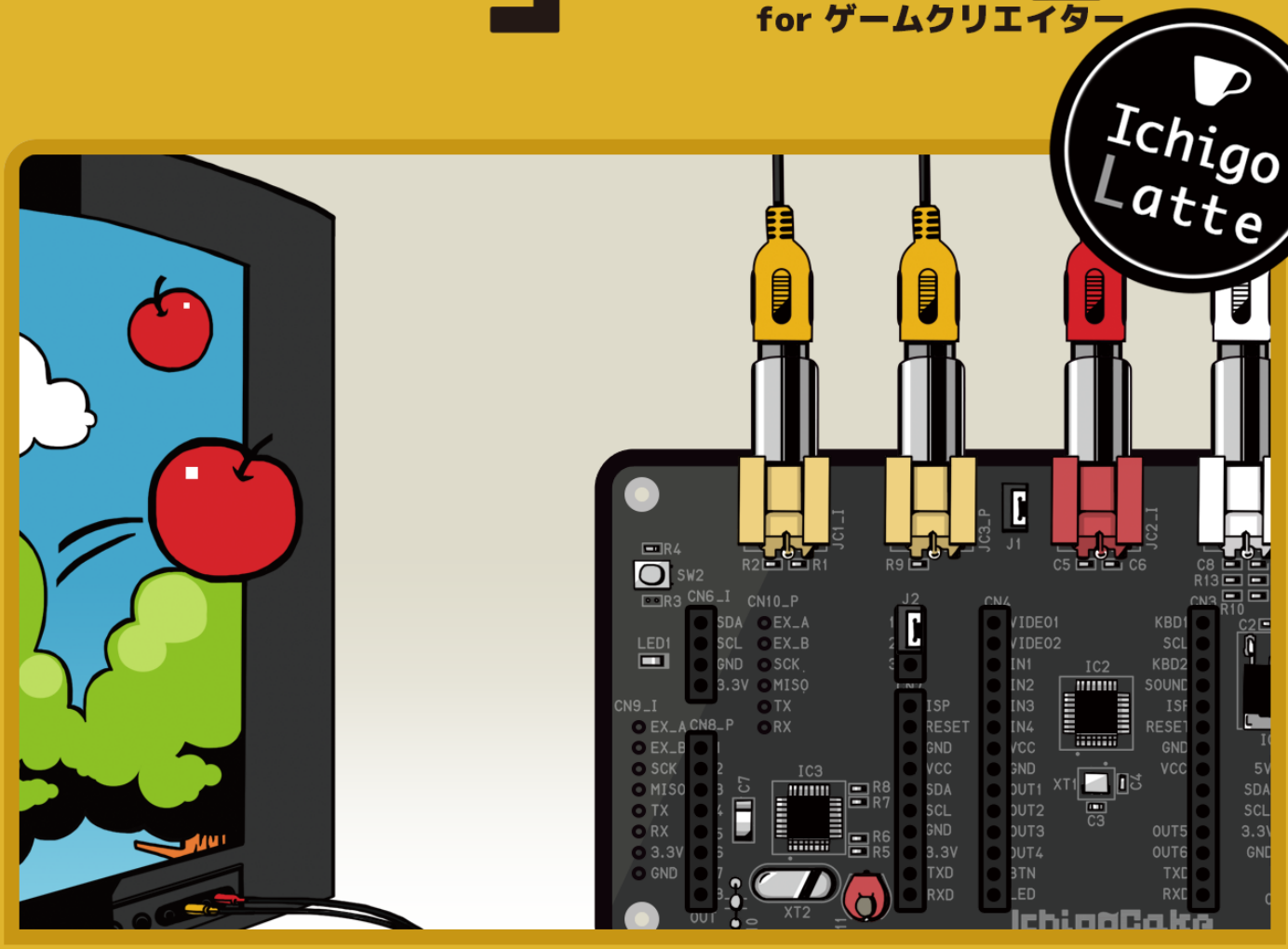
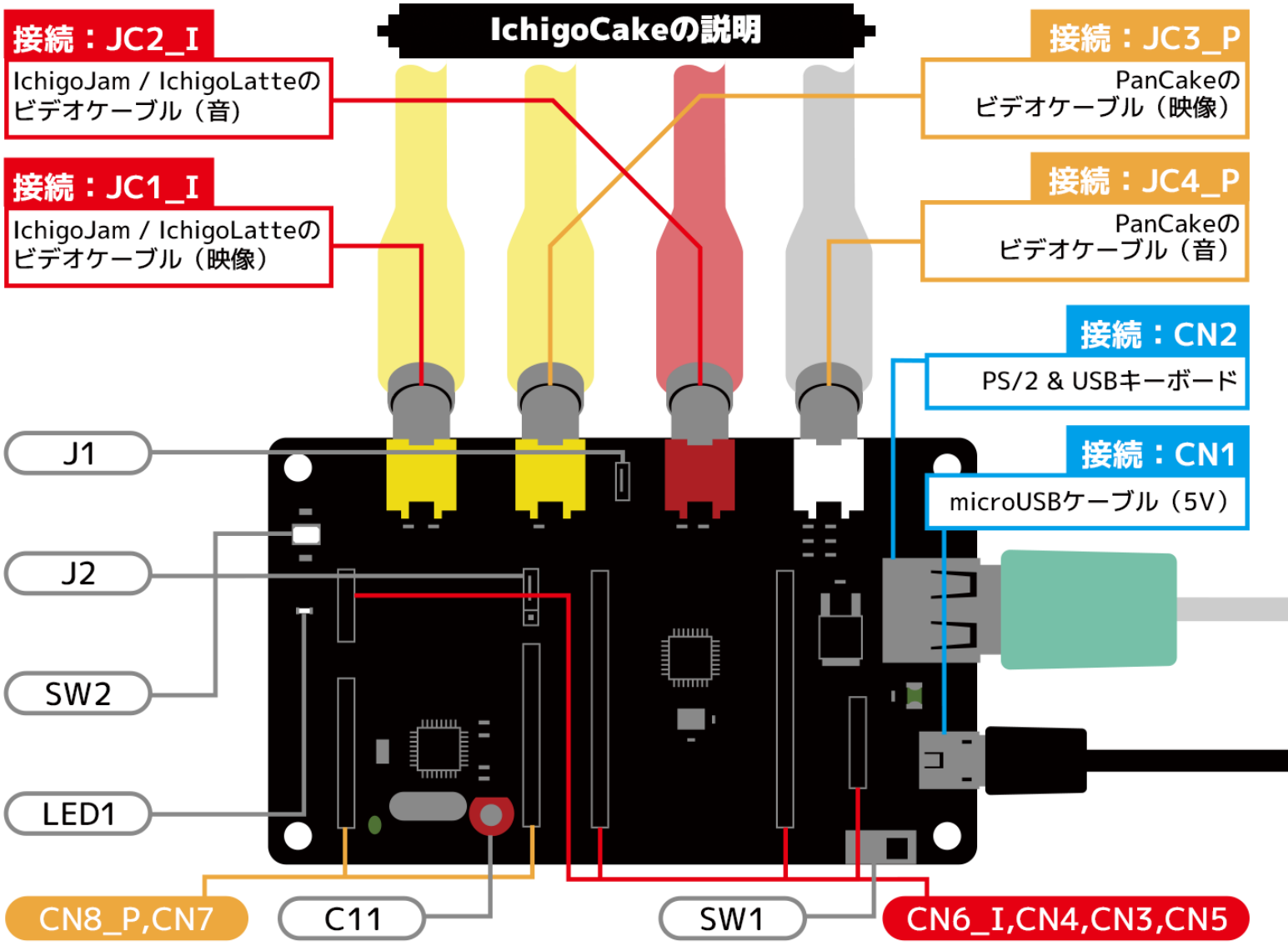
IchigoCake

for ゲームクリエイター



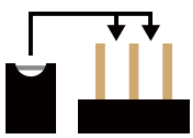
PCN

IchigoCakeの説明



J1	IchigoJam / IchigoLatteのTXと PanCakeのRXを接続します。(※1) ※1 ジャンパーピンを外すとPanCakeは使えません。
J2	ビデオ信号を切り替えます。
SW2	Jam　：BTN()で値[0/1]が取得できます。 Latte：BTN()で値[0/1]が取得できます。
LED1	Jam　：LED1で点灯、LED0で消灯します。 Latte：led(1)で点灯、led(0)で消灯します。
CN8_P,CN7	PanCakeのピンソケット
CN6_I,CN4,CN3,CN5	IchigoJam / IchigoLatteの ピンソケット
C11	PanCakeの映像調整に使います。(※2) ※2 起動時に画面がカラーにならない場合、 内部を精密ドライバーなどで回して調整してください。
SW1	電源をON / OFFします。

J2について：



ジャンパーピンを挿す位置によって、
IchigoJam/IchigoLatteとPanCakeの映像出力を
操作できます。

[ジャンパーピンを挿す位置:1-2]

JC1-IからIchigoJam/IchigoLatte、
JC3-PからPanCakeの映像を出力します。
JC1-I、JC3-Pを
1台のモニタで差し替えるか、
2台のモニタにそれぞれ接続してください。

[ジャンパーピンを挿す位置:2-3]

JC1-IからIchigoJam/IchigoLatte、
PanCakeの両方の映像を出力します。
1台のモニタに接続し、下記コマンドを
プログラムの最初と最後の行に入れる
などして切り替えてください。

- IchigoJam→PanCake
  VIDEO 0:?"PC VIDEO 01"
- PanCake→IchigoJam
  VIDEO 1:?"PC VIDEO 00"

- IchigoLatte→PanCake
  video(0);uart("PC VIDEO 01\n");
- PanCake→IchigoLatte
  video(1);uart("PC VIDEO 00\n");

IchigoLatte ECMAScript

[Implemented]

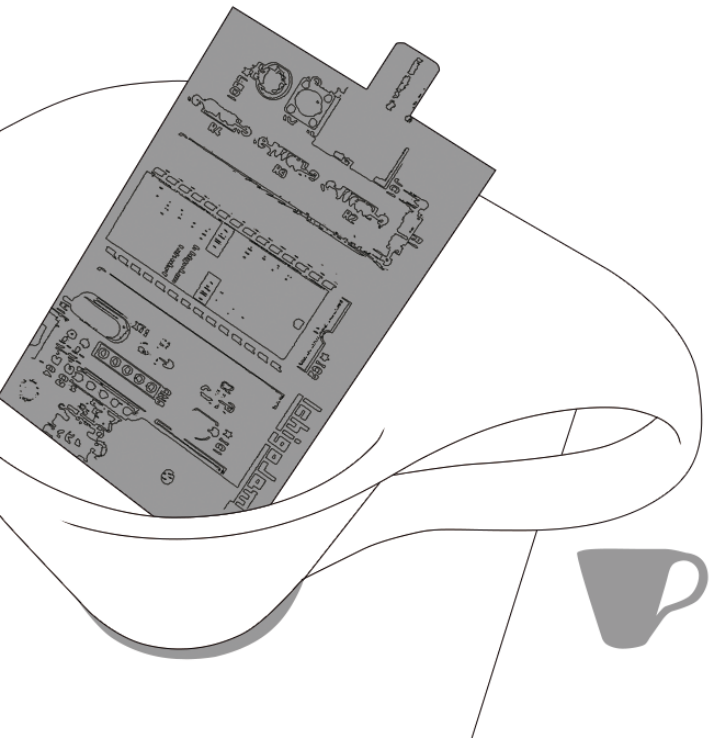
* var
* function, return
* if-else
* while, break
* number literal
* string("") literal
* =, +, -, *, /, ==, !=, <, >, %, <=, >=, !
* , ;
* { }
* new, this
* //
* +, - as immediate value
* multiple var
* try-catch, throw
* >>,<<,&,|,^,~

[Global object: latte]

* "latte."は省略する必要があります。
* 'latte' members
  * log(v, ...)
  * led(v)
  * btn()
  * gpin(v)
  * out(v)
  * cls()
  * lc(x, y)
  * sleep(v)
  * inkey()
  * rnd(v)
  * chr(v)
  * sin8(v)
  * tick()
  * ana(p)
  * scr(x, y)
  * pwm(p, d)
  * pwmt(t)
  * scroll(v)
  * video(v)
  * setTout(f, d)
  * setKprs(f)
  * setBprs(f)
  * input()
  * bps(v)
  * uart()
  * i2cw(a, ...)
  * i2cr(a, ...)
  * exit(v)
  * lrun(n)
  * env(n)
  * mem()
  * ver()
  * beep()
* 'Array' members
  * length

[NOTES]

* new された object が持てるメンバは最大8つです。
* 変数に関数の代入可能です。
* 深刻な構文エラー、実行時エラーなどはLED点滅後ハードリセットします。
* Number型は32bit整数です。
* メソッドチェーンが動きます。
* setTimeout的な仕組みの setTout あります。
* try-catch, throw があります。
* 配列があります。
* 無名関数が動きます。
* GPIO,UART,I2C を用いたハード制御が可能です。
* メモリを直接読み書きできます。



IchigoLatte

IchigoLatte reference 1.0.1

IchigoLatte	- キッズOS * ScrLock テレビモニターがうまく映らないときはNTSCモードを変更してみる。 ビデオがオフのときにオンにすることも可能。 * Ctrl+Alt+Del OSを再起動する。
lash	- latteシェル * .(no) 編集可能なファイルI/O。 no: 0〜3（省略すると0） * uart UARTシリアルI/O。 * @(no) EEPROMファイルI/O。 no: 0〜63（1Mビットの場合） * 例） >cat .0 > uart >cat uart > .0 >cat @0 > .0 >cat .0 > @63 + ボタンを押した状態でシステムを起動すると"ms ."が実行される。 + ボタンを押すかCtrl+Dを入力するとモード（コンソール⇔UART）が 切り替わる。（UARTモード時はviコマンドは使用不可）
echo	- 引数の内容を出力する。 * 例） >echo Hello World.
cat	- ファイルの内容を出力する。 * 'ESC' catを終了する。 * 例） >cat .0 >cat uart
ls	- ファイルの中身を一覧表示する。 * 例）EEPROM >ls * 例）.ファイル >ls .
vi	- テキストエディタを起動する。 * 'ESC' 内容を保存してviを終了する。 * Ctrl+D 変更を破棄してviを終了する。 * Ctrl+C カーソル位置の行をコピーする。 * Ctrl+V コピーした内容を貼り付ける。 * Ctrl+X カーソル位置の行を切り取る。

ms	- JavaScriptを実行する。 * 例）対話モード >ms * 例）実行モード >ms .0 - [latteオブジェクト メンバ関数] * log(v, ...) 文字を表示する。 * led(v) v==0：LEDをOFF v==1：LEDをON * btn() ボタンの状態を0もしくは1で返す。 * gpin(v) INポートの状態を0もしくは1で返す。vはポート番号。 * out(v) OUTポートの状態を変更する。 vのビットフィールドは各ポート番号に対応する。 * cls() 画面をクリアする。 * lc(x, y) カーソルをx,yに移動する。 * sleep(v,[d]) vミリ秒間プログラムを停止する。 dを指定すると抵電力化。 * inkey() 最後に入力されたキーボードのコードを返す。 * rnd(v) 0からv未満の整数をランダムに返す。 vは整数であること。 * chr(v) 文字コードvから文字を返す。 * sin8(v) 角度vのサインを整数（0～256）で返す。（vはディグリー角） * tick() システム起動からの経過時間（ms）を返す。 * ana(p) ポートのアナログ値を返す。 p==0：ボタン p==2：IN2 p==5-8：OUT1～4 * scr(x, y) 画面上の指定座標に書かれた文字コードを返す。 * pwm(p, d) OUTポートpにPWM信号を送る。引数dはパルス幅（μs）。 * pwmt(t) PWM周期（μs）を設定する。[デフォルト:20,000]

- [latteオブジェクト メンバ関数]
  * scroll(v, [r])
    指定した方向に画面をスクロールする。
    v==0：上
    v==1：右
    v==2：下
    v==3：左
    rでローテートを指定する。
    r==0：ローテートしない[デフォルト]
    r==1：ローテートする

  * video(v)
    NTSC信号を切り替える。
    v==0：OFF
    v==1：ON
    v==2：ON（反転）

  * setTout(f, d)
    dミリ秒後に関数fを呼び出す。
    例）
    ```
    function onTime(){...}
    setTout(onTime, 1000);
    ```

  * setTout()
    setTout(f, d)で設定した内容を解除する。

  * setKprs(f)
    キーが押された時に関数fを呼び出す。
    例）
    ```
    function onKey(k){...}
    setKprs(onKey);
    ```

  * setKprs()
    setKprs(f)で設定した内容を解除する。

  * setBprs(f)
    ボタンを押した時/離した時に関数fを呼び出す。
    例）
    ```
    function onBtn(b){...}
    setBprs(onBtn);
    ```

  * setBprs()
    setBprs(f)で設定した内容を解除する。

  * input()
    キーボードからの入力を数値で返す。

  * bps(v)
    uartの伝送速度を設定する。[デフォルト:115,200]

  * uart(v, ...)
    uartにvを書き出す。
    第一引数のvが数値の場合、第二引数以降の数値を1バイトのバイナリとして送信する。
    第一引数のvが文字列の場合、第二引数以降の数値を文字列に変換して送信する。

  * uart()
    引数を省略した場合は受信したデータを数値で返す。
    受信するデータがない場合は-1を返す。

  * i2cw(a, ...)
    指定したアドレスaのI2Cデバイスにデータを書き込む。
    引数"..."の内容がデバイスに送信される。

ms

- [latteオブジェクト メンバ関数]
  * i2cr(a, ...)
    指定したアドレスaのI2Cデバイスからデータを読み込む。
    関数から返されると、引数"..."に値がセットされる。

  * exit(v)
    プログラムを終了する。

  * lrun(n)
    EEPROM'@n'の内容を'.'へ上書きし、実行する。
    備考:
    | この関数を実行すると'.'の内容が消去される。

  * env(n)
    環境変数にnをセットする。
    環境変数は他のプログラムから使用できる。

  * env()
    環境変数を返す。

  * mem(v)
    vが数値の場合、そのアドレスのバイトを返す。
    vが文字列の場合、次のアドレスを返す。
    v=="f"：フォントデータのアドレス。(読み出し専用, 8バイト * 256文字)
    v=="." ：ファイルアドレス。(読み出し専用, 2KB)
    v=="s"：描画領域のアドレス。(読み出し/書き込み, 32*24バイト)
    v==" " ：フリーRAMのアドレス。(読み出し/書き込み, スタックの頂上まで使用可能)
    例)　| var fm = mem("f"); // フォントデータのアドレスを取得
    　　| log(mem(fm+(8*0x41)+0)); // フォント'A'(0x41)の0行目

  * mem(a, v, ...)
    アドレスaのメモリにvを書き出す。
    例)　| var sm = mem("s"); // 描画領域のアドレス
    　　| mem(sm, 0x41); // 画面(0,0)に'A'を出力

  * mem()
    フリーRAMにあるARM Thumbコードを呼び出す。
    R0レジスタの値が返される。
    例)　| var free = mem(" "); // フリーRAMのアドレスを取得
    　　| mem(free, 0x70, 0x47); // 処理を返すだけ
    　　| mem(); // コードを呼び出す

  * ver()
    IchigoLatteのバージョンを取得する。
    新しいバージョンほど番号が大きくなる。

  * beep([hz],[t])
    周波数nzのビープ音をtミリ秒間鳴らす。

  * include(n)
    ‘.n’を読み込む。

- [Arrayメンバ]
  + 配列を作成することができる。（1つのみ）

  * Array(v)
    長さvの配列を作成する。
    例)　| var a = new Array(10);

  * Array()
    最大の長さを持つ配列を作成する。
    例)　| var a = new Array();

  * length
    配列の長さを返す。

ms

-[PanCakeメンバ]
+ PanCakeボードを制御する。

* PanCake()
PanCakeオブジェクトを作成する。
例)
| var pc = new PanCake();

* clear(cn)
画面を消去する。
cnには色番号を指定する。

* line(x1, y1, x2, y2, cn)
線を引く。

* circle(xc, yc, ra, cn)
円を描く。
xc,ycには中心座標を指定する。
raには半径を指定する。

* stamp(px, py, tc, "cc...c")
8x8の絵を描く。
px,pyには出力座標を指定する。
tcには透明色を指定する。
cc...cには左上から右方向への色番号を羅列する。

* stamp1(px, py, cn, "cccccccccccccccc")
8x8の一色絵を描く。
px,pyには出力座標を指定する。
cnには色を指定する。
ccccccccccccccccには左上から右方向への0,1を羅列する。
0は透明になる。

* stamps(px, py, si)
stamps(px, py, si, fs)
stamps(px, py, si, fs, ra)
px,pyには出力座標を指定する。
si には組み込みスプライト画像番号を指定する。
fs を指定すると左右反転の制御ができる。ON/OFF[1/0] (Default:0)
ra を指定すると角度の制御ができる。[0:0°,1:-90°,2:180°,3:90°] (Default:0)

* image(in)
組み込み画像を描く。

* video(os)
ビデオ出力をON/OFF[1/0]する。

ms	* sStart(in) スプライト処理を開始する。 in には背景に使う組み込み画像番号を指定する。 背景をベタ塗りにする場合は in のHighBitsを1にする。[ex. 0x12 == 背景赤] スプライト処理を開始すると"line"や"image"などは使えない。 スプライト処理を停止するには in を 0xFF にする。 * sCreate(sn, si) スプライトを作る。 スプライトは32枚用意されている。 sn はスプライト番号 0～31。 si には組み込みスプライト画像番号を指定する。 スプライトを消すには si を 0xFF にする。 スプライトは番号が大きい方が上に重なる。 * sMove(sn, px, py) スプライトを移動する。 sn は"sCreate"した番号。 px,pyは座標、スプライトの左上が起点。 * sFlip(sn, fs) スプライトを左右反転する。 sn は"sCreate"した番号。 fs は ON/OFF[1/0]。 * sRotate(sn, ra) スプライトを回転する。 sn は"sCreate"した番号。 ra は 角度[0:0°,1:-90°,2:180°,3:90°]。 * sUser(sn, tc, "cc...c") スプライトを自作する。(15個) sn は 番号 0xF0～0xFE。 tcには透明色を指定する。 cc...cには左上から右方向への色番号を羅列する。 * sound(o0, s0, o1, s1, o2, s2, o3, s3) 4ch同時に音を鳴らす。 o0～o3 はオクターブ(0～4～7)、s0～s3 は音程(0～b)。 音程eはノイズ音。 s0～s3 のHighBits4は音色(0～3)。 音を消すには s0～s3 を 0xFF にする。 * sound1(cn, on, sn) 1chのみ音を鳴らす。 * mScore(ch, pn, tt, "mm") MMLをサウンドchに登録する。 ch: チャンネル(0～3) pn: 1:即再生、0:後で再生 tt: テンポ(0～F)+音色 [ex. 0x30 == テンポ3、音色0] mm: MML(MAX:バイナリ変換後64byte)

ms	* mPlay(ss) mPlay(ss, ch) 音楽再生をSTART/STOP[1/0]する。 “mScore”または”mLoad”で4ch分のMMLを先に登録して置き、一気に再生可能。 ch(0～3)を指定すると、そのチャンネルだけ制御できる。 * mLoad(ch, mn) 組み込みMMLをサウンドchに登録する。 mn に組み込みMML番号(0～3)を指定する。 サウンドchを初期状態にしたい場合は mn を 0xFF にする。 * reset() PanCakeを初期状態に戻す。 スプライトや音などがすべて初期状態に戻る。 * out(pf) OUTポートに出力する。 pfにはONにするポート番号を16進数で指定する。 * bps(rrrr) 通信速度を設定する。(Default:115,200) rrrrには速度を指定する。 0を指定すると115,200になる。 * wbuf(on) Wバッファモードを ON/OFF[01/00]する。 ON時に再びONすることでバッファを入れ替えることができる。

PanCake REFERENCE

通常のPanCakeのように、uartでコマンドを送ることもできます。

●Text-Command

- ASCII文字列でコマンドを投げてください。
- 各命令は固定長、大文字、最後にLFが必要、解析可能な命令長は MAX:96(byte) です。

コマンド	解説	例
PANCAKE CLEAR cn	画面を消去します。cnには色番号を16進数で指定します。	PANCAKE CLEAR 0A
PANCAKE LINE x1 y1 x2 y2 cn	線を引きます。	PANCAKE LINE 00 00 10 0A 01
PANCAKE CIRCLE xc yc ra cn	円を描きます。 xc,ycには中心座標を指定します。raには半径を指定します。	PANCAKE CIRCLE 10 10 10 01
PANCAKE STAMP px py tc cc...c	8x8の絵を描きます。px,pyには出力座標を指定します。tcには透明色を指定します。 cc...cには左上から右方向への色番号を羅列します。	PANCAKE CIRCLE 00 08 02 2282828222288822...
PANCAKE STAMP1 px py cn cccccccccccccccc	8x8の一色絵を描きます。px,pyには出力座標を指定します。cnには色を指定します。 ccccccccccccccccには左上から右方向への0,1を羅列します。0は透明になります。	PANCAKE STAMP1 00 08 02 FF00FF00FF00FF00
PANCAKE STAMPS px py si [fs] [ra]	px,pyには出力座標を指定します。si には組み込みスプライト画像番号を指定します。 fs を指定すると左右反転の制御ができます。ON/OFF[01/00] (Default:00) ra を指定すると角度の制御ができます。[00:0°,01:-90°,02:180°,03:90°] (Default:00)	PANCAKE STAMPS 24 12 15 PANCAKE STAMPS 24 12 15 01 02
PANCAKE IMAGE in	組み込み画像を描きます。	PANCAKE IMAGE 02
PANCAKE VIDEO os	ビデオ出力をON/OFF[01,00]します。	PANCAKE VIDEO 00
PANCAKE SPRITE START in	スプライト処理を開始します。in には背景に使う組み込み画像番号を指定します。 背景をベタ塗りにする場合は in のHighBitsを1にします。[ex. 12 == 背景赤] スプライト処理を開始するとLINEやIMAGEなどは使えません。 スプライト処理を停止するには in を FF にします。	PANCAKE SPRITE START 03
PANCAKE SPRITE CREATE sn si	スプライトを作ります。スプライトは16枚用意されています。 sn はスプライト番号 0～15 です。si には組み込みスプライト画像番号を指定します。 スプライトを消すには si を FF にします。スプライトは番号が大きい方が上に重なります。	PANCAKE SPRITE CREATE 00 15
PANCAKE SPRITE MOVE sn px py	スプライトを移動します。 sn は CREATE した番号です。px,pyは座標、スプライトの左上が起点です。	PANCAKE SPRITE MOVE 00 10 10
PANCAKE SPRITE FLIP sn fs	スプライトを左右反転します。sn は CREATE した番号です。fs はON/OFF[01/00]です。	PANCAKE SPRITE FLIP 00 01
PANCAKE SPRITE ROTATE sn ra	スプライトを回転します。 sn は CREATE した番号です。ra は角度[0:0°,1:-90°,2:180°,3:90°]です。	PANCAKE SPRITE ROTATE 00 01
PANCAKE SPRITE USER sn tc cc...c	スプライトを自作します。（2個） sn は番号[FD,FE]です。 tc には透明色を指定します。cc...c には左上から右方向への色番号を羅列します。	PANCAKE SPRITE USER FE 02 2282828222288822...
PANCAKE SOUND o0 s0 o1 s1 o2 s2 o3 s3	4ch同時に音を鳴らします。o0～o3 はオクターブ(0～4～7)、s0～s3 は音程(0～b)です。 音程eはノイズ音です。s0～s3 のHighBits4は音色(0～3)です。 音を消すには s0～s3 を FF にします。	PANCAKE SOUND 04 00 04 04 04 07 04 FF PANCAKE SOUND 04 20 04 24 04 27 04 FF
PANCAKE SOUND1 cn on sn	1chのみ音を鳴らします。	PANCAKE SOUND1 00 04 07
PANCAKE MUSIC SCORE ch pn tt mm	MMLをサウンドchに登録します。ch: チャンネル(00～03)　pn: 01:即再生、00:後で再生 tt: テンポ(0～F)+音色 [ex. 30 == テンポ3、音色0] mm: MML(MAX:バイナリ変換後64byte)	PANCAKE MUSIC SCORE 00 01 30 $CDEFGAB>C
PANCAKE MUSIC PLAY ss [ch]	音楽再生をSTART/STOP[01/00]します。 "MUSIC SCORE"または"MUSIC LOAD"コマンドで4ch分のMMLを先に登録して置き、 一気に再生可能です。ch(00～03)を指定すると、そのチャンネルだけ制御できます。	PANCAKE MUSIC PLAY 01 PANCAKE MUSIC PLAY 01 00
PANCAKE MUSIC LOAD ch mn	組み込みMMLをサウンドchに登録します。mn に組み込みMML番号(00～03)を指定します。 サウンドchを初期状態にしたい場合は mn を FF にします。(from 1.1)	PANCAKE MUSIC LOAD 00 01
PANCAKE RESET	PanCakeを初期状態にもどします。スプライトや音などがすべて初期状態に戻ります。	PANCAKE RESET
PANCAKE OUT pf	OUTポートに出力します。pfにはONにするポート番号を16進数で指定します。	PANCAKE OUT FF
PANCAKE BPS rrrr	通信速度を設定します。（Default:115,200） rrrr には速度を16進数で指定します。0000を指定していると115,200になります。	PANCAKE BPS 2580

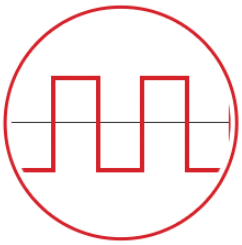
●Binary-Command

- テキストコマンドに対応したバイナリコマンドもあります。詳しくは http://pancake.shizentai.jp/ または FacebookのPanCake-FAN をご参照ください。

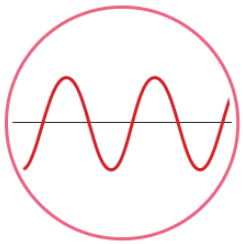
PALETTE

0	黒	0. 0. 0	4	橙	242. 126. 48	8	黄緑	145. 202. 24	c	濃紺	026. 036. 102
1	白	255. 255. 255	5	黄	255. 203. 61	9	緑	28. 77. 55	d	青紫	099. 055. 187
2	赤	227. 27. 41	6	薄茶	255. 222. 169	a	水	67. 175. 215	e	赤紫	178. 063. 171
3	桃	255. 104. 139	7	茶	107. 74. 43	b	青	38. 74. 208	f	灰	204. 204. 204

TIMBRE



0　方形波
Square wave



1　正弦波
Sine wave

2　エイティーズ
80's

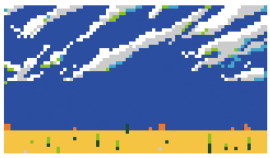
3　ヴァイオリン
Violin

IMAGE

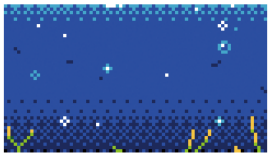
00

01

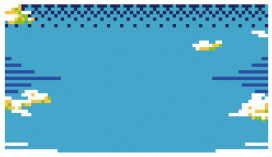
02



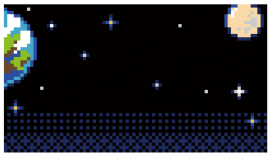
03



04



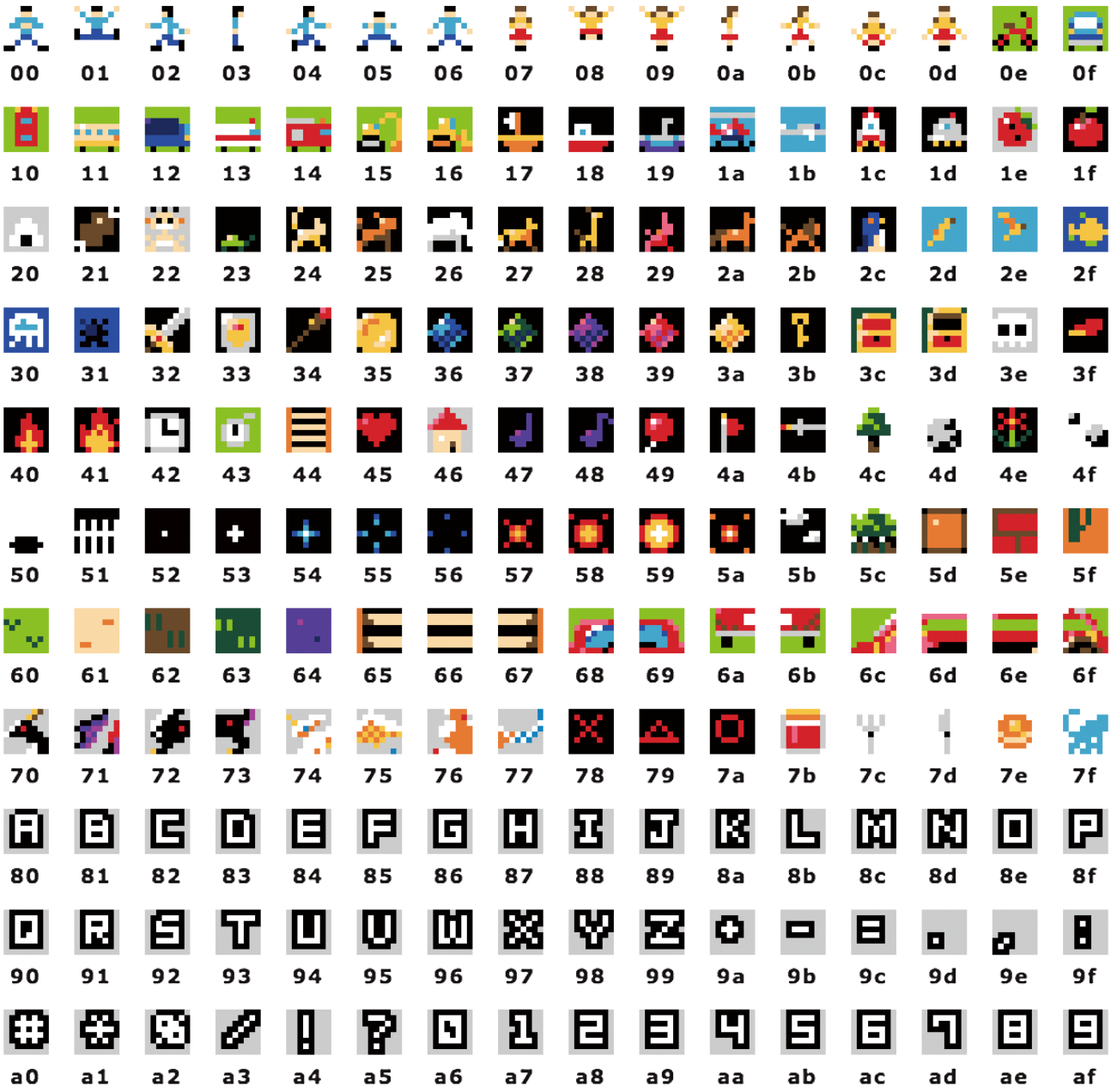
05



06

07

SPRITE


00 01 02 03 04 05 06 07 08 09 0a 0b 0c 0d 0e 0f
10 11 12 13 14 15 16 17 18 19 1a 1b 1c 1d 1e 1f
20 21 22 23 24 25 26 27 28 29 2a 2b 2c 2d 2e 2f
30 31 32 33 34 35 36 37 38 39 3a 3b 3c 3d 3e 3f
40 41 42 43 44 45 46 47 48 49 4a 4b 4c 4d 4e 4f
50 51 52 53 54 55 56 57 58 59 5a 5b 5c 5d 5e 5f
60 61 62 63 64 65 66 67 68 69 6a 6b 6c 6d 6e 6f
70 71 72 73 74 75 76 77 78 79 7a 7b 7c 7d 7e 7f
80 81 82 83 84 85 86 87 88 89 8a 8b 8c 8d 8e 8f
90 91 92 93 94 95 96 97 98 99 9a 9b 9c 9d 9e 9f
a0 a1 a2 a3 a4 a5 a6 a7 a8 a9 aa ab ac ad ae af

通常のPanCakeとの違い

・スプライトを拡張しました
　使用できるスプライト数：32枚
　自作できるスプライト数：15枚

・ダブルバッファリングができます

・別売りのEEPROMを使って、画像と音楽が追加変更できます

通常のPanCakeに加えて、以下のコマンドが追加されています。

●Text-Command
- ASCII文字列でコマンドを投げてください。
- 各命令は固定長、大文字、最後にLFが必要、解析可能な命令長は MAX:96(byte) です。

コマンド	解説	例
PANCAKE WBUF on	Wバッファモードを ON/OFF[01/00]します。 ON時に再びONすることでバッファを入れ替えることができます。 (from 1.2C)	PANCAKE WBUF 01

つくろう！ダンブンゲーム for Latte

```
//danbun kurukuru

var pc = new PanCake();
pc.reset(); sleep(10);

pc.sStart(0x1b);
pc.sCreate(0,0x22).sMove(0,36,20);

var r=0,s=0;

while(1){
   pc.sRotate(0,r);
   sleep(200);

   if(inkey()){
      if(r==0) s=s+1;
      else     s=s-1;
      log("score: ",s,"\n");
   }

   r=(r+1)%4;
}
```

■遊び方
ダンブンが立っているときを狙って
タイミングよくキーボードを押そう！

■改造のヒント
sleep(200) を sleep(100*rnd(10)) にしてみよう
ダンブンのまわる速度がランダムになるよ！

1

```
//dai gyakuso

var pc = new PanCake();
pc.reset(); sleep(10);

pc.mScore(0,0,0x30,"$>ECECECECE
DCDC<G>FC");
pc.mScore(1,0,0x32,"$<CRCRCRCR<
BRBRBRBRARARARARBRBRBRBR");
pc.mScore(2,0,0x33,"$>>GRGRGRGR
GFEFECAG");
pc.mPlay(1);

pc.sStart(2);
pc.sCreate(0,0x10);
pc.sCreate(1,0x0f);
pc.sCreate(2,0xa6).sMove(2,0,0);
pc.sCreate(3,0xa6).sMove(3,8,0);

var rx=0;
var bx=rnd(2),by=0;
var time=0,score=0;

function main(){
  pc.sMove(0,25+23*rx,36);
  pc.sMove(1,25+23*bx,by);

  pc.sCreate(2,score/10+0xa6);
  pc.sCreate(3,score%10+0xa6);

  if((rx==bx)*(27<by)){
    pc.mPlay(0);
    pc.mScore(2,1,0x30,"<ERE~");
    return;
  }
```

```
    if(45<by){
      bx=rnd(2);
      by=0;
    }

    by=by+1;
    time=time+1;
    score=time/10;

    setTout(main,20);
}

function kf(key){
    if(key==28) rx=0;
    if(key==29) rx=1;

    pc.mScore(2,1,0x02,"GAGA");
}

setKprs(kf);
main();
```

■遊び方
← → キーで車を動かして
対向車をよけよう！

■改造のヒント
setTout(main,20) を
setTout(main,20-score/4) にしてみよう
スコアが高くなると対向車のスピードが上がるよ！

つくろう！ダンブンゲーム for Latte

```
//LINGO WO SACCHAN                                    1

var sx=36,f=0;
var bx=sx+7;
var rx=rnd(72),ry=0,rr=0;
var score=0,speed=1;

var pc = new PanCake();
pc.reset(); sleep(10);

pc.mScore(0,0,0x81,"$C>C<EG");
pc.mScore(1,0,0x80,"RRRRR$RRRRR
RRRCCDE~~~~EFED~~~~DEDC~~~~");
pc.mPlay(1);

pc.sStart(1);
pc.sCreate(0,0x5d);
pc.sCreate(1,0x1f);
pc.sCreate(2,0x0b);
pc.sCreate(3,0xa6).sMove(3,0,0);
pc.sCreate(4,0xa6).sMove(4,8,0);

function main(){
  pc.sMove(0,bx,35).sFlip(0,f);
  pc.sMove(2,sx,35).sFlip(2,f);

  pc.sMove(1,rx,ry).sRotate(1,rr);

  rr=(rr+1)%4;
  ry=ry+rnd(speed+1);

  pc.sCreate(3,0xa6+score/10);
  pc.sCreate(4,0xa6+score%10);
```

2

```
   if(35<ry){
      if((bx-4<rx)*(rx<bx+4)){
       score=score+1;
       pc.mScore(3,1,0x22,"">>CE>C");
      }

      if((sx-4<rx)*(rx<sx+4)){
         pc.mPlay(0);
         pc.mScore(3,1,0x40,"<F~ED
~<B>C~~~");
         return;
      }

      rx=rnd(72);
      ry=0;
      speed=(score+10)/10;
   }

   setTout(main,50);
}

function kf(key){
   if(key==28){
      sx=sx-4;
      f=1;
      bx=sx-7;
   }

   if(key==29){
      sx=sx+4;
      f=0;
      bx=sx+7;
   }
}
```

■遊び方
← → キーでさっちゃんを動かして
りんごをカゴに入れよう！
りんごがさっちゃんに当たるとゲームオーバーだよ

■改造のヒント
pc.sCreate(1,0x1f) を
pc.sCreate(1,0x20) にしてみよう
りんごがおにぎりになるよ！