくはその互換機であるEPSONのPC286/386シリーズ）が圧倒的である。

そのため、グラフィック機能などを使用したプログラムを大学で作成しても、その実行形式のプログラムをそのまま自宅のコンピュータで実行することは不可能である。最悪の場合はプログラムのソースコードを書き換え、自宅のコンピュータで再コンパイルする必要に迫られる。

また富士通のFMRシリーズのFMR50とFMR60/70などではそのグラフィックの解像度が違うため、一方で作成したプログラムを他方のコンピュータで作動させた場合、グラフィックの表示のされ方が大きく変わってくる。

このような問題の解答として、MS-OS/2のPresentation Manager[1]やMS-DOSの流れを汲むMS-Windows 3.0[2]などが新しいグラフィックインターフェイスとして注目を集めているが、日本の各社のパーソナルコンピュータ間で完全に互換性があるかどうかは疑わしい限りであり、何よりもそれらを用いたプログラミングはかなり難しく、高度な技量を要求されるという問題がある。

以上のような問題を考慮し、学生が比較的容易にグラフィックなどの機能を扱えるようにするために、ソフトウェア開発支援用のライブラリであるTUIS_LIB（Tokyo University of Information Sciences Software Library）を作成した。

以下第二章ではTUIS_LIBのコンセプト、構造、機能など、TUIS_LIBの考えかたを解説し、第三章では各コンピュータへの実装について説明する。また第四章ではTUIS_LIBを用いた簡単なプログラム例を紹介する。

* 東京情報大学助手

1991年5月15日受理

2．TUIS_LIBの構造と機能

TUIS_LIBはグラフィック・マウス・コンソール機能を主体としたソフトウェア群である。TUIS_LIBには大きな二つのコンセプトが存在する。それは

1．ソフトウェアのマシン独立性
2．データ形式の標準化

である。

「ソフトウェアのマシン独立性」とはソフトウェアがコンピュータの機種に依存することなく、どのようなコンピュータであっても同様に作動することを保証するものである。また「データ形式の標準化」とは標準的なグラフィックなどのデータ形式を定め、TUIS_LIBでこれをサポートすることによりデータの有効利用と普及を計るものである。

2-1　ソフトウェアのマシン独立性と階層構造

「ソフトウェアのマシン独立性」を実現するためにTUIS_LIBは図2-1のような、マシンインターフェイス（MI）、言語インターフェイス（LI）、基本ライブラリ（TUSD）、拡張ライブラリ（TUEX）の4層からなる階層構造になっている。

各階層では階層間のインターフェイスの規約を守っていればその中身の実装についてはどのように行ってもかまわない。

一般に、ユーザは基本ライブラリもしくは拡張ライブラリを使用してアプリケーションを作成することになる。

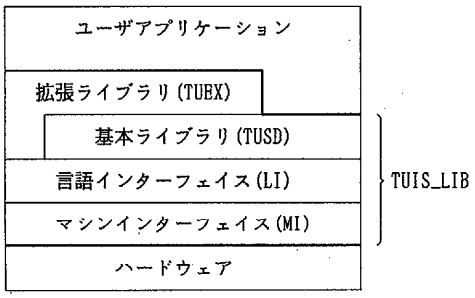

図2-1　TUIS_LIBの階層構造

2-1-1　マシンインターフェイス（MI）

マシンインターフェイス（MI）はハードウェアの違いを吸収し、言語インターフェイス（LI）に対して統一したサービスを提供する。従って言語インターフェイス以上の階層のソフトウェアは（解像度以外の）ハードウェアの違いを意識する必要はなくなる。

マシンインターフェイスは全部で53個の機能からなり、各機能の名前は必ず_Miで始まる（表2-1）。

一般にパーソナルコンピュータの本来のグラフィックの座標系では、左上が原点（0,0）になっているが、マシンインターフェイスでは左下が原点（0,0）になっている。X,Y座標の最大値はそれぞれ、その時の使用マシンに依存し［解像度－1］になる。つまり640×400ドットのマシンでは(0,0)-(639,399)、1120×750ドットのマシンでは、(0,0)-(1119,749)が表示可能範囲となる。

マシンインターフェイスにおけるグラフィック座標を以後ハード座標系と呼ぶ。

使用可能色数も使用マシンに依存するが、内部処理としてはRGB各8ビットつまり約1670万色処理可能である。デフォルトは16色中16色使用可能で、その場合のパレット番号は表2-2ようになる。

2-1-2　言語インターフェイス（LI）

言語インターフェイス（LI）は、マシンインターフェイスの機能呼び出し規約を特定のプログラム言語の機能呼び出し規約に変換する。つまり特定のプログラム言語に対してマシンインターフェイスの機能をそのまま提供する働きをする。従って言語インターフェイスは、機能的にはマシンインターフェイスとまったく同等である。

ただし、特定のプログラム言語の構造上、マシンインターフェイスに含ませるべき機能ではないが、是非とも必要である低レベル機能につ

汎用	
_MiCheckMachine	使用マシンの識別
_MiVarInit	システム変数の初期化
_MiByteMove	バイトデータの転送
_MiByteXor	バイトデータ間の XOR演算
グラフィック機能	
_MiGrInit	グラフィック機能の使用開始
_MiGrTerm	グラフィック機能の使用終了
_MiGrCls	グラフィック画面の消去
_MiSetDispMode	グラフィック環境の初期化
_MiViewPort	ビューポートの設定
_MiGetDispXMax	ハード座標形 Xの最大値
_MiGetDispYMax	ハード座標形 Yの最大値
_MiGetDispXYRate	ドットの縦横比の取得
_MiGetColorMode	カラーモードの取得
_MiSetDrawMode	描画モードの設定
_MiGetDrawMode	描画モードの取得
_MiSetLineMode	ラインモードの設定
_MiGetLineMode	ラインモードの取得
_MiSetPaintMode	ペイントモードの設定
_MiGetPaintMode	ペイントモードの取得
_MiSetPallet	パレットデータの設定
_MiGetPallet	パレットデータの取得指定
_MiGetRedDeg	パレットデータ（赤）の取得
_MiGetGreenDeg	パレットデータ（緑）の取得
_MiGetBlueDeg	パレットデータ（青）の取得
_MiGetXYPallet	XY座標のパレット番号の指定
_MiPset	点を打つ
_MiLine	線を引く
_MiBox	四角形を書く
_MiCircle	円を書く
_MiCircArc	円弧を書く
_MiEllipse	楕円を書く
_MiPaint	領域を塗りつぶす

_MiGrMoveData	画像の移動
_MiGrGetData	画像データの取り込み
_MiGrPutData	画像データの表示
_MiGetCharSize	グラフィック文字サイズの取得
_MiSetCharSpace	グラフィック文字間空白の設定
_MiGrPrintStr	グラフィック文字の表示
マウス機能	
_MiMouseInit	マウスの使用開始
_MiMouseTerm	マウスの使用終了
_MiMouseCursl	マウスカーソルの表示・非表示
_MiMouseButton	マウスボタンの情報取得
_MiMouseCoordX	マウスカーソルの X座標の取得
_MiMouseCoordY	マウスカーソルの Y座標の取得
コンソール機能	
_MiConslCls	コンソール画面のクリア
_MiConslCursl	カーソルの形状指定
_MiConslLocate	カーソル位置の指定
_MiConslColor	コンソール文字の色指定
ファイル入出力機能	
_MiFileOpen	ファイルのオープン
_MiFileClose	ファイルのクローズ
_MiFileRead	ファイルのリード
_MiFileWrite	ファイルへのライト
_MiFileSeek	ファイルのシーク

表 2-1 マシンインターフェイスの機能

0 - 黒	8 - 灰色
1 - 青	9 - 薄い青
2 - 赤	10 - 薄い赤
3 - 紫	11 - 薄い紫
4 - 緑	12 - 薄い緑
5 - 水色	13 - 薄い水色
6 - 黄色	14 - 薄い黄色
7 - 白	15 - 白い

表 2-2 デフォルトのパレット番号

いてはこの階層でサポートする（例えば表3-2)。

FORTRANからTUIS_LIBマシンインターフェイスを使用する場合はFORTRAN言語インターフェイスが、PascalからTUIS_LIBマシンインターフェイスを使用する場合はPascal言語インターフェイスが必要となる。ただし、マシンインターフェイスの機能呼び出し規約はもともとMicrosoft Cの関数呼び出し規約に合わせて作られているので、Microsoft C及びMicrosoft Cと同じ呼び出し規約をもつ言語は言語インターフェイスを必要としない。

言語インターフェイスの各機能の名前は必ずLiで始まる。

この言語インターフェイスの機能により、TUIS_LIBでは複数のプログラム言語をサポートすることが可能となる。

2-1-3　基本ライブラリ（TUSD）

基本ライブラリ（TUSD）は言語インターフェイスにより提供される機能を強化し、より使いやすいものにする。またスクリーン座標系とユーザ座標系をサポートすることによってハード座標系を覆い隠し、マシンの解像度の違いを吸収する。

スクリーン座標系の定義及びスクリーン座標系上の線画データについては「日本語NAPLPS[3]」を参考にし、これに準拠するものとする。

スクリーン座標系では、マシンの解像度に係わりなく、ディスプレイのX方向は0-1（0は含むが1は含まない）であるとする。ディスプレイのY方向はX方向に対する物理的な比で決められ、通常は0-約0.75程度である。

このスクリーン座標を表すためにTUIS_LIBでは符号付固定小数点法（16ビット）を用いる。この16ビット中何ビットを使用して座標系を表現するかは自由であり、デフォルトでは13ビットを用いる。つまり小数点は第13ビットと第14ビットの間にあるものとする（図2-2）。

ユーザはこのスクリーン座標上にグラフィックの表示領域としてビューポートを自由に設定することができる。デフォルトではディスプレイ全体がビューポートとなっている（図2-3）。

スクリーン座標系に対して、ユーザ座標系と呼ばれる仮想的な座標系が存在する。このユーザ座標系は32ビットの浮動小数点で表現（一般にはIEEE標準表記法）できる範囲の空間をサポートしている。ユーザはこの広大な空間のどの部分でも自由に切りだして、ビューポートに表示できる。この切りだした部分をVDC（Virtual Device Coordinate）と呼ぶ（図2-4）。

VDCのデフォルトはスクリーン座標系に一致している。

基本ライブラリの各機能の名前は通常Tsで始まるが、ユーザ座標系に関連した機能の名前はVcで始まる。表2-3にC言語における基本ライブラリの機能の一覧を挙げる。

2-1-4　拡張ライブラリ（TUEX）

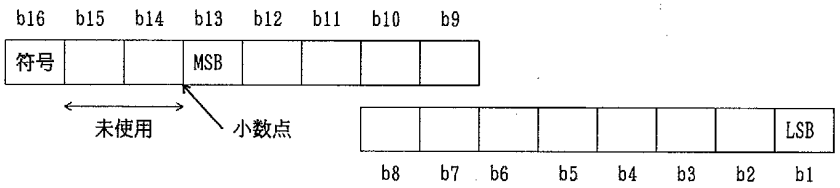

図2-2　デフォルトでのスクリーン座標変数の構造

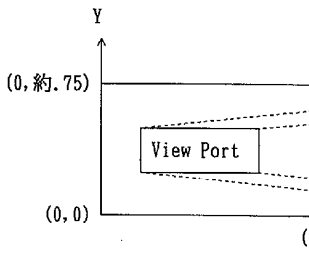

図2-3　スクリーン座標系

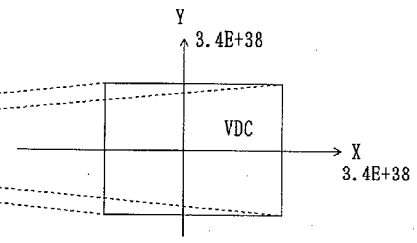

図2-4　ユーザ座標系

汎用

TsCheckMachine	使用マシンの識別
TsInit	簡易イニシャライゼーション
TsTerm	簡易ターミネーション
TsByteMove	バイトデータの転送
TsByteXor	バイトデータ間の XOR演算
TsPower	累乗
TsFarMalloc	メモリ領域の確保
TsFarFree	メモリ領域の開放

グラフィック機能

TsGrInit	グラフィック機能の使用開始
TsGrTerm	グラフィック機能の使用終了
TsGrCls	グラフィック画面の消去
TsSetReso	スクリーン座標系の大きさ指定
TsViewPort	ビューポートの設定
TsTransHrdX	スクリーン X座標をハード座標へ
TsTransHrdY	スクリーン Y座標をハード座標へ
TsTransScnX	ハード X座標をスクリーン座標へ
TsTransScnY	ハード Y座標をスクリーン座標へ
TsTransUsrX	ハード X座標をユーザ座標へ変換
TsTransUsrY	ハード Y座標をユーザ座標へ変換
TsSetDrawMode	描画モードの設定
TsGetDrawMode	描画モードの取得
TsSetLineMode	ラインモードの設定
TsGetLineMode	ラインモードの取得
TsSetPaintMode	ペイントモードの設定
TsGetPaintMode	ペイントモードの取得
TsSetPallet	パレットデータの設定
TsGetPallet	パレットデータの取得指定
TsGetRedDeg	パレットデータ（赤）の取得
TsGetGreenDeg	パレットデータ（緑）の取得
TsGetBlueDeg	パレットデータ（青）の取得
TsGetXYPallet	XY座標のパレット番号の指定
TsPset	点を打つ
TsLine	線を引く
TsBox	四角形を書く
TsCircle	円を書く
TsCircArc	円弧を書く
TsEllipse	楕円を書く
TsPaint	領域を塗りつぶす
TsGrMoveData	画像の移動
TsGrGetData	画像データの取り込み
TsGrPutData	画像データの表示
TsGetCharSize	グラフィック文字サイズの取得
TsSetCharSpace	グラフィック文字間空白の設定
TsGrPrintStr	グラフィック文字の表示

マウス機能

TsMouseInit	マウスの使用開始
TsMouseTerm	マウスの使用終了
TsMouseCursl	マウスカーソルの表示・非表示
TsMouseButton	マウスボタンの情報取得
TsMouseCoordX	マウスカーソルの X座標の取得
TsMouseCoordY	マウスカーソルの Y座標の取得

コンソール機能

TsConslCls	コンソール画面のクリア
TsConslCursl	カーソルの形状指定
TsConslLocate	カーソル位置の指定
TsConslColor	コンソール文字の色指定

グラフィックイメージデータ操作

TsWriteGrFile	グラフィックイメージデータの保存
TsReadGrFile	グラフィックイメージデータの展開
TsGrDataPack	グラフィックイメージデータの圧縮
TsGrDataUnPack	グラフィックイメージデータの解凍

・ユーザ座標系で使用できる関数

グラフィック機能

VcSetCoord	ユーザ座標形の範囲(VDC) を指定
VcTransHrdX	ユーザ X座標をハード座標へ
VcTransHrdY	ユーザ Y座標をハード座標へ
VcTransLenX	ユーザ X座標間距離をユーザ座標へ
VcTransLenY	ユーザ Y座標間距離をユーザ座標へ
VcGetXYPallet	XY座標のパレット番号の指定
VcGrPrintStr	グラフィック文字の表示
VcPset	点を打つ
VcLine	線を引く
VcBox	四角形を書く
VcCircle	円を書く
VcCircArc	円弧を書く
VcEllopse	楕円を書く
VcPaint	領域を塗りつぶす
VcGrMoveData	画像の移動
VcGrGetData	画像データの取り込み
VcGrPutData	画像データの表示

マウス機能

VcMouseCoordX	マウスカーソルの X座標の取得
VcMouseCoordY	マウスカーソルの Y座標の取得

表 2-3　C 言語での基本ライブラリの機能

拡張ライブラリ（TUEX）は基本ライブラリの機能を更に拡張し、より高度な機能を付加する。例えば後述する「データ形式の標準化」のサポート機能、簡易 Window 機能などであ

る。ただしこの階層は必須ではなく、オプションであるとする。

2-2　データ形式の標準化とデータ構造

TUIS_LIBで使用されるデータは、その有効利用を考慮して、すべて同様の形式であることが望ましいと考えられる。そこでTUIS_LIBにおいては以下のようなデータ構造を採用する。

データはTUIS_LIB Envelope（以後単にEnvelopと略す）とDATA本体から成っている（図2-5）。Envelopeは256バイト＋αの大きさを持ち、DATA本体に対する色々な情報が含まれている。しかし、EnvelopeはDATA本体と完全に切り離すことも可能である。

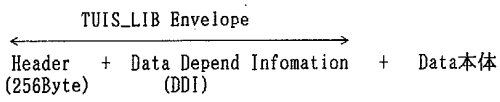

図2-5　TUIS_LIBでのデータ構造

Envelopeのうち、Headerは256バイト固定で種々のデータに共通的な情報を含んでいる（表2-4）。Data Depend Information（DDI）はDATA本体に固有な情報を含んでおり、DATAの種類によりその内容も長さも変化する。

DDI及びDATA本体はデータを取り扱うアプリケーションに依存しており、それらの構造は各アプリケーション毎に別に定めるものとする。

ただし、表2-5に示したデータタイプは基本データタイプとして予約されている。これらのDDIの長さは通常0である。

2-2-1　ノーマルなグラフィックイメージデータの構造

すべてのグラフィックイメージデータ（DATA本体）の先頭4バイトには、そのデー

フィールド名	開始位置(Byte)	長さ(Byte)	内　　容
Envelope Title	0	16	Tuis Envelope,0x0d,0x0a
Data Type	16	16	.....,0x0d,0x0a
Pack Method	32	16	None,0x0d,0x0a
Data Title	48	32	.....,0x0d,0x0a
Auther	80	32	.....,0x0d,0x0a
Date&Time	112	24	yyyy/mm/dd hh:mm:ss,0x0d,0x0a
Memo	136	110	.....,0x0d,0x0a
End Pat	246	2	0x1a,0x00
Data Size	248	4	.... (4Byte 整数)
DDI Size	252	4	.... (4Byte 整数)
DDI	256	α	.....
計	256 + α		

表2-4　Headerの内容

TEXT	テキストデータ
DRAW BINA	バイナリ線画データ（Naplps準拠）
DRAW TEXT	テキスト線画データ（Naplpsコマンド準拠）
GRPH NORM	イメージデータ（ノーマル）
GRPH RLOF	イメージデータ（0x00,0xff についての連長圧縮）
GRPH X1NL	イメージデータ（上下ラインの XORを取って、NULL圧縮）
GRPH X2NL	イメージデータ（上下ラインの XORを二回取って、NULL圧縮）

表2-5　基本データタイプ

タが表示された場合のX方向のドット数とY方向のドット数が、各々2バイト整数で格納されている。

続いてグラフィックイメージのデータが、Y座標の大きい方から1ラインづつ、プレーンの順に並んでいる（図2-6）。この1ライン1プレーン分のデータを以後レコードと呼ぶ。

レコードは必ずバイト境界で区切られている。従ってグラフィックデータのX方向のドット数が8の倍数でない場合、レコードの後方に余分なビットが付くことになる。この余分なビットは一般に不定となる（図2-7）。

2-2-2 グラフィックイメージデータの圧縮

圧縮されていないノーマルなグラフィックイメージデータの場合、データファイルがかなりの大きさになってしまう（640×400ドット、4プレーンで約128Kバイト）。

このためTUIS_LIBの基本ライブラリはグラフィックイメージデータの簡単な圧縮をサポートしている。圧縮はすべて1ライン分のデータを最小単位として行う。圧縮された1ライン分のデータの先頭にはその圧縮されたデータの長さ（バイト単位）を表すための2バイト整数が付加される。

・0x00、0xffについての連長圧縮（RLOF）

1ライン分のデータの中に0x00または0xffが存在した場合、これらについてのみ連長圧縮をかける（図2-8）。例えば0x00が10個続いた場合、この10バイトを0x00、0x0aの2バイト

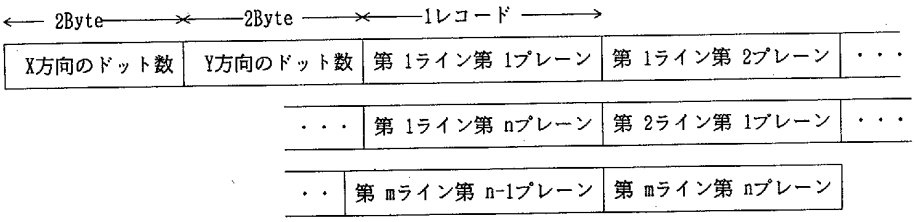

図2-6 ノーマルなグラフィックイメージデータの構造（m ライン n プレーンの場合）

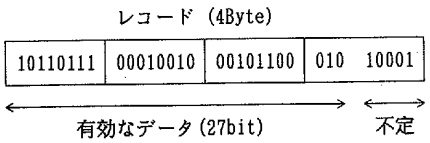

図2-7 レコードのバイト境界（X 方向 27dot の場合）

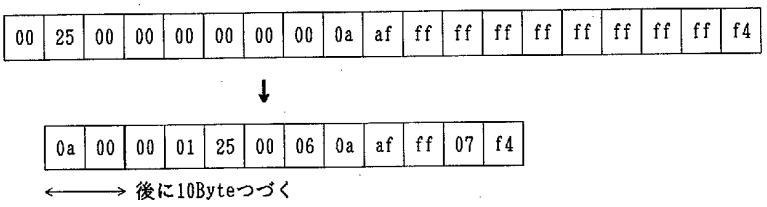

図2-8 0x00、0xff についての連長圧縮

で置き換える。

単純なグラフィックイメージデータは0x00、0xffが続くことが予想されるので、そのようなデータの圧縮に対して効果的である。ただし、0x00、0xffが連続して存在しないような一般的なデータを処理すると、処理前のデータより大きくなるという不合理が生ずる可能性もある。

・NULL(0x00)圧縮（NL）

ビットデータのテーブルを用意して、グラフィックイメージデータ中のすべての0x00を抜き取る。一般には単独では用いず、XOR前処理を行ったデータに対して作用させる。

ビットデータのテーブルは、元々のデータの長さ（バイト単位）分だけのビットデータを用意して、各々のビットを元のバイトに対応させる。つまりテーブルの第10ビット目が0なら元のデータの第10バイト目は0x00であるというように対応をつける。テーブルのビットが1なら対応する元のデータは0x00以外である（図2-9）。

圧縮されたデータの先頭には、そのデータの長さ（バイト単位）を表すための2バイト整数の他に、圧縮前のデータの長さを表すための2バイト整数が付加される。

・XOR前処理（X1、X2）

NULL圧縮を行う場合にその前処理として行う。いわゆる圧縮ではない。

グラフィックイメージデータでは、上下の隣り合ったライン間でデータが著しく変化することは珍しく、ほぼ同じようなデータになるという考えかたに基づいている。上下の隣り合ったライン間でXOR演算を取ると、上下で変化のないドットはすべて0に置き換わる。

XOR演算を二度行えば、更に規則的に変化しているドットも0で置き換えることができる（図2-10）。

3．TUIS_LIBの実装

1991年4月1日現在のTUIS_LIBの最新バージョンはV1.1であり、東京情報大学のパソコンネットであるTUIS-NETの画像処理会議

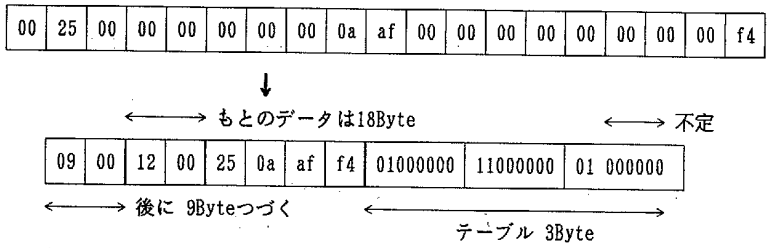

図2-9　NULL圧縮

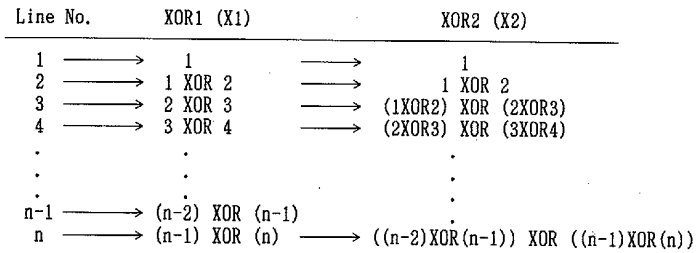

図2-10　ライン間のXORの取り方

室、または商用のパソコンネットNIFTY-SERVEのFFM2フォーラムからフリーウェアとして入手可能である。

ただし、V1.1では拡張ライブラリは未実装となっている。

3-1 マシンインターフェイスの実装

マシンインターフェイスはその性質上インテル社のi8086CPUのアセンブラで記述されている。各機能は、後方の引数（各2バイト）から順番にスタックにプッシュしたあとファーコールで飛び出す。リターンしたら必ずスタック内の引数をクリアしなければならない。

V1.1でサポートされているパーソナルコンピュータは富士通のFMRシリーズ、NECのPC9801シリーズおよびPC9801シリーズと互換性のあるEPSONのPC286/386シリーズである[4]。

グラフィック機能については、FMRではGDS.SYSを、PC9801ではGRAPH.SYSを利用している。またマウス機能では、FMR、PC9801共にMS-DOSに標準で添付されているMOUSE.SYSを利用しているので、これらのデバイスドライバがシステムに組み込まれていない場合はTUIS_LIBを利用することはできない。

それぞれの機種は _MiCheckMachine を実行することによって自動的に判別され、システムに記憶される。以後マシンインターフェイスの機能を呼び出すたびに、システムに記憶された機種用の機能が自動的に選択され呼び出される（図3-1）。

具体的な例として、グラフィック機能の使用開始を宣言する_MiGrInitのリストとマニュアル[5]の一部をそれぞれProgram 3-1と表3-1に挙げる。

3-2 言語インターフェイスの実装

TUIS_LIB V1.1でサポートされている言語は、Microsoft C V5.1、Turbo C V2.0、RM/FORTRAN V2.40、Turbo Pascal V5.0である。

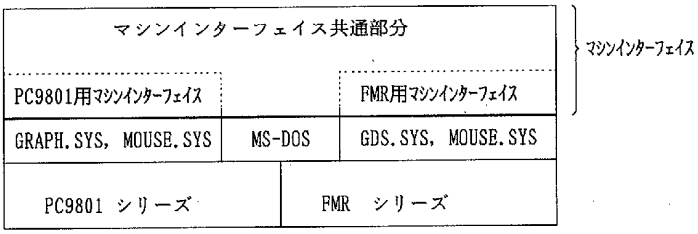

図3-1 マシンインターフェイスの実装

MiGrInit	グラフィクの使用開始
第1引数 作業領域のアドレスのオフセット. 第2引数 作業領域のアドレスのセグメント. 戻り値 AX レジスタ -- 0000h 正常終了 0001h 不正な引数 ffffh 必要なドライバが組み込まれていない. グラフィックの使用を宣言します．グラフィック機能を使用する場合は，他のグラフィック機能を使用する前に必ず呼ばなければなりなせん． 作業領域は予め何らかの方法で確保しておかなければなりません．作業領域は少なくとも 1kバイト以上は確保した方が良いでしょう． この機能に続いて MiSetDispModeを必ず呼ばなければなりません．	

表3-1 マシンインターフェイス機能のマニュアルの一部（_MiGrInit）

このうちMicrosoft CとTurbo Cは言語インターフェイスを介さなくとも直接マシンインターフェイスにアクセスできる。これは元々マシンインターフェイスの呼び出し規約がMicrosoft C（及びTurbo C）の関数呼び出し規約と同じに作られているからである。

RM/FORTRANの言語インターフェイスでは、マシンインターフェイスの機能提供の他に、MS-DOSのシステムコールなど数個の機能が更に付加されている（表3-2、Program 3-2）。

LiINTDOS	MS-DOSのシステムコール
LiGTNPTR	数値変数のアドレス獲得
LiGTCPTR	文字変数のアドレス獲得
LiSTVAL	アドレスへのワードデータの書き込み
LiGTVAL	アドレスのワードデータの取得
LiSTBVAL	アドレスへのバイトデータの書き込み
LiGTBVAL	アドレスのバイトデータの取得
LiGTMEM	メモリ領域の獲得
LiFREMEM	メモリ領域の開放

表3-2　RM/FORTRAN言語インターフェイスの追加機能

Program 3-1　マシンインターフェイス_MiGrInit

```
PUBLIC _MiGrInit                ;  プロシージャのパブリック宣言

_MiGrInit   PROC FAR            ;  プロシージャの始まり
   ASSUME   CS:CODE, DS:DATA    ;  セグメント宣言
   $header                      ;  レジスタ内容の退避
   $get_arg  work_offset,  0    ;  1 番目の引数を work_offsetへ
   $get_arg  work_segment, 1    ;  2 番目の引数を work_segment へ
   mov   es, work_segment       ;  作業領域のセグメントアドレスを es レジスタへ
   mov   bx, work_offset        ;  作業領域のオフセットアドレスを bx レジスタへ
   mov   cx, es:[bx]            ;  作業領域の先頭2バイトの内容を cx レジスタへ
   mov   work_size, cx          ;  cxレジスタの内容を作業領域の大きさとする
   $select_machine FMR_GrInit, PC98_GrInit  ; マシン依存プロシジャーへ分岐
   $tailend                     ;  退避レジスタの復帰
   ret                          ;  リターン
_MiGrInit   ENDP                ;  プロシージャの終わり



; マクロ定義

$header MACRO                   ;  レジスタの退避マクロ
   push  bp                     ;  破壊レジスタ ax
   mov   bp, sp
   push  si
   push  di
   push  ds
   push  es
   mov   ax, DATA
   mov   ds, ax
ENDM


$tailend MACRO                  ;  退避レジスタの復帰マクロ
   pop   es
   pop   ds
   pop   di
   pop   si
   pop   bp
ENDM
```

Program 3-1 マシンインターフェイス_MiGrlnit（続き）

```
$get_arg MACRO  arg, num        ;  num+1 番目の引数を変数 arg へ
   mov   cx, [bp+6+(num*2)]     ;  破壊レジスタ cx
   mov   arg, cx
ENDM


$select_machine  MACRO  fmr, pc98 ; マシン依存プロシジャー選択
   LOCAL Call_FMR                 ; ローカルなラベルの宣言
   LOCAL Call_PC98
   LOCAL Select_End

   cmp   machine_id, FMR
   jz    Call_FMR                 ; 使用マシンが FMRなら Call FMRへジャンプ

   cmp   machine_id, PC98
   jz    Call PC98                ; 使用マシンが PC98なら Call PC98へジャンプ

   mov   ax, 0ffffh
   jmp   Select_End               ; 未サポートマシンなら終わりへ

Call_FMR:
   call  fmr                      ; FMR 用プロシジャーのコール
   jmp   Select_End

Call_PC98:
   call  pc98                     ; PC98用プロシジャーのコール

Select_End:
ENDM
```

Program 3-2 RM/FORTRAN 言語インターフェイス LiNTDOS

```
PUBLIC  LiINTDOS              ; MS-DOS INT21の割り込み

LiINTDOS PROC FAR
   ASSUME  CS:CODE, DS:DATA
   push    es                 ; レジスタの退避
   push    ds
   push    es
   push    bx

   mov     di, 20
   mov     cx, 5
Intdos_Push_Regs:
   sub     di, 4
   lds     si, es:[bx+di]     ; 各引数のアドレスを si レジスタへ
   push    [si]               ; 各引数をスタックへプッシュする
   loop    Intdos_Push_Regs

   pop     ax                 ; スタック中の引数を各レジスタにポップ
   pop     bx
   pop     cx
   pop     dx
   pop     es
   int     21h                ; MS-DOSのファンクションコール

   pop     si                 ; 引数の先頭アドレスの復帰
   pop     di
   push    ax                 ; ファンクションコールの結果をスタックへ
   push    bx
   push    cx
   push    dx
   push    es
   mov     es, di
   mov     bx, si

   mov     di, 20
   mov     cx, 5
```

Program 3-2　RM/FORTRAN 言語インターフェイス LiNTDOS（続き）

```
Intdos_Pop_Regs:
   sub     di, 4
   lds     si, es:[bx+di]   ; 各引数のアドレスを si レジスタへ
   pop     [si]             ; スタックの内容を各引数のアドレスへ
   loop    Intdos_Pop_Regs

   pop     ds
   pop     es
   mov     ax, 00h
   ret
LiINTDOS ENDP
```

3-3　基本ライブラリの実装

基本ライブラリは V1.1 では、言語インターフェイスと同様に Microsoft C V5.1、Turbo C V2.0、RM/FORTRAN V2.40、Turbo Pascal V5.0用が用意されており、各々の言語の機能はすべて関数として実現されている。

Microsoft C と Turbo C 用の基本ライブラリのソースコードはまったく同様のものを使用している。C 言語はパーソナルコンピュータにおいても汎用性が高いので、他の C 言語に対しても基本ライブラリの僅かの変更によって対応可能であることが期待できる（Quick C、Zortech C++でもほぼ作動することを確認している）。

C 言語においては言語本来の機能として、バイト単位の高速ファイル入出力機能をサポートしているので、C 言語用基本ライブラリではファイル入出力機能をサポートしていない。

Turbo Pascal 用の基本ライブラリでは、その機能と名前は C 用のものとほとんど同じであり、ファイル入出力機能と数個の汎用機能が追加されているだけである。TUIS_LIB では Turbo Pascal のユニットもサポートしているので、可能ならユニットを使用すべきである。

他の Pascal 言語については現在のところ何も確認していない。

JIS 規格では FORTRAN77の外部参照名は 6 文字以内となっているが、RM/FORTRAN では機能名は 8 文字以内になっている。従って FORTRAN77に準拠した FORTRAN 言語では RM/FORTRAN 用の基本ライブラリが作動することは絶望的であり、最低限機能名を書き換える必要がある。

4．プログラム例

Microsoft C、Turbo Pascal、RM/FORTRAN 用による簡単なサンプルプログラムを挙げる（Program 4-1、4-2、4-3）。各機能の詳細な記述は各マニュアル[5]を参照すること。

Program 4-1　C 言語によるプログラム例

```
#include <stdio.h>
#include <math.h>
#include "tusd_c.h"


main()
 {
    int c, i, nx=100;
    float xmax, ymax, xmin, ymin;
    float x, y, x1, y1, dx;

    xmin = -2.0; xmax = 2.0;
    ymin = -2.0; ymax = 2.0;
```

Program 4-1 C言語によるプログラム例（続き）

```
    TsInit();                                   /* TUIS_LIBの使用開始 */
    TsSetReso(10);                              /* スクリーン座標の指定 */
    TsViewPort(100,600,900,100,1,6);            /* ビューポートの指定 */
    VcSetCoord(xmin, ymin, xmax, ymax);         /* ユーザ座標の指定 */

    VcLine(xmin, 0.0, xmax, 0.0, 3);            /*  X座標軸 */
    VcLine(0.0, ymin, 0.0, ymax, 3);            /*  Y座標軸 */

    dx = (xmax-xmin)/nx;                        /* 刻み幅 */
    x1 = xmin;
    y1 = sin(x1);
    for(i=1;i<=nx;i++) {
       x = xmin + dx*i;
       y = sin(x);
       VcLine(x1, y1, x, y, 5);                 /* サインカーブ */
       x1 = x;
       y1 = y;
    }

    TsTerm(1);                                  /* TUIS_LIBの使用終了 */
}
```

Program 4-2 Pascal言語によるプログラム例

```
program Test;

uses  Tusd_TPU;

var
   i, c, msok: integer;

begin
    msok := TsInit;                             {TUIS_LIBの使用開始}

    if msok=0 then begin                        {マウスは使用可能か？}
       c := TsMouseCursl(1);
       while (c<>3) do begin                    {押下したマウスボタンが 3以外なら..}
          c := TsMouseButton;
          if (c>0) then begin                   {マウスボタンは押下されたか？}
             i := TsConslColor(5);
             i := TsConslLocate(5,10);
             writeln(c);                        {押下したマウスボタン番号の表示}
             i := TsConslLocate(20,10);
             writeln(TsMouseCoordX);            {マウスカーソルの X座標表示}
             i := TsConslLocate(40,10);
             writeln(TsMouseCoordY);            {マウスカーソルの Y座標表示}
          end;
       end;
    end;

    c := TsConslCls;
    c := TsConslColor(5);
    c := TsConslLocate(20,4);
    writeln(' Hello World !!');

    c := TsConslColor(7);
    c := TsTerm(1);                             {TUIS_LIBの使用終了}
end.
```

Program 4-3　RM/FORTRAN でのプログラム例

```
      PROGRAM TsTEST

      IMPLICIT INTEGER(T,V)
      INTEGER  ERR

      ERR = TsINIT()
      ERR = TsVIEWPT(1000,1000,5000,5000,1,6)
      ERR = VcSTCORD(-2.0, -2.0, 17.0, 2.0)
C     図形の塗りつぶしを指定
      ERR = TsSTMODE(-1, -1, 1)

C     色を変えた円を X方向に順次表示
      DO 100 I=0,15
         ERR = VcCIRCLE(I*1.0, 0.0, 1.0, I, I)
100   CONTINUE

C     図形の塗りつぶし指定を解除
      ERR = TsSTMODE(-1,-1,0)
      ERR = TsBOX(2000,2000,4000,4000,5,0)
C     内接円を描く
      ERR = TsCIRCLE(3000,3000,1000,2,0)

      ERR = TsTERM(1)
      STOP
      END
```

5．まとめ

本稿では、パーソナルコンピュータの機種に依存しない、グラフィック機能を主体としたソフトウェアライブラリTUIS_LIBについて、その考えかた、構造、機能さらに実装について述べた。

実装についてはまだ不十分な点も存在するが、教育現場などにおいては十分有効なツールとして活用できると確信している。

さらにTUIS_LIBを使用したアプリケーションを作成し、そのデータを標準化すれば、他の用途にも活用可能であろう。

TUIS_LIBを使用したアプリケーションとしては日本語NAPLPSのエディタを作成する計画がある。これはTUIS_LIBを計画したときからの目標であり機種に依存しない日本語NAPLPSエディタが完成すればCAIなどにも有効に使用されることが期待される。

参考文献

1）富士ソフトウェア教育部、「OS/2プレゼンテーションマネージャー」、富士ソフトウェア社、1989.

2）松原敦、北郷達郎、中沢信也、「Windows 3.0の衝撃」、『日経バイト』。1990年8月号、no. 75. pp. 240-289.
松原敦、藤田憲治、「日本語Windows 3.0の実像」、『日経バイト』。1991年4月号、no. 84. pp. 188-226.

3）「ニューメディア間インターフェイスの標準化に関する調査研究報告書（別冊）ビデオテックスプレゼンテーションレベルプロトコル日本語機能仕様NAPLPS」、日本企画協会　情報技術標準化研究センター、1986年3月.
CCITT S. 100-1980, *International Information Exchange for Interactive Videotex*, CCITT, 1980.
ISO 2022-1982, *Information Processing-ISO 7-Bit and 8-Bit Corded Character Sets-Code Extension Techniques*, ISO, 1982.

4）インタープログ編、「富士通FMRシリーズ徹底解析マニュアル」、BNN、1989.
アスキー出版局テクライト編、「PC9800シリーズテクニカルデータブック」、アスキー出版局、1990年.

5）F. Iseki and H. Oda、「TUIS_LIBV1. 1 Documentations」、TILMIT、11/1990.