Web エミュレータ対応

HT　アプリケーション開発マニュアル　（REV1.03）

YSB/YSR/VDB　シリーズ

（Ver１．３．８J）

株式会社スタンダード

STANDARD
Radio Communications

目　次

第 1 章 XML タグの文法

本章では、Web-Emulator における XML タグの文法を定義します。HTアプリケーションはこれらの文法に従って作成しなければなりません。
Web-Emulator は以下の XML タグを提供します。

	タグ	用途
1	XML	ページ全体の属性パラメータを定義します
2	TEXT	テキストラベルを定義します
3	BUTTON	ボタンオブジェクトを定義します
4	INPUT	入力フィールドを定義します
5	SPLASH	スプラッシュ画面を定義します
6	SELECT & OPTION	選択オブジェクトを定義します
7	HINT	ヒント（アラーム等）を定義します
8	PRINT	印刷オブジェクトを定義します
9	SETTING	セッティングオブジェクトを定義します

各 XML タグ記述上の注意

各属性値が 0 またはマイナスであってはいけません。不正な属性値がセットされた場合は、プログラムが異常を引き起こします。

1．1 XML

【機能】

カレントページの属性を定義します。

【文法】

<xml [cache=number] [bitmap=number]>

</xml>

【パラメータ説明】

cache カレントページがキャッシュ可能かどうかを定義します。

1： cache 不可

2： cache 可

デフォルト： 1

bitmap カレントページが使用するドットサイズを定義します。

1 ：16 ドット表示

2 ：12 ドット表示

デフォルト： 1

【特記事項】

- cache 可能ページならば、エミュレータがそのページを自動的にHTの RAM 上にある Cache.dat というファイルの中にキャッシュします。このようなページは次回実行時に自動的に Cache.dat から取得するので、サーバーと通信をする必要がありません。そのためページナビゲーションのパフォーマンスがよくなります。
  一般的に静的なページ(.xml)はキャッシュ可能で、スクリプト言語で生成される動的なページ(.asp, .php 等)は、内容が動的に変化するので、キャッシュできません。
- cache の削除はアプリケーションの中で行うようになっています。詳しくは button タグを参照してください。
- 表示文字のドット数構成（縦×横）は
  16 ドットの場合
  漢字：16 ×16 ドット
  半角：16 ×8 ドット
  ANK ：8 ×6 ドット
  12 ドットの場合
  漢字：12 ×12 ドット
  半角：12 ×6 ドット
  ANK ：6 ×6 ドット
  となります。
- 各属性値が 0 またはマイナスであってはいけません。不正な属性値がセットされた場合は、プログラムが異常を引き起こします。以下の各 XML タグについても同様です。

【プログラムサンプル】

例 1.1

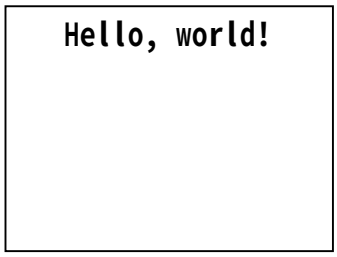
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="5" y="1" >Hello, world!</text>
</xml>
```

これはキャッシュに保存しない 16 ドットのページを示し、例では画面の 1 行目に“Hello, World!”という文字列を表示しています。

例 1.2

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml>
<text mode="1" size="1" x="5" y="1" >Hello, world!</text>
</xml>
```

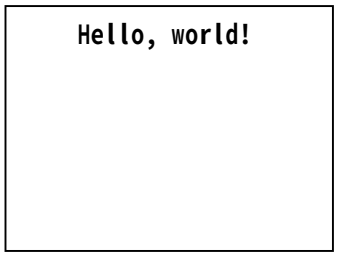
この例では、画面定義パラメータはすべてデフォルトを使用しています。キャッシュ可能で、16 ドットのページを定義しています。



例 1.3

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="2">
<text mode="1" size="1" x="5" y="1" >Hello, world!</text>
</xml>
```

この例では、キャッシュに保存されない 12 ドットのページを示しています。



1．2 TEXT

【機能】

テキストラベルを定義します。
画面に文字列を表示するのに使います。

【文法】

<text [mode=number] [size=number] x=number y=number [name=string]>
[value]
</text>

【パラメータ説明】

mode 文字の表示方式を定義します。
1: 通常表示
2：反転表示
デフォルト：1

size 文字のフォントサイズを定義します。
1：小さい字体
2：大きい字体
デフォルト：1

x, y 文字の表示座標を定義します。
x：左からの桁位置を指定します。
y：上からの行位置を指定します。

name Text タグ名を定義します。
デフォルト：Null

value 表示すべき文字列を定義します。アルファベット、数字および漢字カナ何れも表示可能です。

【数的制約】

一つのページ上に配置できる Text コントロールの数に対する制約はありません。

【特記事項】

- **x、y は必須パラメータ**
- **表示可能文字数（1行の文字数×行数）**

		16 ドット	12 ドット
漢字モード	全角	8 ×4 [16 ×16]	10 × 5 [12×12]
	半角	16 ×4 [16 ×8]	21 × 5 [12× 6]
ANK モード		21 ×8 [8 ×6]	21 ×10 [6×6]

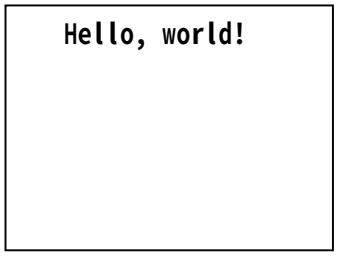
- **表示しようとする文字列の長さが画面の幅を超えた場合、表示可能な分だけ表示します。**

【プログラムサンプル】

例 1.4

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="5" y="1" >Hello, world!</text>
</xml>
```

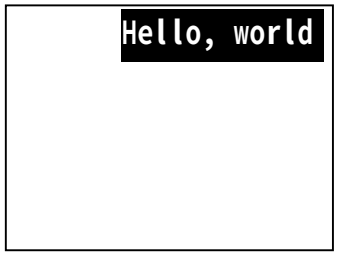
この例では、座標（5,1）の位置に、通常表示方式、小さい字体で“Hello, world！”を表示するテキストラベルを定義しています。



例 1.5

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="2" size="2" x="3" y="1" >Hello, world!</text>
</xml>
```

この例では、座標（5,1）の位置に、反転表示方式、大きい字体で“Hello,world！”を表示するテキストラベルを定義しています。文字列の幅が画面の幅を超えたので、“!”が表示されないわけです。



1．3 BUTTON

【機能】

ボタンを定義します。

キータッチに反応し、所定の動作を実行するのに使います。

【文法】

<button [mode=number] [size=number] x=number y=number key=char type=number [url=string] >
[value]
</button>

【パラメータ説明】

mode ボタンの表示方式を定義します。

1: 通常表示

2：反転表示

デフォルト：1

size ボタン文字のフォントサイズを定義します。

1：小さい字体

2：大きい字体

デフォルト：1

x, y ボタンの表示位置を定義します。

x：左からの桁位置を指定します。

y：上からの行位置を指定します。

key ボタンに対応する HT キー（ASCII 文字）を定義します。

HT キーの指定方法は下表参照

HT キー	対応する ASCII 文字
SF	S
F1	a
F2	b
F3	c
TRG	T
PWR	W
F4	d
F5	E
TRG2	R
BS	B

次ページにつづく

前ページよりつづき

HT キー	対応する ASCII 文字
P1	P
P2	Q
C	C
ENT	e
CH	H
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
.	.
Any key	y

key にはこの ASCII 文字を指定します。

type ボタンに対応する機能を定義します。

1 :所定の url へジャンプ

2 :すべての入力フィールドの内容を所定の url に渡す。

3 :Power OFF

4 :リフレッシュ (cache 削除)

5 :一つ前の入力フィールドに戻る。

6 :入力フィールドの内容を自動的に所定の url に渡す。

7 :カレント Modem を信号の最も強い Modem に切り換える。

8 :垂直スクロールエリアのページダウン (2D コード)

9 :垂直スクロールエリアのページアップ (2D コード)

10:Modem との通信時のリトライ回数及びリトライ間隔を設定する。

11:照合機能実行時に、一つ前の照合データに戻る。

12:照合機能実行時に、全照合結果をクリアした上、最初からやり直す。

13: 照合機能実行時に、カレントデータを照合せずにスキップする。

14：照合機能実行時に、サブページから照合ページに戻る。戻ってからの動作は、対応するoptionのforward値に拠る。
forward＝1の場合、次のデータの照合へ進める。
forward＝2の場合、次のデータの照合へ進めない。
（照合エラー時の再照合などの場合）
15：照合機能実行時に、サブページから照合ページに戻るが、カレント照合データの変更はしない。（動作的には type=14,forward=2 の場合と同じ）
100：バージョン情報を表示します。

url ジャンプ又は渡し先の url を定義します。なお、type が 1,2,6 の場合はこの定義が必須ですが、3, 4, 5, 7, 8, 9, 10, 11, 12, 13, 14, 15, 100 の場合は定義しなくてよい。
デフォルト：Null

value Button 文字を定義します。アルファベット、数字又は漢字カナ何れも指定可能です。また空でもよいです。空の場合はボタンが非可視となるが、キータッチには反応します。

【数的制約】

一つのページに配置できる Button コントロールの数は１６個までです。

【特記事項】

- keyの値は大文字、小文字の区別があるので、注意が必要です。またkeyの値が“y”の時は、どんなキーにも反応するようになっています。
- 入力フィールドの入力状態では、“ENT”、“BS”、“C”という三つのキーがたとえそのページでBUTTONとして定義してあったとしてもBUTTONとしては機能しません。入力状態でのこれらのキーの機能が予め割り当てられているからです。（INPUTタグの特記事項を参照されたい）
- Buttonタイプが6（自動SUBMIT）の場合、オペレータがすべての入力内容を入力後に、そのページの内容が自動的に所定のurlに渡されるようになっていて、何かのキー入力を待つ必要がありません。従ってこのようなボタンに対しては、keyを指定する必要がありません。

【サンプルプログラム】

例 1.6

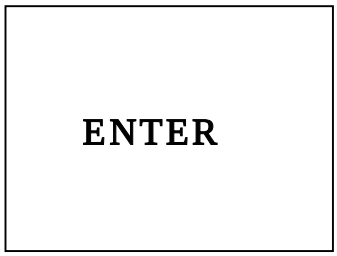
Index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="5" y="5" key="e" type="1"
url="welcome.xml">ENTER</button>
</xml>
```

welcome.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="4" y="2" >Welcome!</text>
</xml>
```

この例は二つのページからなっています。index.xml は座標 (5,5) の所に、通常表示方式、小さい字体の“ENTER”というボタンを表示し、ユーザが ENT キーを押すと welcome.xml ページにジャンプするようになっています。下図にサンプル画面を示します。



例 1.7

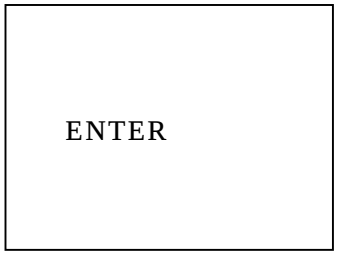
Index.xml:

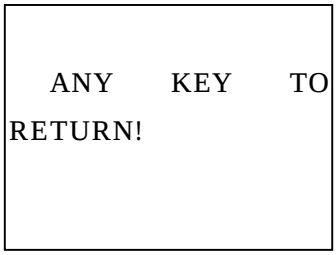
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="5" y="5" key="e" type="1"
url="welcome.xml">ENTER</button>
</xml>
```

welcome.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="4" y="2" key="y" type="1" url="index.xml">ANY
KEY TO RETURN!</button>
</xml>
```

この例では、先ず Index.xml の中で定義した ENTER ボタンを示し、ENT キーを押したら Welcome.xml の中で定義した ANY KEY TO RETURN!を示します。Welcome.xml ページでは、任意のキーを押すと、Index.xml ページに戻るようになっています。





例 1.8

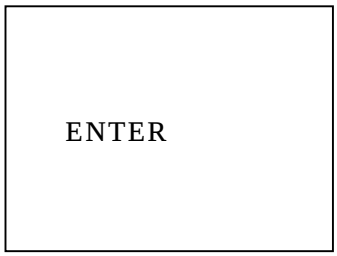
Index.xml:

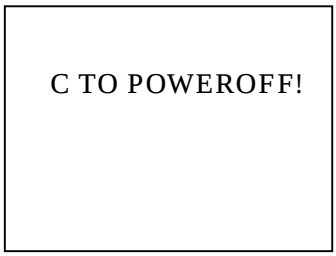
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="5" y="5" key="e" type="1" url="welcome.xml">ENTER</button>
</xml>
```

welcome.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="4" y="2" key="C" type="3" >C TO POWEROFF!</button>
</xml>
```

この例では、先ず Index.xml の中で定義した ENTER ボタンを示し、ENT キーを押したら Welcome.xml の中で定義した C TO POWEROFF!を示します。C キーを押したら、電源が OFF されます。





例 1.9

Index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="2" y="4" key=" " type="2"
url="index2.xml">SENT</button>
</xml>
```

key 属性の指定がなければ、すべてのキーに反応しなくなります。

例 1.10

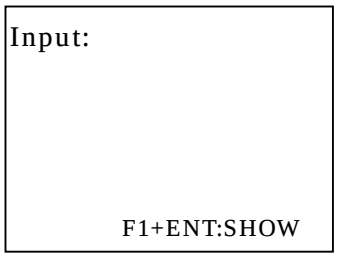
Index.xml:

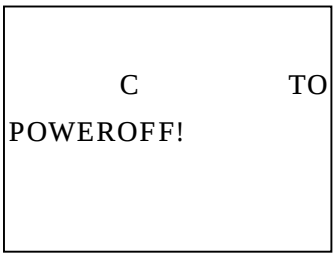
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1" >input:</text>
<input mode="1" size="1" x="8" y="1" format ="1" tip ="1" bar="1" len= "10"
param="number"></input>
<button mode="1" size="1" x="5" y="8" key="ae" type="2"
url="welcome.xml">F1+ENT:SHOW</button>
</xml>
```

この例では、ENT ボタンの key の値が”ae”となっているが、これは複合キーを意味します。この場合入力欄の input タグにデータが入力されてから、複合キーF1+ENT を押すと welcome.xml が呼び出されます。

welcome.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="4" y="2" key="C" type="3" >C TO
POWEROFF!</button>
</xml>
```





例 1.11

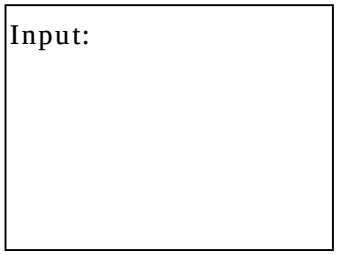
Index.xml:

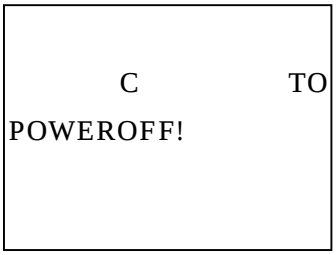
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1" >input:</text>
<input mode="1" size="1" x="8" y="1"   format ="1"   tip ="1" bar="1"
len= "10" param="number"></input>
<button mode="1" size="1" x="5" y="8" type="6"   url="welcome.xml"></button>
</xml>
```

この例では、button の type が 6 なので、入力欄でのデータ入力が確定後直ちに welcome.xml ページが呼び出されます。

welcome.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="4" y="2" key="C" type="3" >C TO
POWEROFF!</button>
</xml>
```

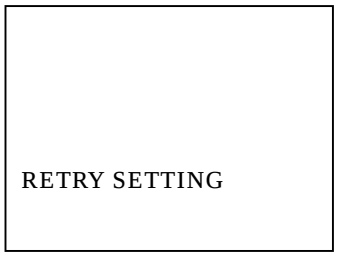




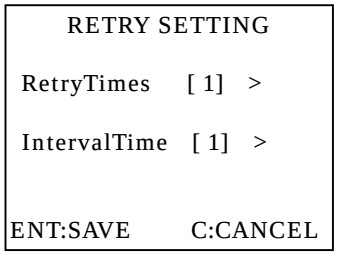
例 1.12

Index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="2" y="4" key="e" type="10">
RETRY SETTING</button>
</xml>
```



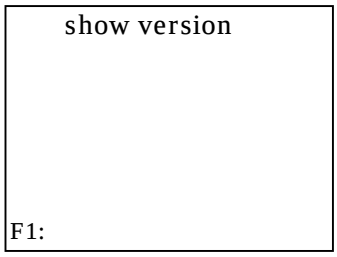
この例では type=10 で、これは Modem との通信時のリトライ回数及びリトライ間隔を指定する RETRY SETTING 画面（組込画面）を呼び出すものです。RetryTimes がリトライ回数を、IntervalTime がリトライ間隔をそれぞれ指します(単位は秒)。



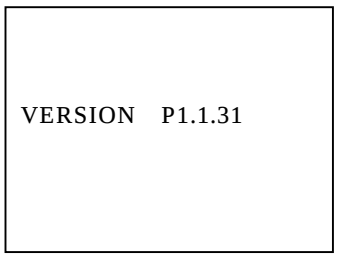
例 1.13

Index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="5" y="1" >show version</text>
<button mode="1" size="1" x="1" y="8" key="a" type="100">F1:</button>
</xml>
```



この例では、type=100 で、F1 キーを押すと、バージョン情報を表示する組込画面を呼び出します。HT ブラウザのバージョン確認に使います。



1．4 INPUT

【機能】

テキスト入力フィールドを定義します

キーボードまたはバーコード入力に使います

【文法】

<input [mode=number] [size=number] [x=number] [y=number] [endy=number] [format=number] [tip=number] [tipx=number] [tipy=number] [bar=number] [barcode=number] [len=number] [min=number] [max=number] [default=string] param=string [goodhint=string] [badhint=string] >
</input>

【パラメータ説明】

mode 文字の表示方式を定義します。

1: 通常表示

2：反転表示

デフォルト：1

size 文字のフォントサイズを定義します。

1： 小さい字体

2：大きい字体

デフォルト：1

x, y 入力フィールドの位置を定義します。

x：左からの桁位置を指定します。

y：上からの行位置を指定します。

endy 2Dバーコード入力時に使われるもので、縦のスクロールエリアの終了行を定義しています。この値が0でなければ、2Dバーコードの入力で且つSize指定が2（大きい字体）の時だけ、縦座標（y，endy）間のスクロールエリアを確保した上、2Dバーコードの内容を表示し、縦スクロールを可能とします（Buttonコントロール項参照）；Sizeが1（小さい字体）の場合は、2Dバーコードの先頭部分の内容を（y，endy）という矩形内に表示するだけで、縦スクロールはできません。

デフォルト： デフォルト無し

【注意】endyの指定がなければ、データの先頭部分だけy行目に表示し、スクロールはできません。ただし、明示的にendy=yとした場合は、スクロールはできます。

format 入力モードを定義します

1 ：数字

2 ：数字＋英字

3 ：数字＋英字＋漢字カナ

4 ：英字

5 ：漢字カナ

6 ：英字＋数字

7：キー入力不可 (バーコード入力のみ)

デフォルト：1

tip 入力モードを表示するかどうかを定義します。

1： 表示しない

2： 表示する

デフォルト： 1

tipx, tipy 入力モードを表示する位置を指定します。

bar バーコード読み取りを許可するかどうかを定義します。

1：許可しない

2：一次元バーコード入力モードとする

3：二次元 (2D) バーコード入力モードとする

4：RFiDの読み取り

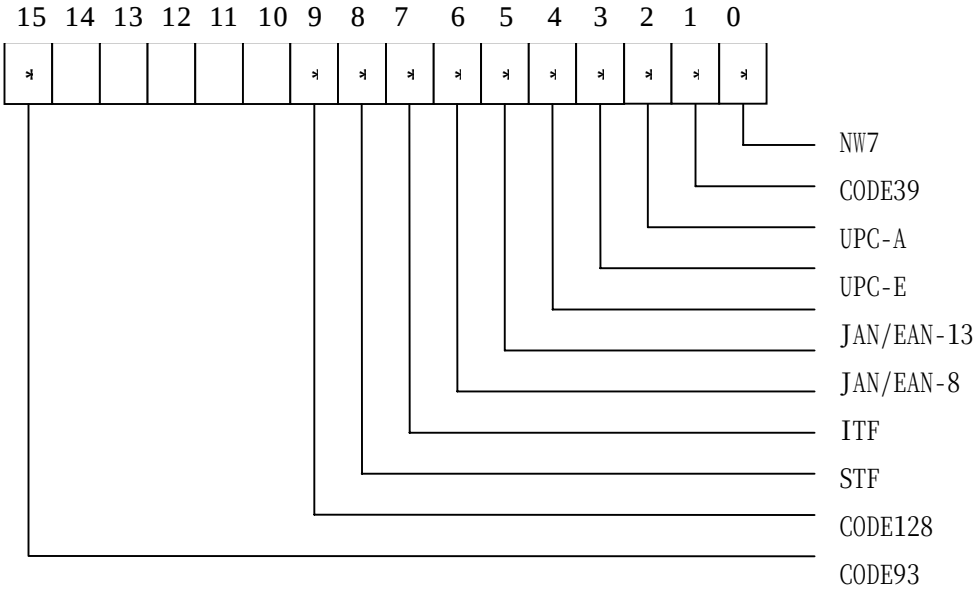
5：RFiD の書き込み

デフォルト： 1

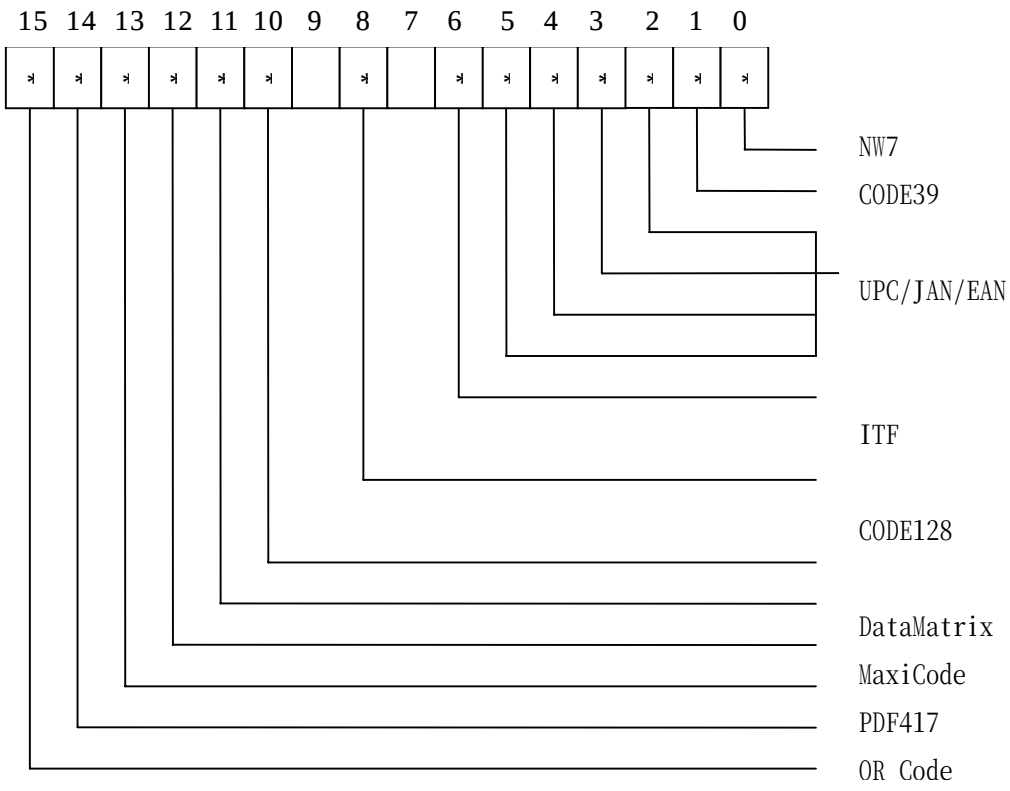
barcode　読み取りを許可／禁止するバーコードの種類を指定します。

各バーコードの許可／禁止は、複数同時設定を可能にするため、ビット単位で指定します。

bar=2の時、barcodeの値の意味は以下の通りです。



bar=3の時、barcodeの値の意味は以下の通りです。



*** 各ビットの指定は次の通りです。**
0 ：禁止、 1 ：許可

bar=4 or 5 の場合の barcode の書式：
"Category：StartBlockNo - Blocks"

ただし、
Category = 1：Tag-it HF-I
= 2：I･CODE SLI
= 3：my-d
StartBlockNo は、読取･書込時の開始ブロック番号、
Blocks は、読取･書込ブロック数 を表わします。

また、StartBlockNo の取り得る値の範囲は、
Tag-it HF-I ： 0～57
I･CODE SLI ： 0～57
my-d ： 0～27
Blocks の取り得る値の範囲は
Tag-it HF-I ： 1～58
I･CODE SLI ： 1～58
my-d ： 1～28

len　入力データの桁数を定義します。
デフォルト：10

min, max　入力データの最小値と最大値を定義します。(文字列として入力データと比較されます)
デフォルト：制限無し

default　入力フィールドのデフォルトデータを定義します。なお、デフォルトデータの先頭に"#@"を付けることにより、ENT キーを押して始めてデフォルト表示をするようになります。
デフォルト：デフォルト無し

param　当該入力フィールドに対応する変数名を定義します。
(デフォルト無し)

goodhint　入力データが正しい (入力データの桁数や値が OK) 時に発するヒントの名前を指定します。(詳細は Hint タグ参照)
デフォルト：Null

badhint　入力データが正しくない (入力データの桁数や値が NG) 時に発するヒントの名前を指定します。(詳細は Hint タグ参照)
デフォルト：Null

【特記事項】

- 入力モードでは、以下のファンクションキーが使えます。

ファンクションキー	対応機能
ENT	入力確定
C	入力フィールドをクリア
BS	直前の文字を削除

- ユーザが最小値と最大値の範囲外のデータを入力してENTキーにて確定しようとした時に、エミュレータがエラー音を出して、エラーデータをクリアします。
- ユーザが入力したデータをサーバーに渡す場合は、type=2のボタンを使わなければなりません。当該ボタンがparamに与えられた変数名と入力データに従ってHTTP QueryStringを生成し、対応する動的ページに渡します。
- 何も入力しないでENTキーを押すと、ある値を表示させたい場合は、その値の先頭に”#@”を付けた文字列を当該入力フィールドのデフォルト値として設定します。
- RfiDの特記事項
  1. RFiD書込み･読出しのデータはASCIIデータのみです。
  2. 書込みのトリガーは、ENTキーによる入力確定時に自動的に行います。トリガーキーを押す必要はありません。
  3. 同一INPUTタグにて、バーコードデータを読み込み、それをRFiDへ書き出すことはできません。また、同じページにバーコード入力用のタグとRFiD書き出し用のタグを別々に用意しても、同じページ内でバーコード入力とRFiD書き出しを同時に実現することはできません。この場合は、バーコードを入力したら、一旦サーバーにSUBMITして、続けてRFiD書き出し用のページを受け取るようにすればよい。（見掛け上は、あたかも同じページ内でバーコードデータ読み込みとRFiD書き出しを同時に実現しているように見えます。）

【数的制約】

一つのページに配置できる Input コントロールの数は 16 個までです。

【サンプルプログラム】

例 1.14

index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1" >common input:</text>
<input mode="1" size="1" x="16" y="1"  format ="1"  tip ="1" bar="1" len= "10" param="number"></input>
<text mode="1" size="1" x="1" y="2" >limited input:</text>
<input mode="1" size="1" x="16" y="2"  format ="1"  tip ="1" bar="1" len= "2" param="number" min="10" max="25" ></input>
<text mode="1" size="1" x="1" y="3" >1 char input:</text>
<input mode="1" size="1" x="16" y="3"  format ="3"  tip ="1" bar="1" len= "1" param="number"></input>
<text mode="1" size="1" x="1" y="4" >barcode input:</text>
<input mode="1" size="1" x="6" y="5"  format ="7"  tip ="1" bar="2" barcode="511" len= "1" param="number"></input>
<text mode="1" size="1" x="1" y="6" >tip input:</text>
<input mode="1" size="1" x="16" y="6"  format ="3"  tip ="2" tipx="12" tipy="7" bar="1" len= "10" param="number"></input>
<button mode="1" size="1" x="5" y="8" key="e" type="2" url="show.asp">ENT:SHOW</button>
</xml>
```

show.asp:

```
<%
 common = request.QueryString("common")
 limited = request.QueryString("limited")
 char = request.QueryString("char")
 barcode = request.QueryString("barcode")
 tip = request.QueryString("tip")

 response.Write "<?xml version=""1.0"" encoding=""ISO-8859-1""
standalone=""yes""?>"&vbCrLf
 response.Write "<xml cache=""1"" bitmap=""1"">"&vbCrLf
 response.write "<text mode=""1"" size=""1"" x=""1"" y=""1"" >common
input:"&common&"</text>"&vbCrLf
 response.write "<text mode=""1"" size=""1"" x=""1"" y=""2"" >limited
input:"&limited&"</text>"&vbCrLf
 response.write "<text mode=""1"" size=""1"" x=""1"" y=""3"" >1 char
input:"&char&"</text>"&vbCrLf
 response.write "<text mode=""1"" size=""1"" x=""1"" y=""4""
>barcode:"&barcode&"</text>"&vbCrLf
 response.write "<text mode=""1"" size=""1"" x=""1"" y=""5"" >tip
input:"&tip&"</text>"&vbCrLf
 response.Write "</xml>"&vbCrLf
%>
```

この例では、以下の入力フィールドがあります。

commom input : 通常の入力フィールド

limited input : 10～25 の間にある数字を受け付けます

1 char input : 一文字のみ入力します

barcode input : バーコード読み取りのみ受け付けます

tip input : 入力時に入力方法を表示してくれる入力フィールド

```
common input:
limited input:
1 char input:
barcode input:
tip input:
  ENT:SHOW
```

ENT キーを押すと、各入力フィールドの入力値を表示します。

例 1.15

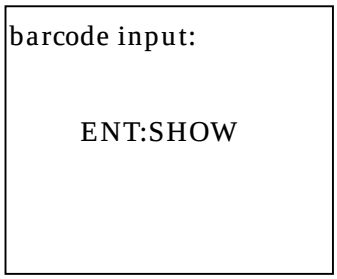
index.xml:

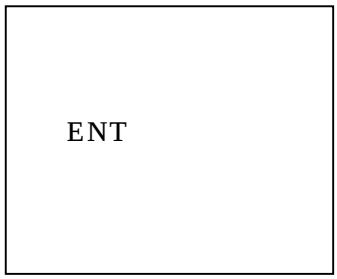
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1" >barcode input:</text>
<input mode="1" size="1" x="6" y="2" format ="7" tip ="2" bar="2"
barcode="511" len= "20" param="number4"></input>
<button mode="1" size="1" x="5" y="3" key="e" type="2"
url="show.asp">ENT:SHOW</button>
</xml>
```

show.asp:

```
<%
 response.Write "<?xml version=""1.0"" encoding=""ISO-8859-1""
standalone=""yes""?>"&vbCrLf
 response.Write "<xml cache=""1"" bitmap=""1"">"&vbCrLf
 response.write "<button mode=""1"" size=""1"" x=""4"" y=""4"" key=""e""
type=""1"" url=""index.xml"" >ENT</button>"&vbCrLf
 response.Write "</xml>"&vbCrLf
%>
```

この例では、先ず index.xml を表示し、バーコード入力後 ENT キーを押してその値をサーバーに送ってから show.asp を呼び出しています。show.asp の ENT キーにて index.xml 画面に戻ります。input タグの format ="7"なので、バーコード入力しかできないことを意味します。この例では、type="2"の時のボタンタグによる入力内容の SUBMIT 方法も示しています。





例 1.16

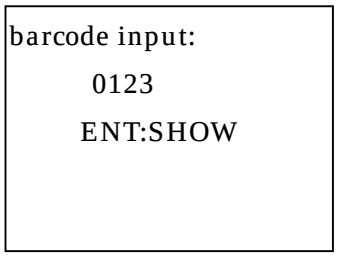
index.xml:

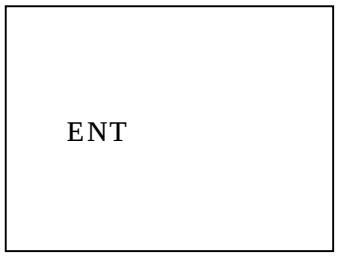
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1" >barcode input:</text>
<input mode="1" size="1" x="6" y="2" format ="3" tip ="1" bar="2"
barcode="511" len= "10" default="0123" param="number4"></input>
<button mode="1" size="1" x="5" y="3" key="e" type="2"
url="show.asp">ENT:SHOW</button>
</xml>
```

show.asp:

```
<%
 response.Write "<?xml version=""1.0"" encoding=""ISO-8859-1""
standalone=""yes""?>"&vbCrLf
 response.Write "<xml cache=""1"" bitmap=""1"">"&vbCrLf
 response.write "<button mode=""1"" size=""1"" x=""4"" y=""4"" key=""e""
type=""1"" url=""index.xml"" >ENT</button>"&vbCrLf
 response.Write "</xml>"&vbCrLf
%>
```

本例は内容的に前例と同じで、唯一の違いは input タグで default="0123"という属性を追加している点です。ここでは、実際のバーコードスキャンをしなければ、デフォルト値 0123 がそのままサーバーに送られるようになっています。





例 1.17

Index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1" >barcode input:</text>
<input mode="1" size="2" x="6" y="2" endy="7" format ="7" tip ="2" bar="3"
barcode="32767" param="number" len="100"></input>
<button mode="1" size="1" x="1" y="8" key="d" type="8">F4:Pagedown</button>
<button mode="1" size="1" x="13" y="8" key="E" type="9">F5:Pageup</button>
</xml>
```

本例では bar=3 なので、2D バーコードの読み取りを意味します。endy は 2D バーコードデータ表示エリアの終了行を示し、二つの button:F4、F5 の type がそれぞれ 8 と 9 で、これは指定エリア内に表示されているデータを上下に改ページする為のキーです。

2D バーコード読取後の画面

```
barcode input:
[)>                 01
968417066
72      840      001
1Z123
```

F4 キーを押すと、下図のように次のページにスクロールされます。

```
barcode input:
72      840      001
1Z123
45675          UPSN
12345
```

F5 キーにて前のページにスクロールされます。

1．5 SPLASH

【機能】

スプラッシュ画面を定義します

スプラッシュ画面とは、所定時間経過後に自動的に他の画面に切り替わる画面のことです。

【文法】

<splash [time=number] [url=string]></splash>

【パラメータ説明】

time 画面表示時間を指定します (単位は 100 ms)

デフォルト：1 秒

url ジャンプ先のページを指定します

デフォルト：Null

【数的制約】

一つのページに配置できる Splash コントロールの数は 1 個だけです。

【特記事項】

- スプラッシュ画面は通常HT電源投入時等に使われます。
- subxmlページの中で使われた場合、URLは指定しなくても良く、この場合は親ページに戻ることを意味します。

【サンプルプログラム】

例 1.18

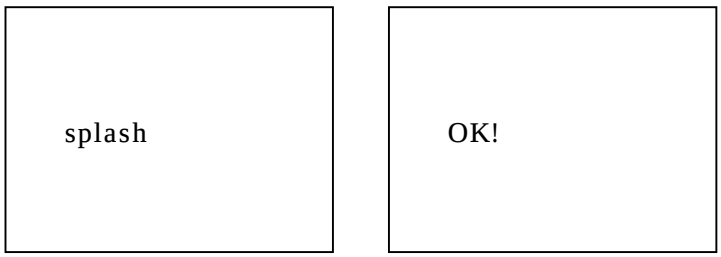
index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="5" y="4" >splash</text>
<splash time="10" url="1.xml"></splash>
</xml>
```

1.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="5" y="4" >OK!</text>
</xml>
```

この例は二つのページからなっています。index.xml はスプラッシュ画面で、1 秒後に自動的に 1.xml ページにジャンプします。



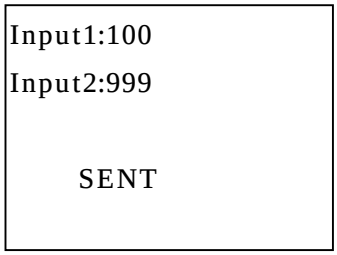
例 1.19

msg.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="5" y="4" >splash</text>
<splash time="20" url=""></splash>
</xml>
```

index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1">input1:</text>
<input mode="1" size="1" x="8" y="1" format ="3" bar="2" len= "20" default="100"
min="99" param="number4"></input>
<text mode="1" size="1" x="1" y="2" >input2:</text>
<input mode="1" size="1" x="8" y="2" format ="3" bar="2" len= "15"
default="999" param="number5"></input>
<button mode="1" size="1" x="5" y="4" key="b" type="2"
url="index2.xml">SENT</button>
</xml>
```



splash タグの url=“”の場合、その spalsh タグを含んだページの種類によって動作が違います。

サブページの場合は、そのサブページを呼び出したページへ

通常ページの場合は、Web サーバーのデフォルトトップページへ

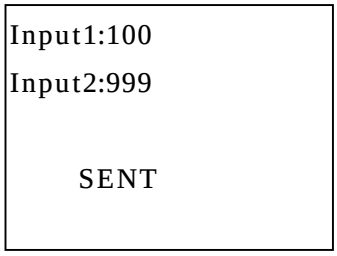
それぞれジャンプするようになっています。

例では、通常ページにある splash タグなので、2 秒後に Web サーバーのデフォルトトップページ、例えば index.xml にジャンプします。

例 1.20

index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="5" y="4" >splash</text>
<splash url＝“ ”></splash>
</xml>
```



前例と同じだが、time 属性が省略されています。省略された場合は、1 秒後に url が指す画面を表示します。

1．6 SELECT＆OPTION

【機能】

選択オブジェクトを定義します

【文法】

<select [mode=number] [size=number] [starty=number] [endy=number] [number=number]>
  <option x=number y=number key=char url=string> option value </option>
  ……
</select>

【パラメータ説明】

mode 文字の表示方式を定義します
1：通常表示
2：反転表示
デフォルト：1

size 文字のフォントサイズを定義します
1：小さい字体
2：大きい字体
デフォルト：1

starty 開始表示位置の縦座標を定義します
デフォルト：1

endy 終了表示位置の縦座標を定義します
デフォルト：2

number option の正確な数を指定します

- **x** 各選択肢の横座標を定義します
- **y** 各選択肢の相対縦座標を定義します
- **key** 当該選択肢に対応する HT キーを定義します
- **url** 当該選択肢よりジャンプする url を定義します

option value 当該選択肢の表示文字列を定義します。アルファベット、数字および漢字カナが指定可能です。

【特記事項】

- 選択オブジェクトは通常メニューに使われます。
- yは**starty**に対する相対座標で、1から数えます。
- **number**値は実際の**option**の数と一致していなければなりません。

【数的制約】

一つのページに配置できる Select コントロールの数は 1 個だけで、一つの Select コントロールに設定できる Option コントロールの数は 20 個までです。

【サンプルプログラム】

例 1.21

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<select mode="1" size="1" starty="2" endy="5" number="4">
<option x="2" y="1" key="1" url="1.xml">1.test1</option>
<option x="2" y="2" key="2" url="2.xml">2.test2</option>
<option x="2" y="3" key="3" url="3.xml">3.test3</option>
<option x="2" y="4" key="4" url="4.xml">4.test4</option>
</select>
</xml>
```

この例は、以下の選択画面を示しています。

1. test1
2. test2
3. test3
4. test4

例 1.22

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<select mode="1" size="1" starty="2" endy="5" number="4">
<option          y="1" key="1" url="1.xml">1.test1</option>
<option x="2" y="2" key="2" url="2.xml">2.test2</option>
<option x="2" y="3" key="3" url="3.xml">3.test3</option>
<option x="2" y="4" key="4" url="4.xml">4.test4</option>
</select>
</xml>
```

1. test1
2. test2
3. test3
4. test4

x 座標又は y 座標の指定がなければ、option タグがデフォルト値に従いその位置を決定します。

1．7 HINT

【機能】

アラームなどのヒントを定義します。

ブザー、LED、またはバイブレータを使ってユーザに何かを知らせる時に使います。

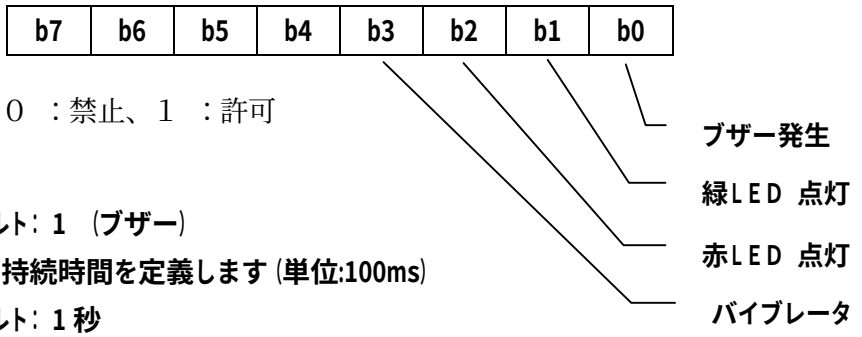
【文法】

<hint [method=number] [time=number] [delay=number] [count=number] [name=string]></hint>

【パラメータ説明】

method ヒントの方式を定義します

整数値の下位4ビットを使って下図のようにヒントの方式およびその組 4 み合わせを表します。



デフォルト：1 (ブザー)

time ヒントの持続時間を定義します (単位:100ms)

デフォルト：1 秒

delay ヒントの間隔時間を定義します (単位:100ms)

デフォルト：500 ms

count ヒントの繰り返し回数を定義します

デフォルト：3

name ヒントの名前を定義します

Input ラベルにおいて、ヒントを引用する時に使います

(デフォルト無し)

【数的制約】

一つのページに設定できる Hint コントロールの数は 40 個までです。

【特記事項】

- INPUTラベルにおいてgoodhint属性またはbadhint属性の値としてnameが指定されている場合は、ユーザが入力確定後に該当するヒントが実行されますが、INPUTラベルのgoodhint属性またはbadhint属性として定義されていないnameの場合は、当該ページがロードされ次第そのヒントが実行されます。
- ヒントが明示的に指定されていない場合に入力NGになった時、HTはデフォルトのブザー音を出し

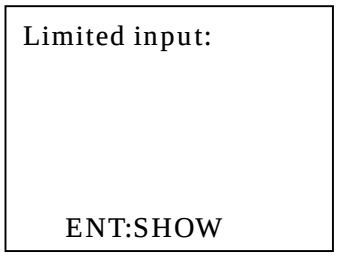
ます。

【サンプルプログラム】

例 1.23

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="2" >limited input:</text>
<input mode="1" size="1" x="16" y="2" format ="1" tip ="1" bar="1" len= "2"
param="limited" min="10" max="25" goodhint="hint2" badhint="hint3"></input>
<button mode="1" size="1" x="5" y="8" key="e" type="2"
url="show.asp">ENT:SHOW</button>
<button mode="1" size="1" x="5" y="8" key="c" type="5" ></button>
<hint method="7" time="5" delay="5" count="3" name="hint1"></hint>
<hint method="2" time="5" delay="5" count="3" name="hint2"></hint>
<hint method="1" time="5" delay="5" count="3" name="hint3"></hint>
</xml>
```

この例の HT 画面は以下の通りです。ページがロードされたら、ブザー音と LED によるヒントがまず発せられます。そしてユーザ入力が正しい場合 (min=10、max=25) は緑の LED を点滅し、正しくない場合はブザー音を鳴らします。



1．8 PRINT

【機能】

印刷機能を実現します。

【文法】

<print url=string type=number pages=number >
  value
</print>

【パラメータ説明】

type プリンタの機種を指定します。
2 バイトから成り、上位バイト（MB）はメイン機種分類、下位バイトはサブ機種分類を指定します。
現在サポートしているプリンタの機種は下記の通り。
0x0101: SATO M3100

url 印刷後のジャンプ先 URL を指定します。

Pages 印刷すべきページ数を指定。印刷スクリプトにあるページ数と一致させなければなりません。（印刷例を参照）

value 印刷内容を定義するスクリプト。プリンタ機種が違えば使用するスクリプトも違います。
詳細は該当プリンタのマニュアルを参照されたい。

【数的制約】

PRINT タグは 1 ページに 1 個しかあってはなりません

【特記事項】

- 印刷後に所定のurlにジャンプする時に、**QueryString**の中で**“printresult”**として印刷結果を渡します。

サンプルプログラム

例 1.24

print.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="8" y="1" >Print</text>
<print url="over.asp" type="257" pages="2">
L01010050300071113 00
L01020050250071113 00
L01030050200151202030
L01040050150071113 00
L01050050100151202030
L01060050050071113 00
D0101IJ  L>
D0102IJ  HV
D0103*123456789*
D0104%m%3%8
D0105123456789
D0106FC  5-
Q0002
C0
</print>
</xml>
```

この例にある”Q0002”が印刷ページ数を示しています。従って pages も 2 に設定してやらなければなりません。

1．9 SETTING

【機能】

HT 内部にパラメータを保存します。

【文法】

<setting name=string value=string validdate=string >
</setting>

【パラメータ説明】

name 保存すべきパラメータの名前を指定します。
value 保存すべきパラメータの値を指定します。
validdate 保存期限を指定します。(現在未使用)

【数的制約】

1ページに設定できる Setting タグの数は 20 個までです。

【特記事項】

- 現在設定できる内部パラメータは以下の通り

RetryTimes	Modemと通信時のリトライ回数
IntervalTime	Modemと通信時のリトライ間隔 (単位はS)
F2URLSetting	Modemとの通信が失敗した場合、HTの内部にF2URLSetting値が設定してあれば、“通信異常”画面 (組込画面) のF2キーを押すと、F2URLSettingが指すURLに、なければ元の画面にジャンプします。
htsessionid	HT電源投入時に自動的に生成されるユニークなID。HTが異なれば生成されるhtsessionidも異なるので、個々のHTを識別するのに使えます。

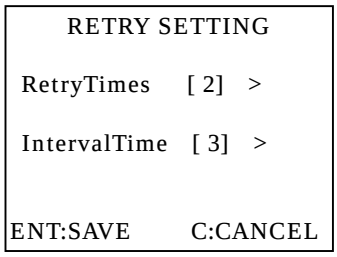
【サンプルプログラム】

例 1.25

setting.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="5" y="1">RETRY SETTING</text>
<setting name="RetryTimes" value="2" validdate=""></setting>
<setting name="IntervalTime" value="3" validdate=""></setting>
<button mode="1" size="1" x="2" y="8" key="e" type="10">
ENT</button>
</xml>
```

この例では、RetryTimes と IntervalTime についてそれぞれ設定を行い、ENT キー (button type=10) を押すと、次のように同パラメータの設定画面 (組込画面) を表示するようになっています。このように、RetryTimes と IntervalTime は、HT 側にて変更することもできます。



第 2 章 動的ページの生成

前章では、Web エミュレータの XML タグを紹介しました。これらのタグを使って必要な XML ページを生成することができます。しかしながら実際のアプリケーションでは、殆どの場合ページの内容が動的に変化するので、静的な XML ページだけですべてのアプリケーションの要求を満たすことはとても不可能です。このため実際のアプリケーションでは、動的な XML ページを作成する必要があります。

動的ページを作成するには、様々なコンピュータ言語を使うことができますが、通常「スクリプト言語」と呼ばれる言語を使います。Web 専用のものだけでも、asp や php、jsp 等が挙げられます。ここでは、asp と php を例に、実際の HT アプリケーションに必要な動的 XML ページを作成する方法を解説します。

2．1 ASP&PHP

ASP 又は PHP を用いて HT の動的ページを生成することは、即ち ASP 又は PHP を用いて DB アクセスを行ったり、論理演算を行ったりして、その結果を XML タグの文法に則って編集し、Web ページとして出力することです。

2.1.1 簡単な XML ページの生成例

ASP の Response.write を用いて簡単なページを出力することができます。次に示す通りです。

```
<%
response.write "<?xml version=""1.0"" enc<button mode="1" size="1" x="14" y="8"
key="e" type="1" url="ite "<xml cache=""1"" bitmap=""1"">"&vbCrLf
response.write "<text mode=""1"" size=""1"" x=""5"" y=""1"" >Hello,
world!</text>"&vbCrLf
response.write "</xml>"&vbCrLf
%>
```

この例では、“Hello,world!”という文字列を出力しています。正確に言うと、この文字列を出力する XML ページを出力しているのです。
【注意】ASP では””で一つの”を表現しています。また、行末には必ず vbCrlf という改行記号を付けなければなりません。

一方、PHP の場合は print 文を使えば上記と同様のページを出力することができます。次に示す通りです。

```
<?php
print "<?xml version=¥"1.0¥" encoding=¥"ISO-8859-1¥"
standalone=¥"yes¥"?>¥n";
print "<xml cache=¥"1¥" bitmap=¥"1¥">¥n";
print "<text mode=¥"1¥" size=¥"1¥" x=¥"5¥" y=¥"1¥" >Hello, world!</text>¥n";
print "</xml>¥n";
?>
```

【注意】PHP の print 文で”を出力したい場合はその前に\を付ける必要があります。また行末には \n という改行記号を付けなければなりません。

なお、XML ページの中に&を出力する場合は、同様に amp を使います。例えば、

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="5" y="5" key="e" type="1"
url="test.asp?a=1&amp;b=2 &amp;c=3">TEST</button>
</xml>
```

2.1.2 セッション管理について

Web エミュレータ Ver1.x.x からは、アプリケーション側のセッション管理機能をサポートしています。セッション管理とは、HT 毎の変数データを区別して管理することです。以下、ASP のセッション管理機能を使って HT 毎の変数データを管理する例を示します。

(セッション管理機能利用の例)

index.asp:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="2" y="1" >Session Test Page 1</text>
<input mode="1" size="1" x="10" y="3"  format ="1"  tip ="1" bar="1" len= "10"
param="value" ></input>
<button mode="1" size="1" x="2" y="8" key="a" type="1" url="
showvalue.asp">F1:Show</button>
<button mode="1" size="1" x="12" y="8" key="e" type="2"
url="savevalue.asp">ENT:Submit</button>
</xml>
```

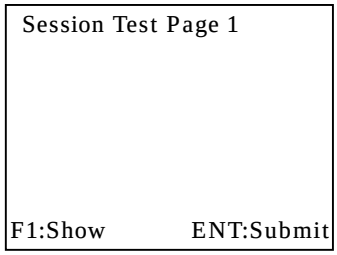
savevalue.asp:

```
<%
value = Request("value")
session("value") = value
%><?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="2" y="1" >Session Test Page 2</text>
<text mode="1" size="1" x="5" y="3" >Value : <%=value%></text>
<button mode="1" size="1" x="2" y="8" key="b" type="1" url="index.asp"
>F2:BACK</button>
<button mode="1" size="1" x="14" y="8" key="e" type="1" url=" showvalue.asp"
>ENT:SHOW</button>
</xml>
```

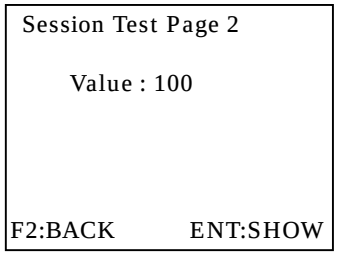
showvalue.asp:

```
<%
value = session("value")
%><?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="2" y="1" >Session Test Page 3</text>
<text mode="1" size="1" x="5" y="3" >Value : <%=value%></text>
<button mode="1" size="1" x="2" y="8" key="b" type="1" url="index.asp"
>F2:BACK</button>
<button mode="1" size="1" x="14" y="8" key="e" type="1" url="
showvalue.asp">ENT:OK</button>
</xml>
```

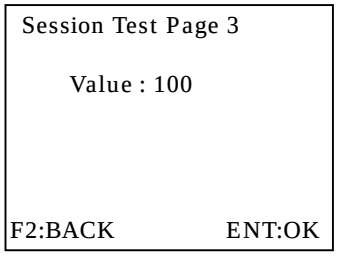
ここで、ある HT（例えば IP1）を使って index.asp を実行すると、下記画面が得られます。



上記画面にて 100 を入力し、ENT キーにて SUBMIT すると、savevalue.asp が呼び出され、下記画面が表示されます。

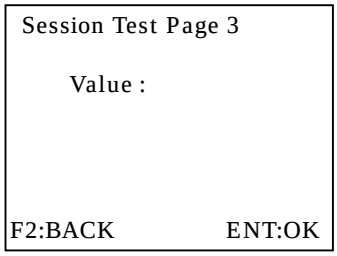


上記画面で表示されている 100 は QueryString の中から取り出したものです。例では更にそれを Session("value")の中に書き込んでいます。ここで ENT キーを押すと、下記画面が表示されます。



この例で分かるように、100 という変数データが IP1 という HT に属するセッションの中で正しく管理されています。

では、ここで別の HT（例えば IP2）を使って index.asp を実行したらどうなるかを見てみましょう。index.asp の画面にて F1 を押すと、下記のような画面が表示されるはずです。



上記画面では、Value の値がありません。なぜなら、100 という変数データが IP1 という HT で入力したものなので、その HT のセッションの中でしか有効でないからです。

この例で分かるように、HT 毎のセッション管理を正しく行うことができます。

ただ、セッション管理機能を使わなくても、HT 毎の変数データを区別して管理することができます。それは QueryString を利用する方法です。以下、その例を示します。（上記例を QueryString を利用する形で書き換えた例です。）

（QueryString 利用の例）

index.asp:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="2" y="1" >Session Test Page 1</text>
<input mode="1" size="1" x="10" y="3"  format ="1"  tip ="1" bar="1" len= "10"
param="value" ></input>
<button mode="1" size="1" x="2" y="8" key="a" type="1"
url="showvalue.asp">F1:Show</button>
<button mode="1" size="1" x="12" y="8" key="e" type="2"
url="savevalue.asp">ENT:Submit</button>
</xml>
```

savevalue.asp:

```
<%
value = Request("value")
%><?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="2" y="1" >Session Test Page 2</text>
<text mode="1" size="1" x="5" y="3" >Value : <%=value%></text>
<button mode="1" size="1" x="2" y="8" key="b" type="1" url="index.asp"
>F2:BACK</button>
<button mode="1" size="1" x="14" y="8" key="e" type="1"
url="showvalue.asp?value=<%=value%>" >ENT:SHOW</button>
</xml>
```

showvalue.asp:

```
<%
value = Request("value")
%><?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="2" y="1" >Session Test Page 3</text>
<text mode="1" size="1" x="5" y="3" >Value : <%=value%></text>
<button mode="1" size="1" x="2" y="8" key="b" type="1" url="index.asp"
>F2:BACK</button>
<button mode="1" size="1" x="14" y="8" key="e" type="1"
url="showvalue.asp">ENT:OK</button>
</xml>
```

実際、どちらの方法を利用して HT 毎の変数データを区別して管理するかは、アプリケーション次第です。

一方、Web エミュレータが htsessionid、htguid 及び modemguid といった Querystring 変数パラメータを出力しています。アプリケーションは、htsessionid に基づいて個々の HT をユニークに識別することができます。また、htguid により当該 url がどの HT から、modemguid によりどの modern を経由して送られてきたのかを知ることができます。それぞれのパラメータは、以下のようなルールで生成されます。

htsessionid(19 桁) = 時刻 (YYMMDDHHMMSS) +m_gid(2 桁)+htguid(4 桁)+ﾊﾞｯﾃﾘ残量 (1 桁)
htguid = ht_gid*100+ht_id
modemguid=modem_gid*100+modem_id

以下、これらのパラメータを取得する例を示します。

(ASP の例)

index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="5" y="5" key="e" type="1"
url="showid.asp">ENT:SHOWID</button>
</xml>
```

showid.asp:

```
<%
guid = request.QueryString("htguid")
modemguid= request.QueryString("modemguid ")
htsessionid = request.QueryString("htsessionid ")
response.Write "<?xml version=""1.0"" encoding=""ISO-8859-1""
standalone=""yes""?>"&vbCrLf
 response.Write "<xml cache=""1"" bitmap=""1"">"&vbCrLf
 response.Write "<text mode=""1"" size=""1"" x=""1""
y=""1"">HTGUID:"&guid&"</text>"&vbCrLf
 response.Write "<text mode=""1"" size=""1"" x=""1""
y=""2"">MODEMGUID:"&modemguid&"</text>"&vbCrLf
 response.Write "<text mode=""1"" size=""1"" x=""1""
y=""4"">HTSESSIONID:</text>"&vbCrLf
 response.Write "<text mode=""1"" size=""1"" x=""1""
y=""5"">"&htsessionid&"</text>"&vbCrLf
 response.Write "</xml>"&vbCrLf
%>
```

この例で分かるように、asp の場合は request.Querystring()を使って url の中の変数データを取得しています。

(PHP の例)

index.xml:

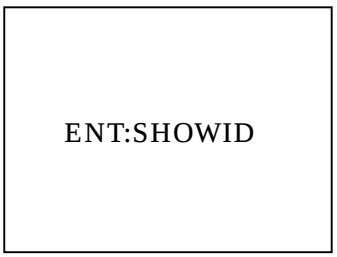
```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<button mode="1" size="1" x="5" y="5" key="e" type="1"
url="showid.php">ENT:SHOWID</button>
```

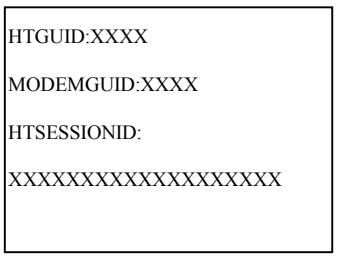
showid.php:

```
<?php
$htguid = $_GET['htguid'];
$modemguid = $_GET['modemguid'];
$htsessionid = $_GET['htsessionid'];
 print "<?xml version=¥"1.0¥" encoding=¥"ISO-8859-1¥"
standalone=¥"yes¥"?>¥n";
 print "<xml cache=¥"1¥" bitmap=¥"1¥">¥n";
 print "<text mode=¥"1¥" size=¥"1¥" x=¥"1¥"
y=¥"1¥">HTGUID:{$htguid}</text>¥n";
 print "<text mode=¥"1¥" size=¥"1¥" x=¥"1¥"
y=¥"2¥">MODEMGUID:{$modemguid}</text>¥n";
 print "<text mode=¥"1¥" size=¥"1¥" x=¥"1¥"
y=¥"4¥">HTSESSIONID:</text>¥n";
 print "<text mode=¥"1¥" size=¥"1¥" x=¥"1¥"
y=¥"5¥">{$htsessionid}</text>¥n";
```

php の場合は、一般的に$_GET[]を用いて url の中の変数データを取得しています。

上記二つの例の動作はまったく同じです。index.xml は“ENT:SHOWID”というボタンを表示して、そのボタンを押すと、当該 HT の HTGUID、MODEMGUID、HTSESSIONID を見ることができます。





2.1.3 BUTTON による動的ページの SUBMIT

type=2,6 の button を含むページを submit する時に、当該ページ内の input タグの入力データはそれぞれの param 属性が指す変数名に従って QueryString に付加されます。

(ASP の例)

index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1" >input:</text>
<input mode="1" size="1" x="8" y="1"  format ="1"  tip ="1" bar="1" len= "10"
param="number"></input>
<button mode="1" size="1" x="5" y="8" key="e" type="2"
url="show.asp">ENT:SHOW</button>
</xml>
```

show.asp:

```
<%
  number = request.QueryString("number")
 response.Write "<?xml version=""1.0"" encoding=""ISO-8859-1""
standalone=""yes""?>"&vbCrLf
 response.Write "<xml cache=""1"" bitmap=""1"">"&vbCrLf
 response.Write "<text mode=""1"" size=""2"" x=""5""
y=""2"">number="&number&"</text>"&vbCrLf
response.Write "</xml>"&vbCrLf
%>
```

この例で分かるように、index.xml では button type=2 となっているので、それを show.asp に submit する時に、入力内容が number という変数名にて QueryString の中に付加されます。一方、show.asp では、request.QueryString を使って number の内容を取り出しています。

(PHP の例)

index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="1" y="1" >input:</text>
<input mode="1" size="1" x="8" y="1"  format ="1"  tip ="1" bar="1" len= "10"
param="number"></input>
<button mode="1" size="1" x="5" y="8" key="e" type="6"
url="show.php">ENT:SHOW</button>
</xml>
```

show.php:

```
<?php
 $number = $_GET['number'];
 print "<?xml version=¥"1.0¥" encoding=¥"ISO-8859-1¥"
standalone=¥"yes¥"?>¥n";
 print "<xml cache=¥"1¥" bitmap=¥"1¥">¥n";
 print "<text mode=¥"1¥" size=¥"2¥" x=¥"5¥" y=¥"2¥">number={$number}
</text>¥n";
  print "</xml>";
```

この例は上記 asp の例と似ています。button type=6 の場合も index.xml の中の入力データは show.php に渡されます。

2.1.4 PRINT による動的ページの SUBMIT

印刷完了後所定の url にジャンプする時に、QueryString の中に printresult という変数名にて印刷結果をサーバーに渡します。

(ASP の例)

index.xml:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<xml cache="1" bitmap="1">
<text mode="1" size="1" x="8" y="1" >Print</text>
<print url="show.asp" type="257" pages="2">
L01010050360011223 00
L01020050300071113 00
L01030050250071113 00
L01040050201512020 30
L01050050150071113 00
L01060050100151202030
L01070050050071113 00
D0101 撞・ o　XXX
D0102IJ　L>
D0103IJ　HV
D0104*123456789*
D0105%m%3%8
D0106123456789
D0107FC　5-
Q0002
C0
</print>
</xml>
```

show.asp:

```
<%
 printresult = request.QueryString("printresult")
 response.Write "<?xml version=""1.0"" encoding=""ISO-8859-1"" standalone=""yes""?>"&vbCrLf
 response.Write "<xml cache=""1"" bitmap=""1"">"&vbCrLf
 response.Write "<text mode=""1"" size=""2"" x=""5"" y=""2"">printresult ="&printresult&"</text>"&vbCrLf
response.Write "</xml>"&vbCrLf
%>
```

(**PHP の例**)

show.php:

```
<?php
 $ printresult = $_GET[' printresult '];
 print "<?xml version=¥"1.0¥" encoding=¥"ISO-8859-1¥"
standalone=¥"yes¥"?>¥n";
 print "<xml cache=¥"1¥" bitmap=¥"1¥">¥n";
 print "<text mode=¥"1¥" size=¥"2¥" x=¥"5¥" y=¥"2¥"> printresult
={$ printresult } </text>¥n";
  print "</xml>;"
```