

0 TAD概要

TAD(タッド)は TRON Application Databus の略で、主に BTRON(『TRON プロジェクトとは』を参照)上でデータ交換をおこなうのに利用される、アプリケーション非依存のデータフォーマットである。BTRON上の各種アプリケーションは、データを入出力する場合に TAD を使用することで、他のアプリケーションとのデータの通過性を確保することができる。

TAD の特徴は以下の通りである。

- 一次元データ(文章)と二次元データ(図形・画像)を扱うことができ、相互にネストできる
- 文字コードとして TRON コードを使用しており、多数の文字を扱うことができる
- アプリケーション依存のデータを、他と干渉しないように保持することができる
- TAD 仕様全体をサポートしなくても、必要とするデータのみをアクセスすることができる
- ハイパーテキスト(実身仮身)をサポートしている

TAD は主に BTRON 上で使用されるが、ハイパーテキスト機能 * を除けば、BTRON 以外でも入出力することが可能である。本書の解説では、ハイパーテキスト機能以外の TAD を扱う。

* BTRON のハイパーテキストは、実身仮身とマルチレコードをサポートするファイルシステムによって実現されているため、BTRON 以外ではそのままではアクセスすることができない。

0.1 TADの構造

TADには文章と図形の2タイプがある。

文章 TAD は、主構造が文章であり、文字の並びは行やページに自動的に割りつけられる。文章の中に図形や画像を入れることも可能で、これも自動的にレイアウトされる。これらの図形の中に、さらに文章があっても構わない。

図形 TAD は、主構造が図形であり、図形要素(直線・長方形など)には座標を指定する。座標を指定して文章を配置することが可能で、文章は自動的にレイアウトされ、指定した位置にレイアウトされる。この文章の中に、さらに図形や画像があっても構わない。

もし文字を自由な座標に配置したい場合は、図形 TAD を使用する。図形中に、1文字ずつの文章を入れることで、座標を指定して文字を置くことができる。逆に、もし図形を、用紙内に順に並べたい場合は、文章 TAD を使用する。文章中に、図形を1つずつ入れることで、自動的に配置させることができる。

文章の中に、さらに文章を入れることもできる。たとえば文章 A の中に文章 B が入っている場合、文章 A での状態(現在スクリプトや各種文字修飾など)は文章 B の直前で保持される。文章 B が終わると、さきほど保持された文章 A の状態が復帰される。(文章開始セグメント[1.1] 参照)

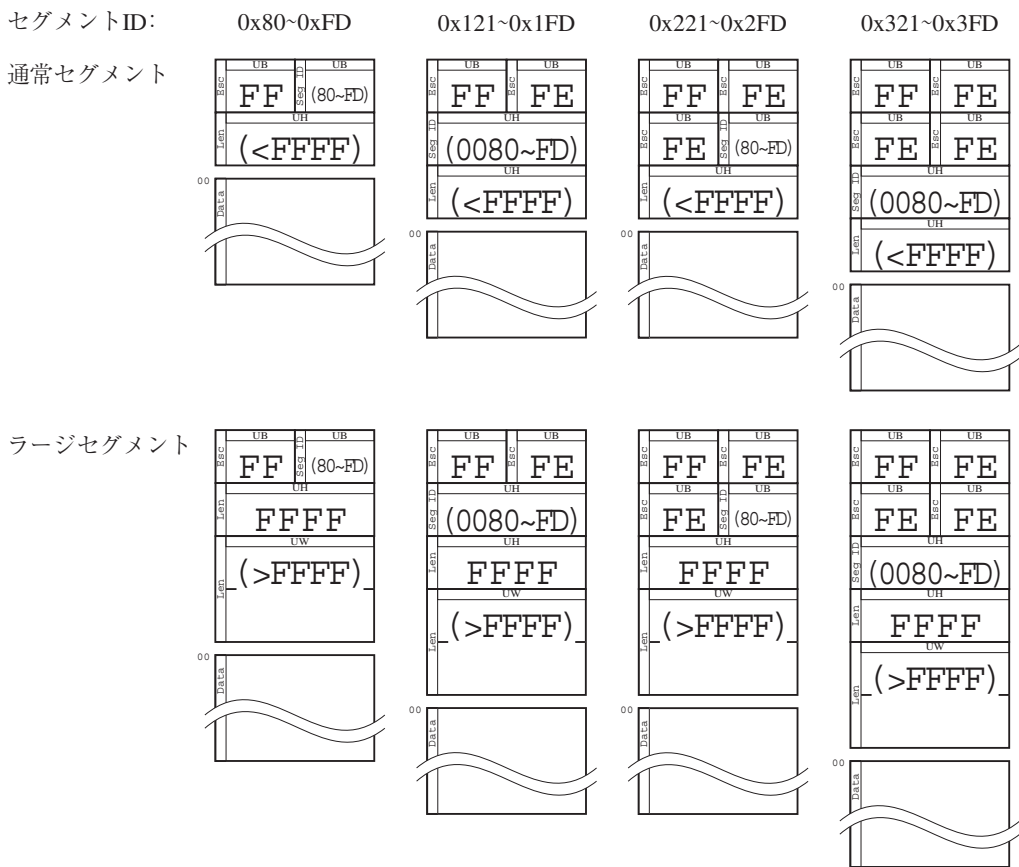
図形の中に、さらに図形をいれることもできる。たとえば図形 A の中に図形 B が入っている場合、図形 A での状態(各種データ定義など)は図形 B の直前で保持される。図形 B が終わると、さきほど保持された図形 A の状態が復帰される。(図形開始セグメント[1.3] 参照)

0.1.1 セグメントの構造と機能

TAD は、「セグメント」と呼ばれるデータが順に並んだバイト列である (TAD における 1 バイトは 8 ビットである)。セグメントには、文字を表すもの、文字修飾や書式などを表すもの、直線や画像を表すもの、用紙サイズを表すものなどがあり、これらを必要に応じて組み合わせることで、文章や図形を表現することができる。いくつかのセグメントは TAD 中での出現位置が決まっているが、多くのセグメントは自由な位置に置くことができる。BTRON の文章エディタでは、文字修飾や書式などのセグメントを「付箋」として表示させることができ、文字と同様に移動・複写・削除が可能である。

セグメントは、セグメントの種類を示すセグメント ID (セグメント ID は特殊な値を持つ文字コードである。TRON コード[0.2]を参照) と、セグメントの長さを示すセグメント長、そしてセグメントの内容を示すバイト列からなる。未知のセグメントでもセグメント長がわかるので、内容を読み飛ばして次のセグメントにたどり着くことができる。従って、もし文字だけを読み取ればよい場合は、文字以外のセグメントをすべて読み飛ばしてしまうことで、処理を簡略化することができる。*

図: セグメントの構造



* 厳密には、データの構造を定義する「文章開始セグメント」「文章終了セグメント」を読み取る必要がある。

逆にエディタのようなアプリケーションでは、自分が出力したデータが、他のアプリケーションによってセグメント単位で移動・複製・削除されていることがあるため、データによってはチェックが必要である。また、一部のセグメントを除き、多くのセグメントは必須とはされていないため、たとえば用紙サイズを定義するセグメントが存在しない場合に、デフォルトの用紙サイズを仮定するような処理も必要になる。

TAD と準 TAD

TAD では 2 バイト以上のデータ型を使用しているが、このとき下位バイトを先に置くか、上位バイトを先に置くかで、2 種類のフォーマットが定義されている。上位バイトを先に置く(ビッグエンディアン)ものが本来の TAD であり、下位バイトを先に置く(リトルエンディアン)ものは準 TAD と呼ばれている。本来の TAD (ビッグエンディアン形式の TAD) を、ここではビッグエンディアン TAD と呼ぶことにする。

仕様上は、準 TAD はあくまで『リトルエンディアン CPU の救済手段』となっているが、現実には流通している TAD はすべて準 TAD であり、ビッグエンディアン TAD は流通していない。さらに、ビッグエンディアン TAD は TRONWARE VOL.50 (パーソナルメディア 1998) においてデータフォーマットの変更がアナウンスされており、部分的に従来のビッグエンディアン TAD との互換性が失われている。

本書では、実装例がない点、情報に揺れがある点を重くみて、断腸の思いでビッグエンディアン TAD を使用しないことを推奨する。

0.1.2 データ型

TAD セグメントのデータ部分は、セグメントによって内容が異なる。しかし、長さや文字サイズなどを示すのに、以下のような共通のデータ型を使用することが多い。例えば文字送りの量を指定する場合には SCALE 型を使用するので、ドット単位で指定するほか、文字サイズの半分として指定することも可能になっている。

- B 8ビット符号付き整数型。値域は -128 ~ +127。
- UB 8ビット符号なし整数型。値域は 0 ~ 255。
- H 16ビット符号付き整数型。値域は -32,768 ~ +32,767。座標表現でも利用される。
- UH 16ビット符号なし整数型。値域は 0 ~ 65,535。
- W 32ビット符号付き整数型。値域は -2,147,483,648 ~ +2,147,483,647。
- UW 32ビット符号なし整数型。値域は 0 ~ 4,294,967,295。
符号なし整数は、主にビットパターンをあらわす場合に使用される。
- PNT {h, v} による 2 次元座標値を表す構造体データ。h、v ともに 32 ビット符号付き整数型。値域はそれぞれ -32,768 ~ +32,767。ただし、負の座標系は使用できない場合がある。
- RECT 左上の点の座標 {left, top} と、右下の点の座標 {right, bottom} による、長方形領域を表す構造体データ。half-open property のため、右の座標 (right) と下の座標 (bottom) は、領域よりも 1 ドット大きくする必要がある。
- COLOR 色を表現する 32 ビットデータ型。TAD ver. 1.20 では「符号あり」、TAD ver. 1.21 以降は「符号無し」とであるとされる。28 ビット

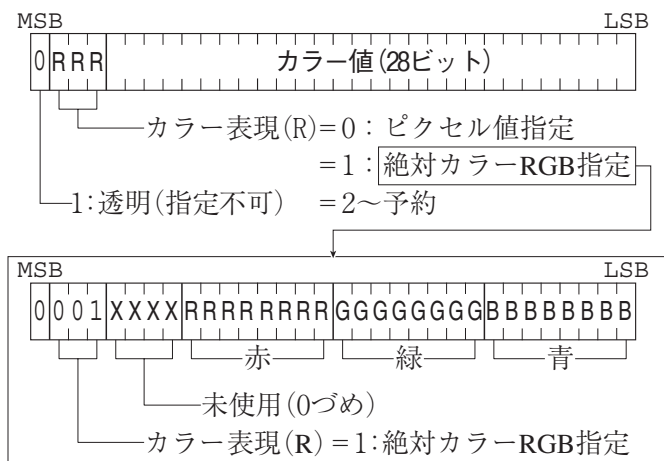
図：COLOR型データ

ト符号なし整数によるピクセル値表現(間接表現)と、RGB 各 8 ビットによる RGB 表現 (直接表現。右図)とができる。文章 TAD においては、ピクセル値表現は実装依存となるため通常は指定しない。

TAD ver. 1.21 以降の仕様では COLOR の MSB は常に 0 で

あるとされるが、これはパターン定義セグメント [3.1.2] の記述と食い違う。この問題は現仕様では解決されない。BTRON のディスプレイプリミティブでは MSB を 1 とした場合は透明色となるが、TAD の COLOR 型では透明色は指定できないとされる。

- UNITS 分解能を示す 16 ビット符号付きデータ型。正の値は 1 cm あたりのドット数を、負の値は 1 inch (25.4 mm) あたりのドット数(の負数)を、それぞれ示す。0 の場合、外側の座標系の継



承を示す。文章開始セグメント[1.1]と図形開始セグメント[1.3]でのみ使用され、たとえば-120 (0xff88) は 120DPI(Dot Per Inch) を示す。

CHSIZE 文字サイズを示す 16 ビットデータ型。ドット数、級、ポイントでの指定をサポートする。U USS SSSS SSSS SSSS の形式を取る。

U の値によって、残る S(14bit 符号無し) 値の単位が決定される。

U=0 外部座標系単位

U=1 1/20 級(Q) 単位(1Q は 1/4 mm。よって 1/80 mm 単位)

U=2 1/20 ポイント(pt) 単位(1 pt は 1/72 inch。よって 1/1440 inch 単位) *

U=3 予約

S は U で指定された単位系での文字サイズを指定する。S=0 の場合は、「未定義」を意味し、アプリケーションに依存となる。

SCALE 長さを示す 16 ビットデータ型。ドット数、比率での指定をサポートする。1NNN NNNN NNNN NNNN もしくは 0AAA AAAA BBBB BBBB の形式で表現され、MSB が 1 の場合は UNITS で指定される座標単位での絶対指定、MSB が 0 の場合は A/B の比率指定となる(B=0 の場合は比率 1)。比率の基準値は、セグメントによって異なる。

RATIO 比率を示す 16 ビットデータ型。AAAA AAAA BBBB BBBB の形式を取る。A/B の値が比率となり、B=0 の時は比率 1 と看做される。比率の基準値は、セグメントによって異なる。

* 1 inch = 25.4 mm。1 pt ≒ 0.3528 mm。ポイントには主として欧州で使われる didot point (1/72 french inch)、米国で使われる American point (1/72.27 inch) などの異なったポイント単位があるが、TAD のポイントはあくまで 1/72 inch である (これを big point と呼ぶ)。

0.1.3 処理系への要求と限界

TAD はきわめて自由度の高いデータ形式であるが、処理系（アプリケーション）によっては処理できないセグメントがあったり、データ量などが制限を超えてしまうことがある。従って、多くのアプリケーションで同じ結果を得る必要がある場合は、できるだけ保守的なデータとする必要がある。処理系にとって負担となったり、処理できなかつたりする可能性のあるデータを、以下に示す。

- 文章や図形の深いネスト。一般には、文章－図形－文章のレベルは問題なくサポートされると考えられる。
- 非常に多い定義や、同一 ID へ多重登録。

画像セグメント〔1.5〕は文章中にも図形中にも置くことができる。ただし 1B の文章エディタでは、文章中の画像は、必ず図形開始〔1.3〕～終了セグメント〔1.4〕に囲まれて存在している。

実装例では、回転角度はサポートされていない。

0.2 TRONコード

TRON コードは原則として 2 バイト固定長の文字コードである。TRON コードには図形文字のほか、制御文字、言語指定コード、特殊文字、TRON エスケープが含まれる。図形文字は、スクリプト切り替え(後述)によって複数の「面」を持つことができるため、無限の文字セットを扱うことができる。

図形文字は、上位 8bit が 0x21 ~ 0x7e、0x80 ~ 0xfd の範囲を、下位 8bit も 0x21 ~ 0x7e、0x80 ~ 0xfd の範囲を、それぞれとる。0x2121 ~ 0x7e7e の範囲を A ゾーン、0x8021 ~ 0xfd7e の範囲を B ゾーン、0x2180 ~ 0x7efd の範囲を C ゾーン、0x8080 ~ 0xfdfd の範囲を D ゾーンと呼ぶ(E ゾーンについては、関連リンクを参照されたい)。例えばシステムスクリプトでは、A ゾーンには JIS 第 1 / 2 水準が、B ゾーンには JIS 補助漢字が、C ゾーンには中国簡体字(GB) が、D ゾーンには、韓国(KSC) が、それぞれ割り当てられている。

制御文字には、0x00 (無効コード)、0x09 (タブ)、0x0a(改段落)、0x0b(改コラム)、0x0c(改ページ)、0x0d(改行)、0x20 (セパレータ) がある(制御文字[0.2.1]を参照)。常に 2 バイトである。

言語指定コードは、0xfe21 ~ 0xfe7e、0xfe80 ~ 0xfefe の範囲である。0xfe21 はスクリプト= 0x21 を、0xfe22 はスクリプト= 0x22 を意味する。言語指定コードによりスクリプトを切り替えることで、2 バイトを超える文字コードを扱うことができる。また言語指定は可変長であり、0xfefe の指定によって言語指定コードがさらに続くことを意味するため、スクリプトは無制限に拡張することができる(言語指定 / 多国語処理[0.2.2]を参照)。

特殊文字は、BTRON のアプリケーション・OS 内部・API でのみ使用され、TAD 中では使用されない。

TRON エスケープは、0xff80 ~ 0xffff の範囲である。ここから TAD セグメントが始まることを意味し、TAD セグメントのセグメントタイプを示す。0xffff の指定によって TRON エスケープがさらに続くことが示されるため、TRON エスケープ自体が可変長である。また、直後にセグメント長、セグメント本体が続く(セグメントの構造と機能[0.1.1]を参照)。

図:TRONコード

(下位バイト)

	0	0x20 0x21	0x7e 0x7f 0x80	0xa0	0xfd 0xfe 0xff
(上位バイト)	0	制御文字	Eゾーン		
	0x20 0x21		Aゾーン	Cゾーン	
	0x7e 0x7f 0x80		Bゾーン	Dゾーン	
	0xfd 0xfe		言語指定	言語指定	
	0xff		特殊文字	TRONエスケープ	☒

未定義

0~0x0020: 制御文字
 0x0021~0x007e: Eゾーン(関連リンク参照)
 0x007f: 制御文字
 0x00a0: 英文固定ピッチ空白
 0xfe21~0xfe7e, 0xfe80~0xfefe: 言語指定
 0xff21~0xff7e: 特殊文字
 0xff80~0xfffe: TRONエスケープ
 0xffff: [EOF]と規定されているが、TAD中では使用しないことを推奨する。

0.2.1 制御文字

制御文字は、改行などのレイアウト制御をおこなうための文字で、通常は画面に表示されず、印刷もされない。以下の制御文字が定義されている。

0x00 (無効コード) は、何の効果も持たない。BTRON のアプリケーションでは文字列の終端を意味するのに利用するが、TAD では通常は使用しない。

0x09 (タブ) は、タブ書式指定付箋 [2.1.2] の有効範囲内ではタブとして機能し、次のタブストップ位置まで描画位置を進めることを意味する。フィールド書式指定付箋 [2.1.3] の有効範囲では改フィールドを意味し、次のフィールドへの移動を意味する。それ以外の範囲では何の効果も持たない。

0x0a (改段落) は、タブ書式指定付箋の有効範囲内では改段落として機能し、ここで段落の切れ目とすることを意味する。フィールド書式指定付箋の有効範囲では改レコードとなり、次のレコードの先頭フィールドへの移動を意味する。それ以外の範囲では、改段落として機能する。

0x0b (改コラム) は、コラム指定付箋 [2.0.2] の有効範囲内では改コラムとして機能し、次のコラムへの移動を意味する。それ以外の範囲では、改ページとして機能する。


0x0c (改ページ) は、ここでページの切れ目となることを意味する。図形内の文章の場合にはページ概念がないので、改段落として機能する。

0x0d (改行) は、段落内もしくはフィールド内での改行を意味する。

0x20 (セパレータ) は、文字幅が不定の空白である。通常は空白と同様に扱われる。

参考 BTRONの基本文章エディタでの表示

0x09 タブ 

0x0a 改段落 

0x0c 改ページ 

0x0d 改行 

0.2.2 言語指定/多国語処理

【言語指定】

TRON コードでは、言語指定コードによって文字属/スクリプトを切り替えることで、無制限の文字コードを扱うことができる。言語指定コードは 0xfe21 ~ 0xfe7e、0xfe80 ~ 0xfefe の範囲であり、特に 0xfefe によってさらに言語指定コードが続くことを意味する。

言語指定コード	切り替わる文字属/スクリプト
0xfe21	0x21 (システムスクリプト)
0xfe22	0x22
0xfefd	0xfd
0xfefe 0x0021	0x121
0xfefe 0x0022	0x122
0xfefe 0x00fd	0x1fd
0xfefe 0xfe21	0x221
0xfefe 0xfe22	0x222
0xfefe 0xfefd	0x2fd
0xfefe 0xfefe 0x0021	0x321
0xfefe 0xfefe 0x0022	0x322
0xfefe 0xfefe 0x00fd	0x3fd

言語指定コードの出現により、以降の文字コードの立脚する文字属/スクリプトが切り替わる。



対応する図形文字の出現しないような、無意味な言語指定をおこなうこと自体は、禁止されていない。

スクリプトの現在値は、文章開始セグメント[1.1]～文章終了セグメント[1.2]によるネスト管理の対象となる。また、文章開始セグメントで言語を直接指定することができる。

切り替わる具体的なスクリプトは、関連リンクにある TRON 文字収録センター (<http://www2.tron.org/>) から参照、確認することができる。

【多国語処理】

多国語処理については、その殆どが予告段階であり、実際の処理形態を提示するまでに到っていない。

TRON の多言語処理は 4 層構造をなしており、言語層、文字属層、スクリプト層、フォント層に別れる。言語層で記述言語を指定し、文字属層では文字概念を記述する（この両者は 1 対多の関係にある）。文字概念は必ずしも固定した字体を持たない。文字概念は、言語依存の処理を経てスクリプト（または「字体」「抽象字形」「glyph」と呼ばれる）へ転写される。文字属層とスクリプト層の関係は多対多となり、単一記述言語が複数のスクリプトを使用したり、複数記述言語が同一のスクリプトを使用したりする。最終的に、スクリプトとフォント指定（フォント指定付箋[2.2.0]とフォント属性指定付箋[2.2.1]）から視覚イメージ（「字形」、**「glyph image」**）を得て、画面表示や印字が行われる。

現時点ではスクリプト層ならびにフォント層のみが実装されており、文字コードとしてスクリプトコードを直接指定している。

主たる参照先は「TRONWARE vol.50」である。

0.3 共通ルール

次章以降の各セグメントの解説において、複数のセグメントに跨って存在するいくつかの共通する特性について、本節で解説する。一般的にセグメントの特徴はセグメントタイプ毎に共通するが、その分類に当てはまらないものを特に詳述する。

0.3.1 付箋の影響範囲

【付箋の影響範囲】

付箋は、それ自体では視覚イメージを持たず、対象となるセグメントや状態、付箋の内部に保持した情報などに対し影響を及ぼすものである。その影響を及ぼした結果として、セグメントの視覚イメージを変化させたり、あるいは付箋の機能の結果を視覚イメージの形で得ることになる。

付箋には大きく分けて「単体で完結している付箋」と「影響対象を持つ付箋」の二種類が存在する。

単体で完結している付箋とは、付箋の内部に保持したデータによって、なんらかの図形イメージを生成したり、あるいはアプリケーションプログラムを呼び出したり、場合によっては単なるメモであったりするものである。固定幅空白指定付箋[2.3.0]、変数参照指定付箋(ID)[2.13.0]、変数参照指定付箋(指定)[2.13.1]がそれに該当し、これらは基本的には影響対象を持たず、その結果が視覚イメージや数値という形で得られるだけである。

影響対象を持つ付箋は、付箋が出現した次の対象セグメント(文字や図形など)に対してから影響を及ぼし、その効果は次の付箋が登場するまで継続する。たとえば、文字カラー指定付箋[2.2.6]によって変更された文字色は、次の文字カラー指定付箋によって別の色が指定されるまで継続する。

一部の付箋では、効果の影響が、付箋の出現位置より前になるものがある。用紙指定付箋[2.0.0]、マージン指定付箋[2.0.1]、枠あげ指定付箋[2.0.5]、ページ番号指定付箋[2.0.6]では付箋が出現したそのページに影響を及ぼす場合が考えられるため、場合によっては影響の結果が付箋の位置を変更してしまい、その結果が再び再レイアウトを発生させる、「レイアウト処理の発振」を起こす場合がある。この問題は現仕様では解決されない。

文字割付指定付箋に含まれる、結合開始／終了指定付箋[2.4.0 / 2.4.1]、文字割付け開始／終了指定付箋[2.4.2 / 2.4.3]、添字開始／終了指定付箋[2.4.4 / 2.4.5]、ルビ開始／終了指定付箋[2.4.6 / 2.4.7]、及び、文字修飾指定付箋[2.5]は、開始付箋と終了付箋をワンセットとして扱い、効果範囲は常に開始付箋と終了付箋の間となる。終了付箋がない場合には、効果が発現しないと考えるのが妥当であるが、現在の実装では終了付箋がない場合、文書の最後までを効果範囲としている。この問題には、継承が行われた場合、より大きな問題に発展する。

文章メモ指定付箋[2.14.0]と文章アプリケーション指定付箋[2.15]、図形メモ指定付箋[3.14.0]と図形アプリケーション指定付箋[3.15]は、その内容をアプリケーションが判断するため、仕様からだけでは、その影響範囲を特定することはできない。

【付箋の継承】

文章 TAD の中に文章 TAD を入れ子にした場合、仕様では「同種の(文章 / 図形) データがネスティングした場合は、原則として未定義の属性は、外側で定義された属性のうち、最も内側で定義された属性が適用される。」ことになっている。このため、付箋の影響はネストした内側の TAD データに継承される。この

際、Push/Popによる状態の保持が行われるが、開始／終了のある付箋の場合、開始が外側にあり、終了が内側にあったとしても、内側のTADが終了し、外側へ戻った際に状態が復帰し、開始付箋の影響が復活する。つまり、TAD全体で開始／終了付箋の数が必ずしも一致しない状況が発生しうる(この場合には、内側の文章TADに終了付箋が単独で存在できなければいけない)。

【付箋の省略】

基本的に、全ての付箋は省略が可能である。付箋は省略されているが、処理上その情報が必要である場合(例えば文字カラー指定付箋[2.2.6]など)は、アプリケーション依存となる。このため、「意図しない結果」を招くことがある。これを避けるために、できるだけ付箋を省略しないことを推奨する。依存関係のある付箋の場合、依存する付箋がなければ、基本的に効果は発揮しない。詳細は各付箋を参照されたい。

0.3.2 TRONコードを含むセグメント

次のセグメントは、セグメントの本体内に TRON コード列を含んでいる。

- 指定付箋セグメント〔1.7〕
- 機能付箋セグメント〔1.8〕
- 設定付箋セグメント〔1.9〕
- フォント指定付箋〔2.2.0〕
- 充填文字指定付箋〔2.3.0〕
- 文字罫線指定付箋(罫線文字)〔2.3.2c〕
- ルビ開始指定付箋〔2.4.6〕
- 行頭禁則指定付箋〔2.4.8〕
- 行末禁則指定付箋〔2.4.9〕
- 変数参照指定付箋(指定)〔2.13.0〕
- 文章メモ指定付箋〔2.14.0〕
- 図形メモ指定付箋〔3.14.0〕

【言語情報】

これらのセグメント自体は、言語指定を持っていない。従って、内包される TRON コード列が言語指定以外の文字コードから始まっていた場合、そのセグメントが存在する位置における言語指定だけが、その識別のための情報ということになってしまう。

しかし、そうであった場合、言語指定情報がセグメントとは無関係に存在することになり、セグメントの移動・複写・削除などの操作を行った場合、意図しない動作を招きかねない。具体的には、システムスクリプトが指定された場所で作られていた充填文字指定付箋を、別の面、例えばラテンスクリプトが指定された場所へ移動させると、内包された TRON コードが指し示す字形が変わってしまう、などという問題が発生する。また、逆にセグメント内の TRON コード列に言語指定コードが入っていた場合、予期せぬ言語切り替えの発生も予想される。

セグメントはその内容を解釈しないアプリケーションによる移動や削除が行われる、という TAD の前提からも、上記のような事態は好ましくない。

このため、TRON コードを含むセグメントに於いては、

1. 内包される TRON コード列の言語指定は、セグメント内に限定して効果を発揮し、外側へは影響を及ぼさない。
2. 内包される TRON コード列が言語指定以外で始まった場合はシステムスクリプト (0x21) が指定されているものとして扱う。

という、二つのルールを推奨する。

【制御文字】

TRON コードを含むセグメントの中に制御文字〔0.2.1〕を含めることができるかについては、セパレータ (0x20) 以外を含めた場合の動作や挙動は予測できない。セパレータ以外の制御文字は含めないことを推奨する。

【固定長フィールド】

- 指定付箋セグメント〔1.7〕
- 機能付箋セグメント〔1.8〕
- 設定付箋セグメント〔1.9〕

以上のセグメントの TRON コード列は固定長となっている。文字列がフィールド長より短い場合は、0 パディングを行うことを推奨する。

0.3.3 アプリケーションIDを含む付箋

アプリケーション ID を含む付箋には、以下の付箋が含まれる。

- 指定付箋セグメント〔1.7〕
- 機能付箋セグメント〔1.8〕
- 設定付箋セグメント〔1.9〕
- 文章アプリケーション指定付箋〔2.15〕
- 図形アプリケーション指定付箋〔3.15〕

アプリケーション ID は、BTRON 上でアプリケーション（実身処理するプログラム）を一意に識別するための、6 バイトの ID である。

BTRON の実身仮身モデルでは、処理対象となる実身には実行機能付箋が、また処理をおこなうアプリケーションには機能付箋セグメント〔1.8〕が存在する。実行機能付箋に含まれるアプリケーション ID を OS が読み取り、同じアプリケーション ID を持つアプリケーションを OS が検索して実行することで、実身とアプリケーションが対応づけられる。アプリケーションがバージョンアップされた場合や他環境に移植された場合には同じアプリケーション ID が使用されるため、一つの実身をさまざまな環境でシームレスに利用することができる。一つの実身に複数の実行機能付箋がある場合、どのアプリケーションを使って実身処理するかを選択することができる。*

TAD は共通データフォーマットであるが、TAD でサポートされていない機能をアプリケーション依存のデータとして TAD 中に保持し、流通させることが可能になっている。このデータを指定付箋（→指定付箋セグメント〔1.7〕、文章アプリケーション指定付箋〔2.15〕、図形アプリケーション指定付箋〔3.15〕）と呼ぶ。BTRON アプリケーションは自分のアプリケーション ID と同じアプリケーション ID を持つ指定付箋を解釈・変更することができる。異なるアプリケーション ID を持つアプリケーションから解釈できる場合もあるが、基本的にはアプリケーション ID が異なる指定付箋は解釈されず、単に指定付箋の移動・複写・削除の操作ができるだけとなる。

アプリケーション ID は、最初の 2 バイトがメーカー ID、次の 2 バイトがデータタイプ ID、最後の 2 バイトがプログラム ID となっている。メーカー ID は、TRON 協会から交付される。メーカー ID とデータタイプ ID の組によってデータタイプが示され、機能付箋などにはデータタイプに対応する名前がデータタイプ名として格納されている。データタイプ名は一般に、そのアプリケーションでの TAD の使用目的にあわせて決められることが多く、実装例では、文章、図形、表などが使用されている。

* TAD は共通データフォーマットであるため、データを作成するアプリケーションと解釈するアプリケーションとが異なってもよい。ただし作業する上では、実身に対してアプリケーションを対応づけておくことが便利であるため、このようになっている。また、アプリケーション依存の各種状態を、固有データとして実行機能付箋に保存しておくことができる。これにより実身ごと・アプリケーションごとに、ウィンドウ位置・表示モードなどを別々に保持することができるという特徴がある。

アプリケーション ID を取得するには、TRON 協会からメーカー ID の交付を受けるか、獲得済みのメーカー ID から必要数だけ切りわけて配布しているところからデータタイプ ID の交付を受ける必要がある（関連リンクを参照）。後者は、たとえばパーソナルメディアの超漢字開発者サイトがおこなっている。なお、アプリケーション ID は一意である必要があるため、交付されていないアプリケーション ID を使用してはならない。

0.3.4 改段落を伴う付箋

次の付箋は段落の先頭になければならない。段落の先頭でない場合は、付箋の直前に改段落があるものと看做し、処理される。

- タブ書式指定付箋[2.1.2]
- フィールド書式指定付箋[2.1.3]
- 文字方向指定付箋[2.1.4]

これらの付箋は、段落を単位として効果を発揮するため、段落の途中で現れた場合は、直前に改段落(0x0a)が存在するものとして処理される。ただし、直前の改段落コードの後(もしくは文書の先頭)から上記の各付箋までの間に可視セグメントおよびタブ(0x09)、改行(0x0d)、セパレータ(0x20)(制御文字[0.2.1]参照)がなければよい、という意味であるので、不可視の付箋がある分には改段落は必要ない。

- 用紙指定付箋[2.0.0]
- マージン指定付箋[2.0.1]
- コラム指定付箋[2.0.2]
- 用紙オーバーレイ定義付箋[2.0.3]
- 用紙オーバーレイ指定付箋[2.0.4]
- 枠あけ指定付箋[2.0.5]
- ページ番号指定付箋[2.0.6]
- 条件改ページ指定付箋[2.0.7]
- 充填行指定付箋[2.0.8]
- 行間隔指定付箋[2.1.0]
- 行揃え指定付箋[2.1.1]
- タブ書式指定付箋[2.1.2]
- フィールド書式指定付箋[2.1.3]
- 文字方向指定付箋[2.1.4]
- 行頭移動指定付箋[2.1.5]
- フォント指定付箋[2.2.0]
- フォント属性指定付箋[2.2.1]
- 文字サイズ指定付箋[2.2.2]
- 文字拡大縮小指定付箋[2.2.3]
- 文字間隔指定付箋[2.2.4]
- 文字回転指定付箋[2.2.5]
- 文字カラー指定付箋[2.2.6]
- 文字基準位置移動付箋[2.2.7]

- 行頭禁則指定付箋[2.4.8]
- 行末禁則指定付箋[2.4.9]
- 文字修飾指定付箋[2.5]

以上の付箋については、「改段落を伴う付箋」の前に存在しても、可視オブジェクトとして描画されないの
で、問題がない。

- 文章メモ指定付箋[2.14.0]
- 文章アプリケーション指定付箋[2.15]

上記 2 付箋については、その動作がアプリケーション依存となるので、問題が発生するか否かを仕様上か
ら判断することができない。

0.3.5 改行を伴う付箋

次の付箋は行または段落の先頭になければならない。行または段落の先頭でない場合は、付箋の直前に改行(0x0d)があるものと看做し、処理される。

- 充填行指定付箋[2.0.8]
- 行間隔指定付箋[2.1.0]
- 行揃え指定付箋[2.1.1]

これらの付箋は、行を単位として効果を発揮するため、行の途中で現れた場合は、直前に改行(0x0d)が存在するものとして処理される。ただし、直前の改段落コードの後(もしくは文書の先頭)から上記の各付箋までの間に可視セグメントタブ(0x09)、セパレータ(0x20)(制御文字[0.2.1]参照)がなければよい、という意味であるので、不可視の付箋がある分には改段落は必要ない。

- 用紙指定付箋[2.0.0]
- マージン指定付箋[2.0.1]
- コラム指定付箋[2.0.2]
- 用紙オーバーレイ定義付箋[2.0.3]
- 用紙オーバーレイ指定付箋[2.0.4]
- 枠あけ指定付箋[2.0.5]
- ページ番号指定付箋[2.0.6]
- 条件改ページ指定付箋[2.0.7]
- タブ書式指定付箋[2.1.2]
- フィールド書式指定付箋[2.1.3]
- 文字方向指定付箋[2.1.4]
- 行頭移動指定付箋[2.1.5]
- フォント指定付箋[2.2.0]
- フォント属性指定付箋[2.2.1]
- 文字サイズ指定付箋[2.2.2]
- 文字拡大縮小指定付箋[2.2.3]
- 文字間隔指定付箋[2.2.4]
- 文字回転指定付箋[2.2.5]
- 文字カラー指定付箋[2.2.6]
- 文字基準位置移動付箋[2.2.7]
- 行頭禁則指定付箋[2.4.8]
- 行末禁則指定付箋[2.4.9]
- 文字修飾指定付箋[2.5]

以上の付箋については、「改行を伴う付箋」の前に存在しても、可視オブジェクトとして描画されないので、問題がない。

- 文章メモ指定付箋[2.14.0]
- 文章アプリケーション指定付箋[2.15]

上記 2 付箋については、その動作がアプリケーション依存となるので、問題が発生するか否かを仕様上から判断することができない。