

—目次—

1. 機能概要

2. 動作環境

- (1) ブロック図
- (2) 使用回路図
- (3) 抵抗値について
- (4) それ以外の外部回路
- (5) サンプル回路図

3. 仕様

- (1) 表示可能領域
- (2) 表示可能スプライト数
- (3) U A R T 接続仕様
- (4) 内部レジスタとその制御
- (5) 背景表示器の制御信号について

4. 使用方法

- (1) 初期化
- (2) 座標指定及びビットマップパターン選択
- (3) 背景制御
- (4) その他

5. 注意点など

- (1) 表示方式・信号について (NTSCとの差異など)
- (2) インターレースについて
- (3) カラー信号について
- (4) 信号のタイミングについて
- (5) シンクロ信号について
- (6) 出力電圧について

6. 付録

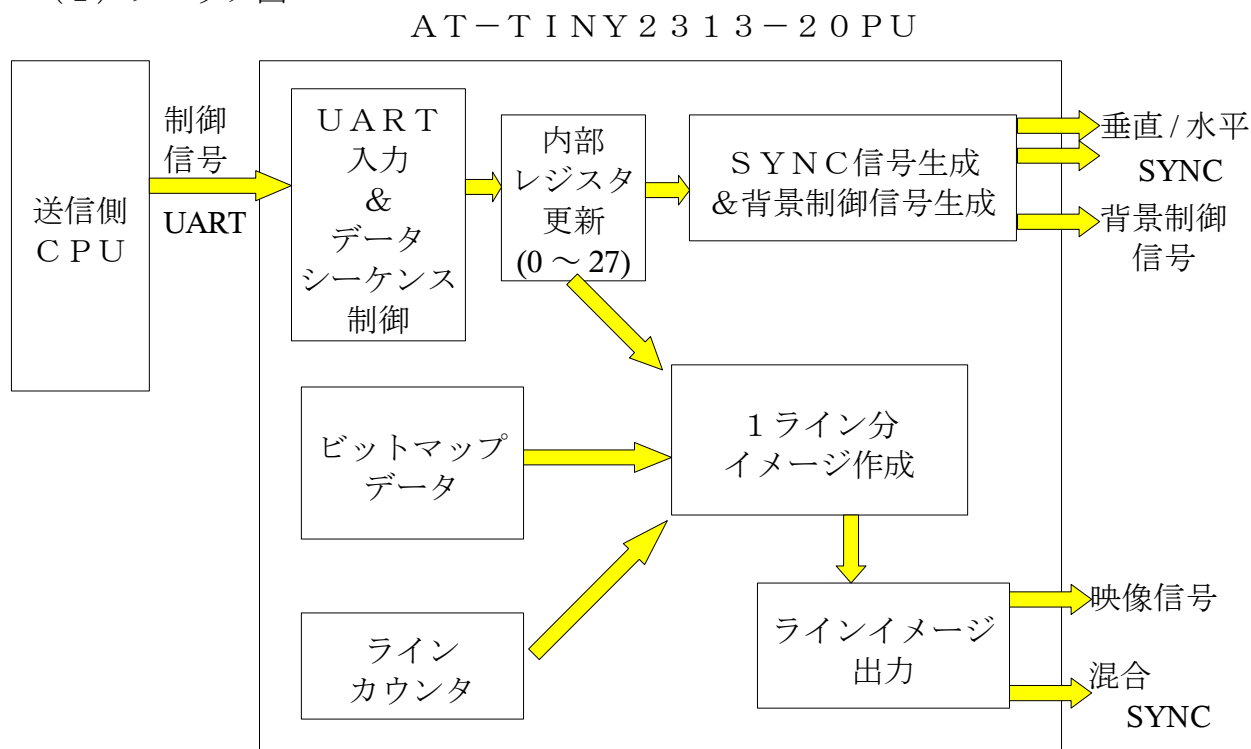
- (1) スプライトビットマップ書き換えツールについて
- (2) デフォルトのビットマップについて

1. 機能概要

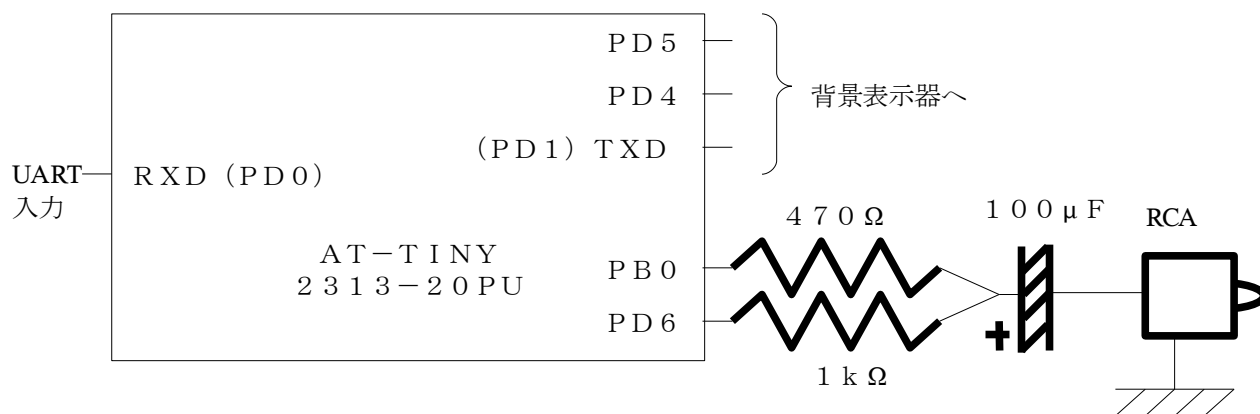
- U A R T信号からスプライトの座標及びビットマップ番号を入力し、N T S Cコンポジット信号（の類似信号）を生成、出力する。
- スプライト1個のサイズは8ドット×8ドット。白黒のみの表示。100種のビットマップをプログラムROM上に格納。（実行中に動的に書き換えることはできない）
- スクリーン座標系は、左上（0，0）～右下（151，113）の範囲。このうち外側の8ドット分を除いた（8，8）～（143，105）の範囲が表示される。
- 同時表示可能なスプライト数は9個。この9個すべてを横1ライン上に同時表示することが可能。
- 背景表示器との接続用として、垂直同期信号、水平同期信号、U A R Tによる背景制御信号の3つの信号を出力する。

2. 動作環境

(1) ブロック図



(2) 使用回路図

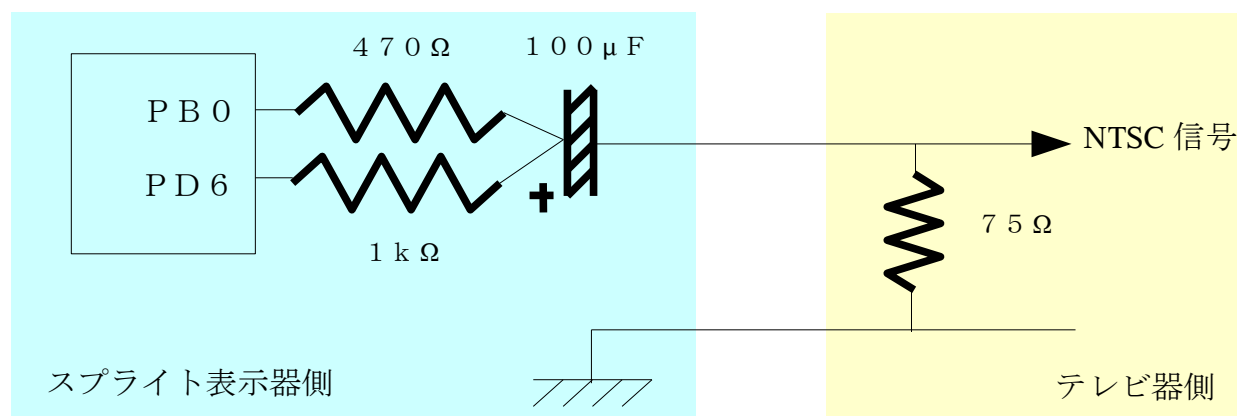


- ・ UART制御信号入力 : RXD端子 (PD0)
- ・ 背景用垂直SYNC : PD5端子
- ・ 背景用水平SYNC : PD4端子
- ・ UART背景制御信号 : TXD端子 (PD1)
- ・ 映像信号 : PB0端子
- ・ 合成SYNC : PD6端子

(注1) ポートBのPB0以外の端子は、常時ノイズが出力されているためNCとする。

(注2) RCA端子と抵抗の間にカップリングコンデンサを置く。

(3) 抵抗値について



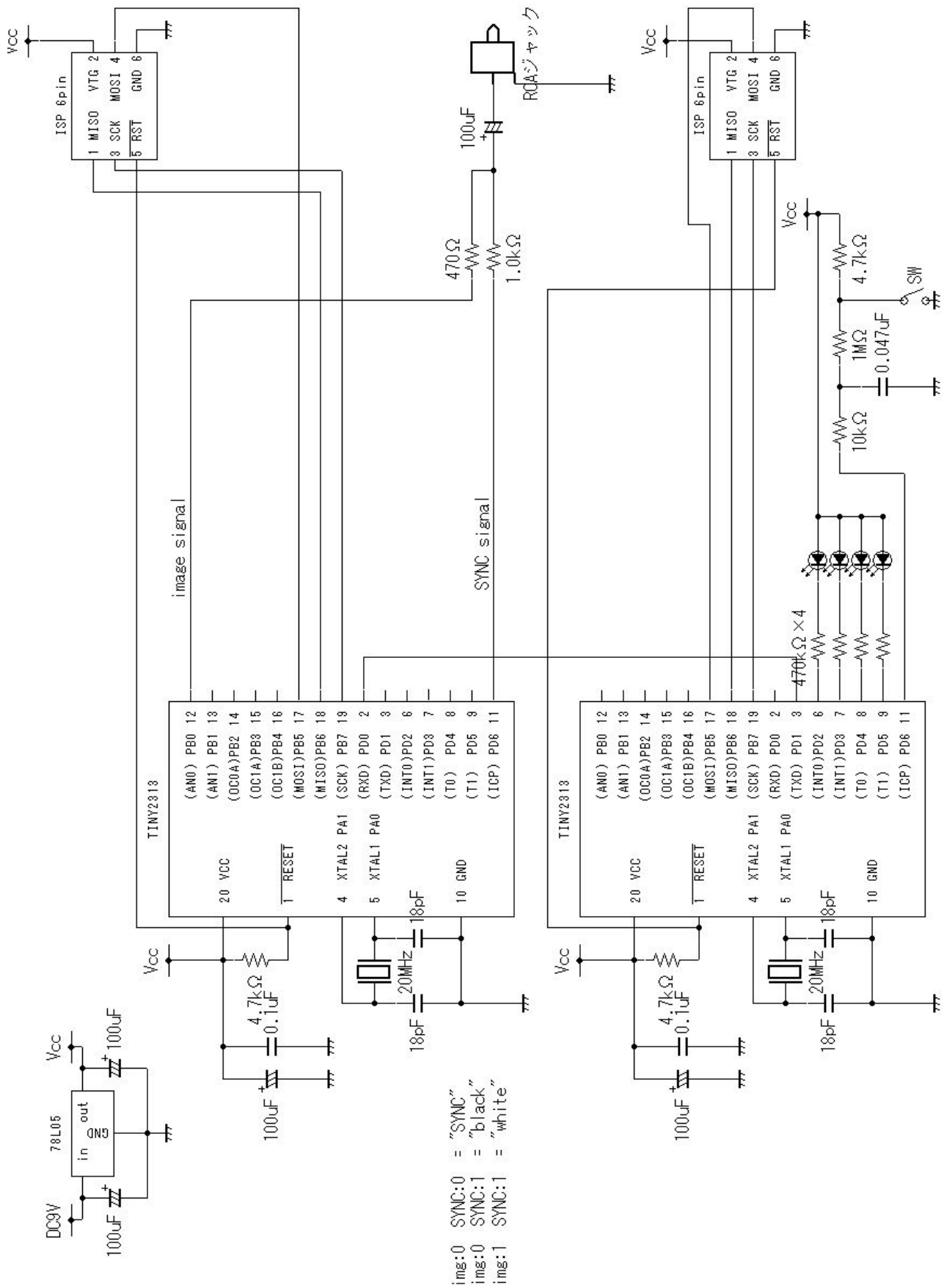
- SMPTE-170Mで規定された電圧（SYNC信号から白までが1Vppで、SYNC信号から黒までが0.3Vpp）に合わせるために、5Vの電圧を抵抗で分圧して実現する。その際の抵抗は上図のとおり470Ω、1kΩ、75Ωの3つを利用する。出力電圧は以下のとおり。（合成抵抗値と電圧の計算式は割愛する）

	SYNC	灰色	黒	白
PB0出力（映像）	0	1	0	1
PD6出力（SYNC）	0	0	1	1
出力電圧	0V	0.646V	0.3V	0.949V

(4) それ以外の外部回路

- 動作クロック：20MHz。セラミック発振子、クリスタル発振子、クリスタルオシレータのいずれかを使用。その際、フューズビットでCLKDIV8を指定してはならない。（クロックプリスケールレジスタが1/1倍でなければならない）
- UART接続端子は5Vpp（通常のTTLレベルの電圧）とする。（RS232Cレベルの電圧ではない）
- 背景表示器と組み合わせて使用する場合は、PB0（映像出力）の端子を背景表示器の映像出力とOR回路で合成してから470Ωの抵抗に接続する。（詳しくは背景表示器のマニュアルを参照）

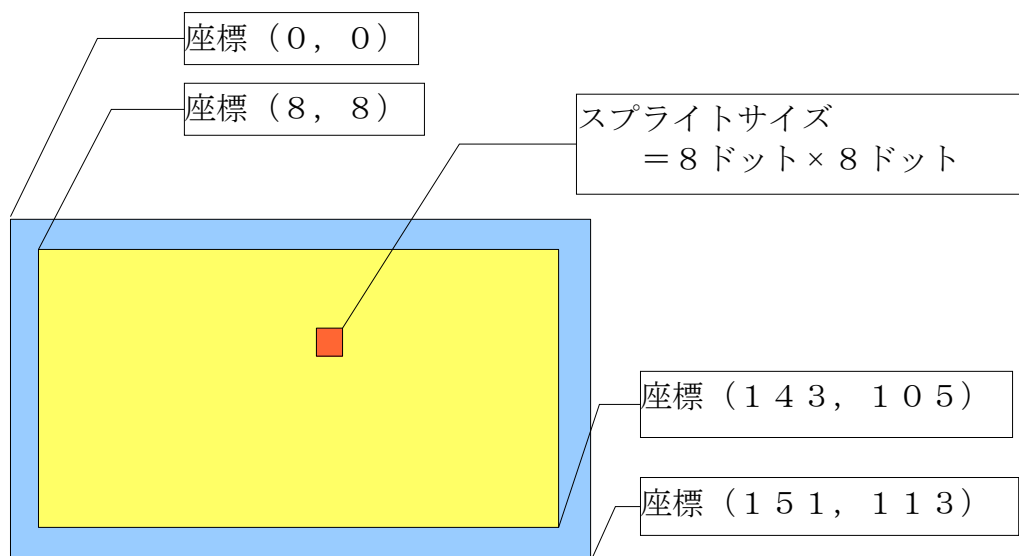
(5) サンプル回路図



3. 仕様

(1) 表示可能領域

- ・ スプライトが配置可能な座標は、左上 (0, 0) ~ 右下 (151, 113) の範囲。(下図水色の範囲)
- ・ ただし、このうち上下左右の端各8ドット分を除いた (8, 8) ~ (143, 105) の範囲だけが表示される。(下図黄色の範囲)
- ・ 白黒表示のみ対応。



- ・ スプライト1個のサイズは8ドット×8ドット。以下にそのサンプルとしてキャプチャー画面の部分拡大図を示す。(この例は、インベーダー風のキャラと、パックマンのモンスター風のキャラ)



- ・ 100種のビットマップをプログラムROM上に格納。(ROM中に存在するので、実行中に動的に書き換えることはできない)

(2) 表示可能スプライト数

- ・ 画面上に同時表示が可能なスプライト数は9個。(スプライト番号0~8)
- ・ 横1ライン上に9個同時表示が可能。
- ・ スプライトの表示/非表示を切り替える機能は実装していない(内部では9個全部のスプライトについて表示処理を常時行っている)。特定のスプライトを非表示にするためには、座標をユーザープログラム側で(0, 0)等の表示枠外に設

定する必要がある（表示枠外＝上図の水色部分）。

- その際、X軸やY軸を-1に設定してはならない。-1は16進表記で0xFFだが、この0xFFを受信するとデータシーケンスクリア（詳細は4章（1）を参照のこと）とみなされるので、座標の設定が意図どおりに反映されない。プログラムロジック中で座標の値が負値をとる可能性がある場合は、座標の値を0に補正してからスプライトの座標に指定するなどの工夫が必要。
- 当スプライト表示プログラムの初期化処理にて各スプライトの座標を（0，0）に初期化しているが、電源投入時にノイズをUARTが拾ってしまうと内部レジスタにノイズが書き込まれる恐れがあるので、ユーザープログラム側では念のために各スプライトの座標を（0，0）に初期化するルーチンを組み込んでおくことを推奨。初期化については4章（1）にて詳説する。

（3）UART接続仕様

- 115.2kbps、データ長8ビット、ノンパリ、ストップビット1。
- 連続してデータを受信することが可能。（送信側CPUで1バイト毎にウェイトを入れる必要なし）
- UARTの初期化処理が完了してデータ受信が可能になるまでに29.5μ秒を要する。（フューズビットで指定するスタートアップタイムは含まない。）

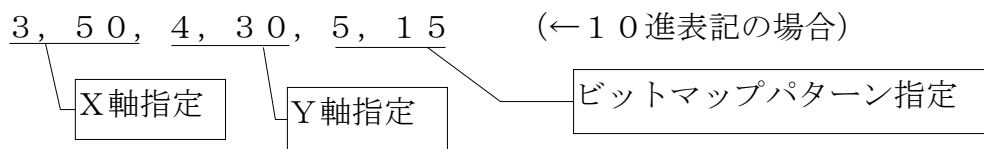
（4）内部レジスタとその制御

- 当システム内には、0番～27番の全28個の内部レジスタが存在する。（OPNチップのPSG制御につかうレジスタのようなイメージ）
- 内部レジスタ番号とデータの計2バイトのデータをCPU側から順に送ることで、各レジスタにデータが書き込まれる。（例：レジスタ1番に15を書き込む場合は、「1」と「15」の2バイトを順に書き込む）
- 0番～26番はスプライト制御用に使用。27番は特殊レジスタで、背景制御専用を使用する。（背景制御については（5）を参照）
- 各レジスタは以下のように定義されている。

内部レジスタ番号	意味
0番	スプライト0のX軸
1番	スプライト0のY軸
2番	スプライト0のビットマップパターン
3番	スプライト1のX軸
4番	スプライト1のY軸
5番	スプライト1のビットマップパターン
6番	スプライト2のX軸
:	:
:	:
25番	スプライト9のY軸

内部レジスタ番号	意味
26番	スプライト9のビットマップパターン
27番	特殊レジスタ (背景制御用)

- 例えば、スプライト1のX軸に50、Y軸に30、ビットマップパターンに15を設定する場合、CPU側から以下の6バイトを順に送信すればよい。



(5) 背景表示器の制御信号について

- 内部レジスタ27番は、背景制御用に使用する。
- 内部レジスタ27に書き込んだデータは、そのままTXD端子に出力される。
(特殊レジスタを介して背景表示器に制御信号をスルーさせることにより、CPU側のUART端子は1個しか使用せずに、スプライト表示器と背景表示器の2つのICを制御可能としている)
- 背景表示器とやり取りする実際のデータフォーマットについては、背景表示器のマニュアルを参照。

4. 使用方法

(1) 初期化

- ・電源ON時もしくはリセット時に、内部レジスタ0～26番を自動的に0クリアしている。よって通常はユーザープログラム側で初期化を行う必要は無い。
- ・ただし、電源ONやリセットの際に生じるノイズがUARTに乗ってレジスタが誤更新されたり、データシーケンス(※)が狂ったりする恐れがある場合等は、ユーザープログラム側にてデータシーケンスのクリアと全内部レジスタのクリアを行っておく必要がある。

(※データシーケンス：CPUからスプライト表示器へは「内部レジスタ番号」+「データ内容」の計2バイトを順に送信するが、この順序を便宜上データシーケンスと呼ぶ。もし電源オン時やリセット時にUARTにノイズが乗る等してそのノイズが正常データとみなされると、データシーケンスがずれたり内部レジスタを誤更新することになる。AT-TINY2313のフューズビットの設定や、電源回路・リセット回路の設計にも配慮のこと。)
- ・データシーケンスが狂っている恐れがある場合、16進で0xFF(10進で255)を1バイト送信することでデータシーケンスがクリアされて、1バイト目の受信待ちとなる。(内部レジスタ番号の指定を待つ状態)
- ・データシーケンスのクリア、及び各レジスタの0クリアを行うことで完全な初期化を行うことができる。具体的には、一旦0xFFを1バイト送信し、内部レジスタ0～26にすべて0を埋め込めばよい。初期化時のノイズによる誤動作が心配な場合、この手順にて初期化を行うことで回避ができる。
- ・内部レジスタ27番(背景表示器用の特殊レジスタ)に関しては初期化非対象(立ち上げ時は背景表示器側で独自に初期化処理を行っているが、スプライト表示器から背景表示器への初期化データは特に送信されない)。もし立ち上げ時に背景表示器の初期化が必要となる場合は、別途ユーザープログラム上から背景初期化用の制御データを送信する必要がある。
- ・スプライト表示器がUARTの初期化を終えるまでの必要時間は、シミュレータによると29.5μ秒(フューズビットで指定するスタートアップタイムは含まない)。よってスタートアップタイム+29.5μ秒以内にUARTの受信データが届くと正常に受信することが出来ないため、起動直後は送信側CPUからデータを送信する前に適宜タイミング待ちを行うこと。

(なお参考までに、背景表示器(Ver2)は同様に30.2μ秒なので、当表示器を経由して背景表示器の初期化データを送る場合にはスタートアップタイム+30.2μ秒以上のタイミング待ちを行う必要がある。詳細は背景表示器のマニュアルを参照。)

(2) 座標指定及びビットマップパターン選択

- ・初期化直後は、全9個のスプライトすべてが座標=(0, 0)、ビットマップパターン=0に設定されており、画面全体が黒となるビデオ信号が出力された状態になっている。(垂直/水平同期信号だけの映像信号)
- ・特定のスプライトを表示させるには、該当スプライトのX座標、Y座標、ビットマップパターン番号を該当の内部レジスタに書き込めばよい。以下に、WINA

VR用のサンプルを示す。(スプライト3番を(10, 20)の位置に移動し、ビットマップパターンを15にする場合の例)

```
void uart_putc(char c) {
    loop_until_bit_is_set(UCSR0A, UDRE0); //UDRE ビットが1になるまで待つ
    UDR0 = c;                               //データをuartに出力する
}

void set_sprite(unsigned char n, unsigned char x, unsigned char y, unsigned
char p) {
    uart_putc(n*3);           // n 番目の X座標
    uart_putc(x);            // x を指定
    uart_putc(n*3+1);        // n 番目の Y座標
    uart_putc(y);            // y を指定
    uart_putc(n*3+2);        // n 番目のパターン番号
    uart_putc(p);            // p を指定
}

int main(void)
{
    unsigned char n=3;
    unsigned char x=10;
    unsigned char y=20;
    unsigned char p=15;
    set_sprite(n, x, y, p);
    for(;;);
}
```

(3) 背景制御

- 特殊レジスタ (=内部レジスタ27番) にデータを書き込むことで、1バイトずつ背景表示器にデータを渡すことができる。
- 以下にWINAVR用のサンプルを示す。(背景座標(5, 4)に背景ビットマップパターン=2を書き込む場合の例…データの意味については背景表示器のマニュアルに詳説することとし、ここでは割愛する)

```
void uart_putc(char c) {
    loop_until_bit_is_set(UCSR0A, UDRE0); //UDRE ビットが1になるまで待つ
    UDR0 = c;                               //データをuartに出力する
}

void background(int x, int y, int p_no)
{
    uart_putc(27);           //特殊レジスタ (27番=UARTバイパス用) に
    uart_putc(y*16 + x);    //X座標、Y座標の合成を出力する
    uart_putc(27);           //特殊レジスタ (27番=UARTバイパス用) に
```

```
    uart_putc(p_no);    //背景番号を出力する
}

int main(void)
{
    unsigned char x=5
    unsigned char y=4
    unsigned char p=2
    background(x, y, p);
    for(;;);
}
```

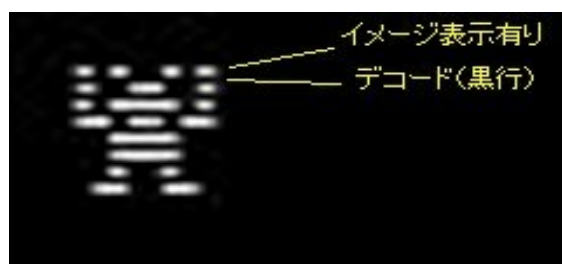
(4) その他

- プログラムROM（正確にはhexファイル）には、デフォルトで6章（2）のようなビットマップパターンが収録されている。このビットマップを各ユーザーの用途に合わせて自由にカスタマイズできるよう、hexファイルに対してビットマップを上書きするツールを別途用意してある。このツールの使用法は6章（1）を参照のこと。

5. 注意点など

(1) 表示方式・信号について (NTSCとの差異など)

- ・当表示器にて出力するビデオ信号は、NTSC信号 (SMPTE 170M) に類似する信号となっているが、以下の点で異なるので注意を要する。(ビデオ表示機器、録画機器によっては表示が正常に行われなかったり、全く表示されなかったり、場合によっては機器にダメージを与える恐れがあるかもしれないことを注意)
- ・座標データとビットマップデータから画面表示のイメージに変換するためのデコード処理には現実問題かなりの時間を要する。加えて、ATTINY2313はSRAMが128バイトと小さいため、1画面全体のビットイメージをSRAMに展開することはできない。現実解として、1ライン分だけのビットイメージをSRAM上に生成(デコード処理)し、そのイメージを表示する、ということをライン数分繰り返すことによって画面全体の表示を行っている。
この1行分のイメージを作成する処理時間を確保するために、走査線を2本1組にして1ラインはデコード(この間は黒行の表示)、1ラインはイメージ表示という具合に処理を行っている。この結果、1つのドットは画面上では横長の白い棒となって表示されることとなる。以下に実際のキャプチャー画面を示す。
(なお黒行についても他の行と同じように水平同期信号は出力されている。)



(2) インターレースについて

- ・SMPTE 170Mでは、1フィールドは262.5本の走査線、2フィールドで1フレームのインターレース表示を行うと定義されているが、当表示器においては1フィールドを262本の走査線で構成するため、インターレース表示が行われず、一般に市販されているテレビ受像機やビデオ機器等では問題なく表示可能であることをおおよそ確認しているが、機器によっては表示などに支障が起こる可能性がある。

(3) カラー信号について

- ・当表示器ではカラーバースト信号等のカラー画像生成の信号が含まれておらず、表示は常にモノクロのみとなる。

(4) 信号のタイミングについて

- ・一般的にSMPTE 170Mに近いタイミングで信号を生成しているが、回路に組み込む発振器の精度や、ソフトウェアで擬似的に信号を生成している都合、厳

密には一致していないという点と、処理の都合で部分的に信号のタイミングがずれている部分が存在する。およそ以下の点で信号のタイミングにズレがあることを認識。

<信号のズレその1：走査線1本の長さについて>

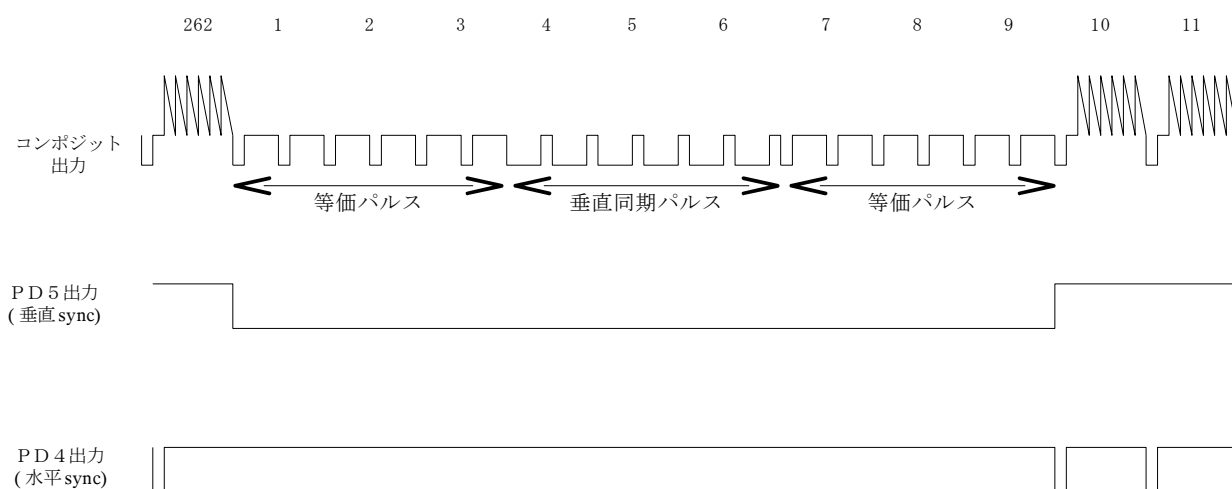
走査線1本あたり63.40 μ 秒で表示しており、厳密なSMPTE170Mとは若干のズレがある。

<信号のズレその2：走査線間のズレについて>

走査線と走査線の一部ラインカウンタ更新等の追加処理を行っているところなどがあり、他の走査線より1~2クロック分ほど水平同期期間が長いところが生じる。一般的なテレビ表示器では、水平同期期間が延びた場合その走査線の1本~数本下側において表示位置のズレが生じることが多い（このズレは数本内で収束される）。これを加味して、このような追加処理の下側には表示物が存在しない（黒行）ように設計してあり、基本的には表示内容に悪影響が及ばないように配慮している。ただし、画像表示器によっては表示がズレる可能性もあると考えられるので留意。

(5) シンクロ信号について

- ここで触れるシンクロ信号とは、PD6から出力されるコンポジットビデオ信号のシンクロ信号のことではなく、背景表示器に動作タイミングを伝えるための同期信号（PD5、PD4端子に出力する同期信号）についてである。
- 同期信号分離ICの代表LM1881を用いた場合に得られる同期信号は複合同期信号と垂直同期信号の2つであるが、当表示器が出力する同期信号は垂直同期信号と水平同期信号であることに留意。
- また、垂直/水平の各同期信号は以下のような波形であり、通常同期信号と比較して特に垂直同期信号は大きく異なることに留意。以下に、コンポジット映像出力とPD5端子（垂直同期）、PD4端子（水平同期）の出力内容を図示する。



(補足) 垂直同期信号は1H~9Hの間ずっとシンクロ信号が出力されている。またその間は水平同期信号は出力されない。また実際の処理としては、10H~37H、及び234H~262Hの間は水平同期信号だけの出力であり、映像信号は存在しないが、1H~9Hの垂直同期信号と区別しやすいように便宜上10H、

1 1 H、2 6 2 Hに映像信号を描き込んである。

- 背景表示器側ではこのPD 5 と PD 4 からの同期信号を外部割込みにて受け取り、以下の処理を行う。

① PD 5 が立ち下がったらラインカウンタを0クリアする

② PD 4 が立ち上がったからラインカウンタを1アップしてから、ラインカウンタの値に相当する行のイメージを出力する

(6) 出力電圧について

- 出力されるコンポジット信号は、SMPTE 1 7 0 Mに相当するよう1 V p pのうち同期信号が0. 3 V p p、映像信号分が0. 7 V p pとなっている。ただし、既成品の抵抗による電圧の分圧なので、厳密な電圧値とは若干誤差があることに留意。

6. 付録

(1) スプライトビットマップ書き換えツールについて

- HEXファイル上の0x2DB番地からの800バイトがビットマップデータとなっている。(スプライト1個分のデータが8バイト×100個分)
- データ構造は、ビットイメージの最上ライン8ドットが1バイト目。以下1ライン毎に1バイトで8ライン分の計8バイトがスプライト1個を構成する。また、ビットイメージの一番左のドットがLSB、右のドットがMSBのように並んでいる。
- このHEXファイル上のビットマップデータを直接書き換えることで、スプライトのビットマップをユーザーが独自に定義できる。(書き換えの際には、チェックサムも更新する必要有り)
- この書き換え処理と書き換え用データを含めたツールを、sprite_changer02.txtに収録している。このツールは、フリーウェアとして公開されている99basic上で動くようになっているが、基本的な命令だけを使っているため、変更無しもしくは若干の変更で大抵のBASICインタプリタ/コンパイラで動作可能。
- sprite_changer02.txtプログラム中のDATA内容を適当に書き換えて実行することで、ビットマップデータを置き換えたHEXファイルが生成される。このHEXファイルを元のHEXファイルの代わりに使用することで、ユーザー独自のビットマップを組み込んだスプライト表示が可能となる。
- なお、このツールにおいて更新前ファイル、更新後ファイルはデフォルトでそれぞれntsc_sprite02_orgin.hex、ntsc_sprite02_update.hexとなっている。必要に応じて適宜修正されたい。
- ntsc_sprite02_tmp.hexは一時ファイルのため、作業後は削除可能。
- 99basic等のBASICインタプリタ/コンパイラの入手にあたっては、vectorなどのソフトウェアダウンロードサービスを利用されたい。もしくは、適宜使い慣れた言語にリライトのうえ使用願いたい。

(2) デフォルトのビットマップについて

- sprite_changer02.txtのDATA中に、デフォルトで格納されているビットマップに相当するものが格納されている。(0が黒、1が白)
- DATA文に併せて、キャラクターの名称をコメントで記入してあるので、ビットマップ番号とビットマップイメージの紐付けを調べる際や、ビットマップをオリジナルのものに置き換える際に適宜参考にされたい。