ＡＶＲ２３１３－スプライト表示プログラムＶ４　マニュアル

－目次－

１．機能概要

・ＵＡＲＴ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、ＮＴＳＣ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）

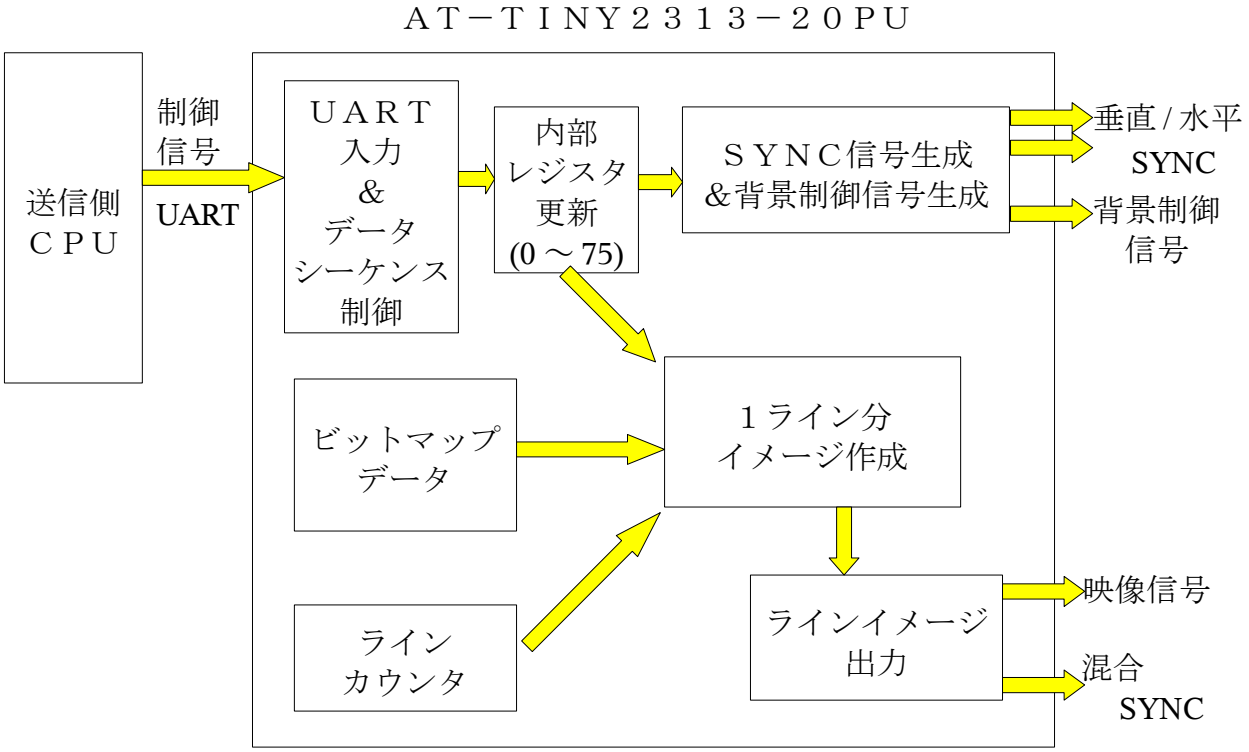
・スクリーン座標系は、左上（０，０）～右下（１５１，１１３）の範囲。このうち外側の８ドット分を除いた（８，８）～（１４３，１０５）の範囲が表示される。

・同時表示可能なスプライト数は２５個。ただし、横１ライン上に同時表示できる限度は４個まで。

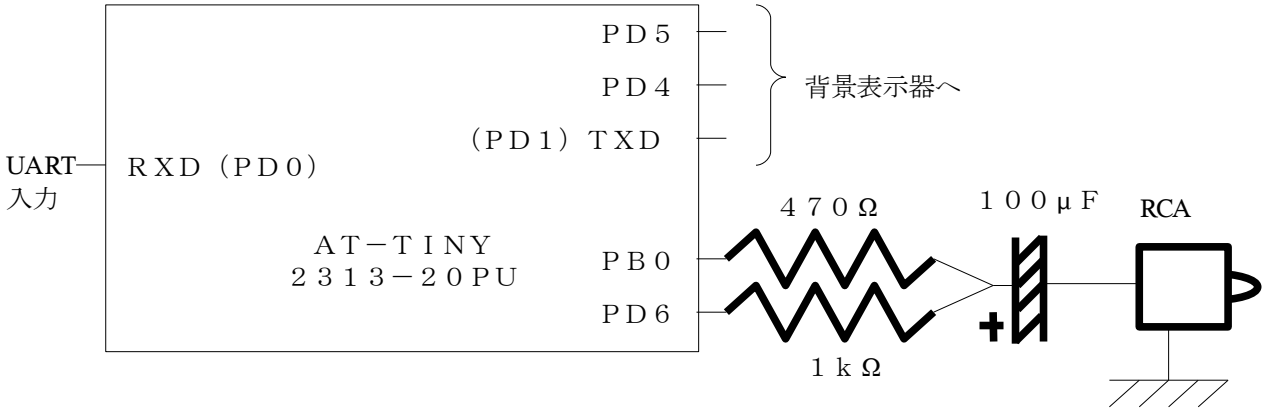
・背景表示器との接続用として、垂直同期信号、水平同期信号、ＵＡＲＴによる背景制御信号の３つの信号を出力する。

２．動作環境

（１）ブロック図



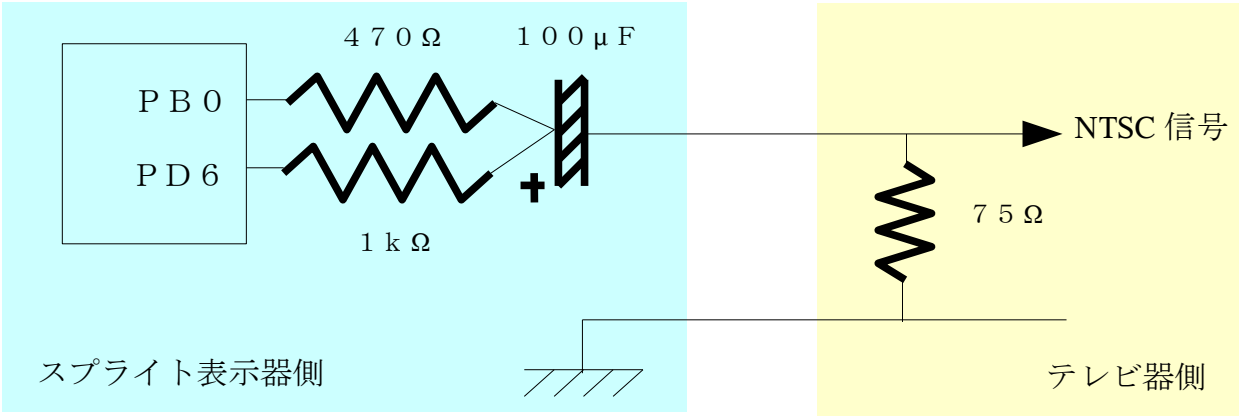
（２）使用回路図



- ・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



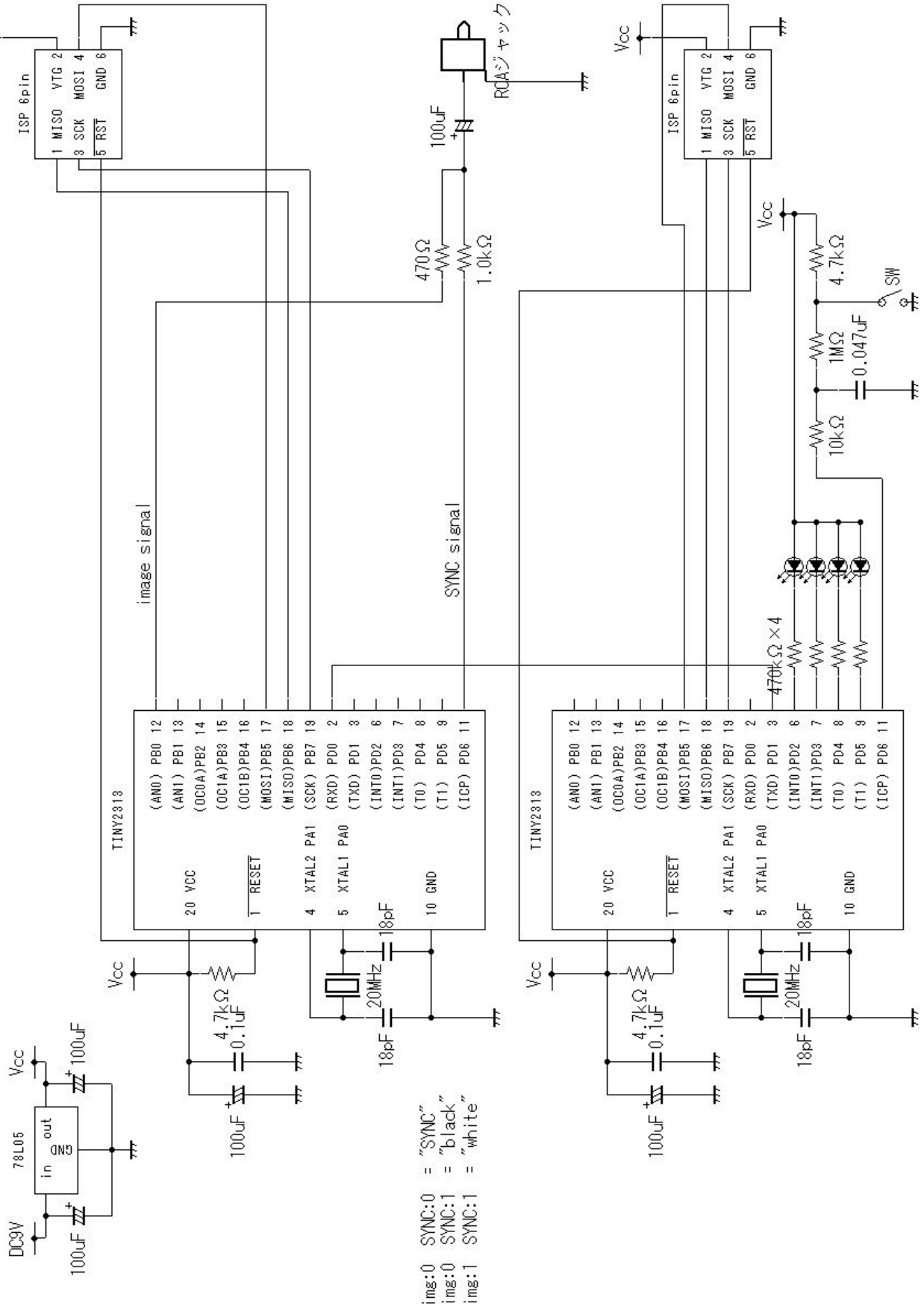
・ＳＭＰＴＥ－１７０Ｍで規定された電圧（ＳＹＮＣ信号から白までが１Ｖｐｐで、ＳＹＮＣ信号から黒までが０．３Ｖｐｐ）に合わせるために、５Ｖの電圧を抵抗で分圧して実現する。その際の抵抗は上図のとおり４７０Ω、１ｋΩ、７５Ωの３つを利用する。出力電圧は以下のとおり。（合成抵抗値と電圧の計算式は割愛する）

	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。セラロック発信子、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）

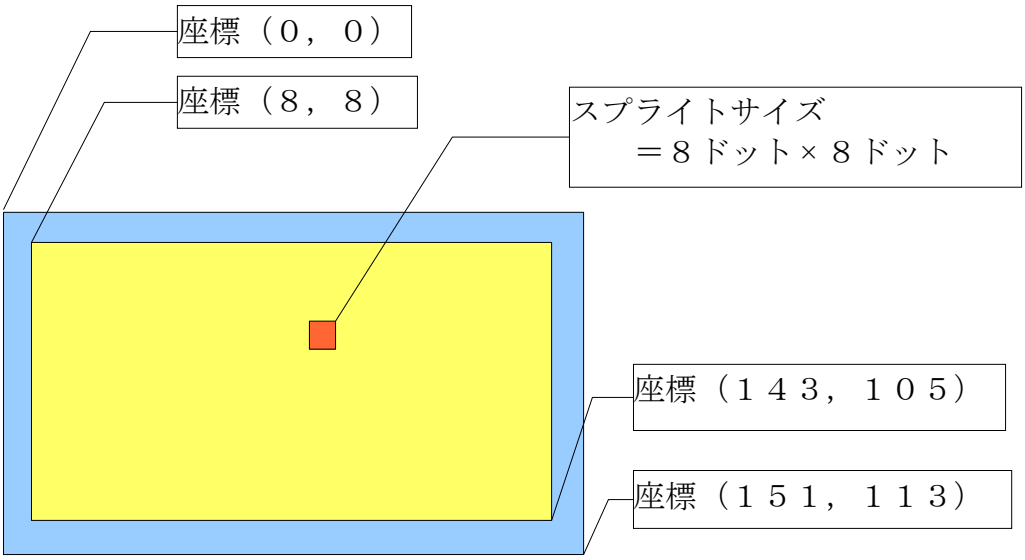
（5）サンプル回路図


Vcc
ISP 6pin
1 MISO
VTG 2
3 SCK
MOSI 4
5 RST
GND 6
RCAジャック
100uF
470Ω
1.0kΩ
4.7kΩ
SW
1MΩ
0.047uF
10kΩ
470kΩ×4
image signal
SYNC signal
TINY2313
20 VCC
1 RESET
4 XTAL2
5 XTAL1
10 GND
(AN0) PB0 12
(AN1) PB1 13
(OC0A)PB2 14
(OC1A)PB3 15
(OC1B)PB4 16
(MOSI)PB5 17
(MISO)PB6 18
(SCK) PB7 19
(RXD) PD0 2
(TXD) PD1 3
(INT0)PD2 6
(INT1)PD3 7
(T0) PD4 8
(T1) PD5 9
(ICP) PD6 11
PA1
PA0
18pF
20MHz
4.7kΩ
0.1uF
78L05
in
out
GND
DC9V
img:0 SYNC:0 = "SYNC"
img:0 SYNC:1 = "black"
img:1 SYNC:1 = "white"

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２５個。（スプライト番号０～２４）
・ただし、横１ライン上に同時表示できる限度は４個まで。（横１ライン上にスプライトが５つ以上並んでいる場合、スプライト番号が若い４つが優先して表示される）
・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２５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。）

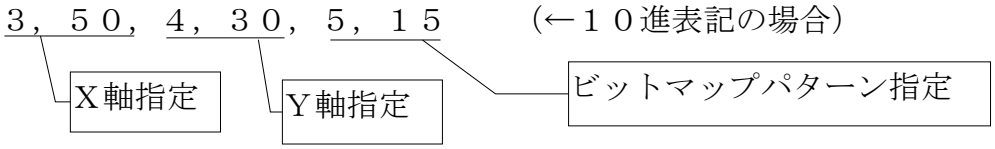
（４）内部レジスタとその制御

・当システム内には、０番～７５番の全７６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～７４番はスプライト制御用に使用。７５番は特殊レジスタで、背景制御専用に使用する。（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
：	：
：	：

内部レジスタ番号	*意味*
７３番	スプライト２４のＹ軸
７４番	スプライト２４のビットマップパターン
７５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７５番は、背景制御用に使用する。
・内部レジスタ７５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０～７４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０～７４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７５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２５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７５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(75);        //特殊レジスタ（75 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(75);        //特殊レジスタ（75 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(;;);
}
```

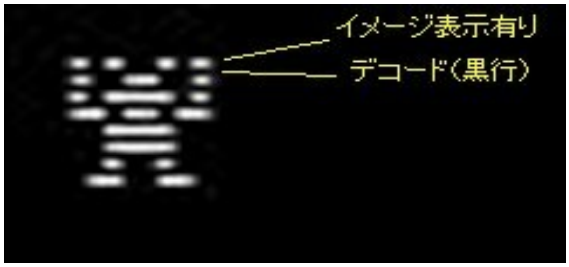
（４）その他

・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（ＳＭＰＴＥ１７０Ｍ）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、ＡＴ－ＴＩＮＹ２３１３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

・ＳＭＰＴＥ１７０Ｍ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ない。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

・全般的にＳＭＰＴＥ１７０Ｍ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＞

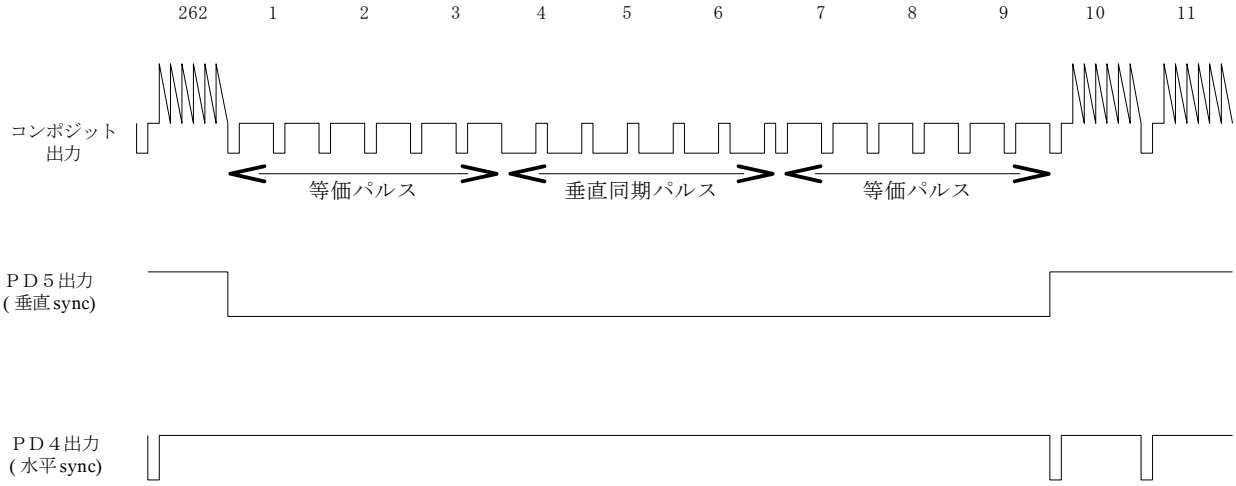
走査線１本あたり６３．４０μ秒で表示しており、厳密なＳＭＰＴＥ１７０Ｍとは若干のズレがある。

＜信号のズレその２：走査線間のズレについて＞

走査線と走査線の間に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、通常の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、

１１Ｈ、２６２Ｈに映像信号を描き込んである。

・背景表示器側ではこのＰＤ５とＰＤ４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。

①ＰＤ５が立ち下がったらラインカウンタを０クリアする
②ＰＤ４が立ち上がった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、ＳＭＰＴＥ１７０Ｍに相当するよう１Ｖｐｐのうち同期信号が０．３Ｖｐｐ、映像信号分が０．７Ｖｐｐ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０ｘ３２２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_changer04.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ので、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_changer04.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_sprite04_origin.hex、ntsc_sprite04_update.hex となっている。必要に応じて適宜修正されたい。
・ntsc_sprite04_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_changer04.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。