ＡＶＲ２３１３－背景表示プログラムＶ２　マニュアル

－目次－

１．機能概要

・ＵＡＲＴ信号から背景キャラクターの座標及びビットマップ番号と垂直／水平同期信号を入力し、同期信号に合わせて背景画像の信号を出力する。
（単体で同期信号を出力する機能は有しないため、スプライト表示器のような同期信号出力機能を持った機器と接続して、両者の映像信号同士をＯＲ回路で合成して用いる）

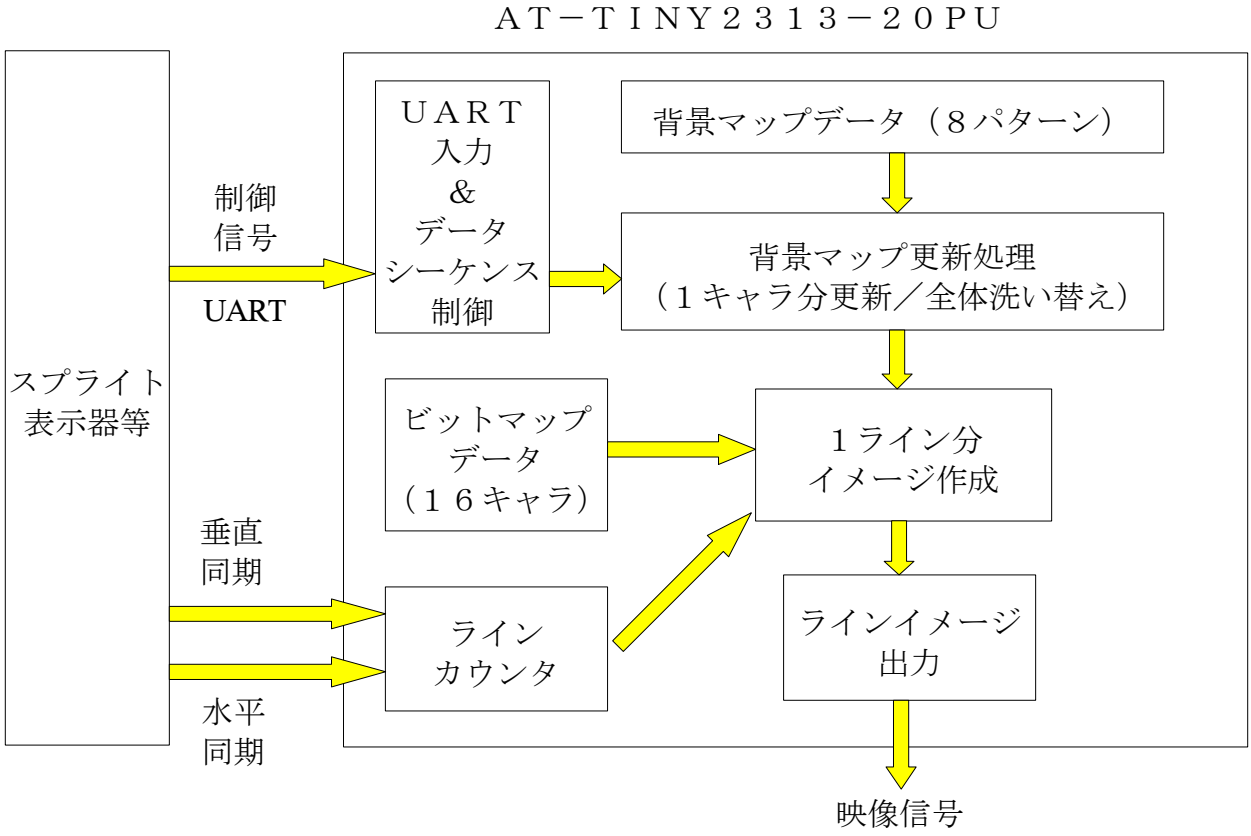
・背景マップの座標系は、左上（０，０）～右下（１５，１１）の範囲。１６×１２キャラクタで構成される。

・背景画像１キャラクター分のサイズはスプライトサイズに合わせて８ドット×８ドット。白黒のみの表示。１６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）

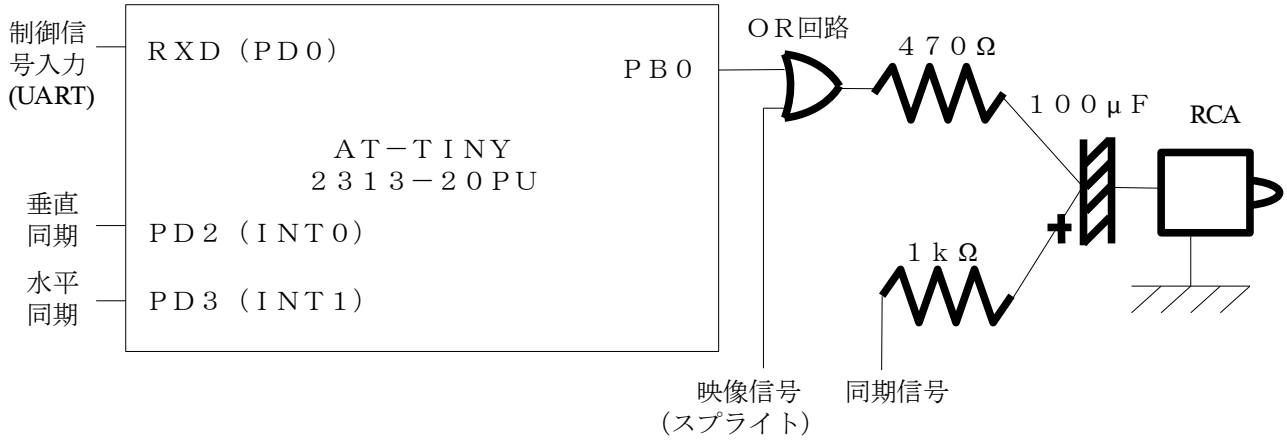
・マップ全体の洗い替え機能有り。洗い替えるマップ８パターンをＲＯＭ上に格納。（実行中に動的に書き換えることは出来ない）

２．動作環境

（１）ブロック図



（２）使用回路図



・制御信号入力　　：　ＲＸＤ端子（ＰＤ０）
・垂直同期入力　　：　ＰＤ２端子（ＩＮＴ０）
・水平同期入力　　：　ＰＤ３端子（ＩＮＴ１）
・映像信号　　：　ＰＢ０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）ＰＢ０からの映像出力は、スプライトの映像信号とＯＲ回路で合成した後に抵抗器を用いて同期信号と合成、コンポジット信号を生成する。
（注３）ＲＣＡ端子と抵抗の間にカップリングコンデンサを置く。

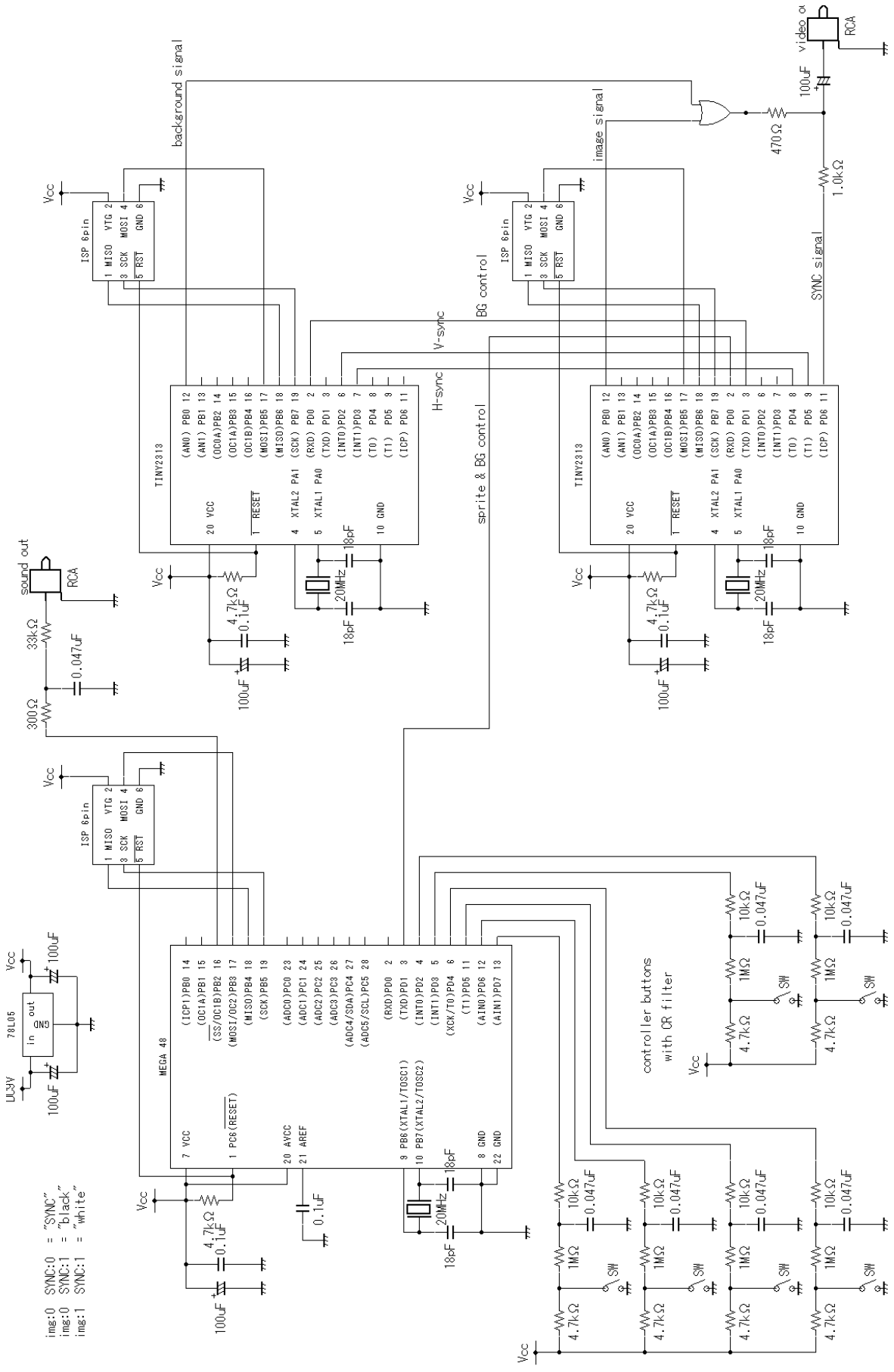
（３）抵抗値について

・スプライト表示器と同様に、ＳＭＰＴＥ－１７０Ｍで規定された電圧に合わせるために５Ｖを抵抗で分圧して実現する。詳細はスプライト表示器のマニュアルを参照。

（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）

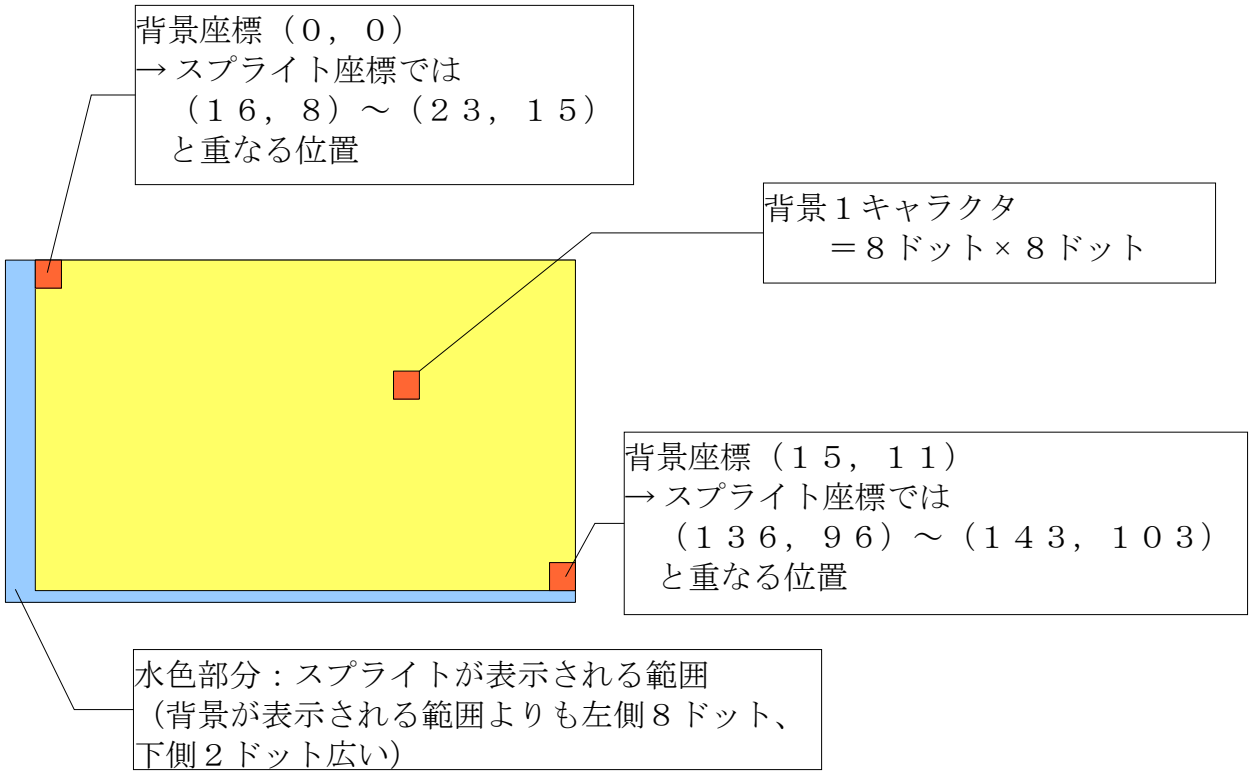
（5）サンプル回路図


video out
RCA
100uF
470Ω
1.0kΩ
background signal
image signal
SYNC signal
BG control
V-sync
H-sync
sprite & BG control
TINY2313
ISP 6pin
MEGA 48
sound out
33kΩ
0.047uF
300Ω
78L05
LIC3V
controller buttons
with CR filter
10kΩ
1MΩ
4.7kΩ
SW
4.7kΩ
0.1uF
20MHz
18pF
Vcc
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

・背景キャラクターが配置可能な座標は、左上（０，０）～右下（１５，１１）の１６×１２の範囲。（下図黄色の範囲）
・これをスプライトと重ね合わせると、このうち左端８ドット分と下端２ドット分はスプライトの表示領域の方が広いことに留意。（下図水色の範囲）
・白黒表示のみ対応。



・背景ビットマップ１個のサイズは８ドット×８ドット。
・１６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）
・Ｘ座標、Ｙ座標及び背景ビットマップ番号を指定することで、その座標に表示中の背景キャラクターを動的に更新することが可能。
・また、背景全面を一気に更新する「洗い替え機能」も搭載。洗い替えするための背景データとして、８画面分のデータをプログラムＲＯＭ上に格納。（８画面分のマップパターンデータはＲＯＭ中に存在するため、実行中に動的に書き換えることは出来ない。実行中は８つの画面のうちどれかを選択して切り替えることができる。）
・スクロール機能は搭載していない。
・背景マップに書かれたデータを読み出すことは出来ない。（書き込みだけ可）

（２）座標指定方法と洗い替え機能

・Ｘ座標、Ｙ座標、及び背景ビットマップ番号を指定することで、該当座標に表示

するキャラクターを切り替えることができる。
ＵＡＲＴから２バイトの連続したデータを入力してその２バイトからＸ軸、Ｙ軸、ビットマップ番号を取り出し、該当の座標にビットマップ番号に相当するデータを書き込む。これにより該当座標に任意のキャラクターを表示することを可能とする。

・Ｘ軸は０～１５の範囲。Ｙ軸は０～１１の範囲、ビットマップ番号は０～１６の範囲の値をとる。
・ＵＡＲＴから入力する２バイトのデータのうち、１バイト目でＸ軸とＹ軸の指定を行う。上位４ビットがＹ軸、下位４ビットがＸ軸をあらわす。２バイト目はビットマップ番号を表す。上位４ビットは常に０、下位４ビットがビットマップ番号をあらわす。
・洗い替え機能を用いて背景全体を一気に洗い替える場合は、Ｘ軸に０、Ｙ軸に１５、ビットマップ番号の代わりに背景マップ番号（０～８）を指定することで実施する。
・電源投入時やリセット時に、もしＵＡＲＴがノイズを正常データとして取り込んだ場合、データシーケンスが狂うことが想定される。また、Ｘ軸Ｙ軸が指し示す座標が表示可能領域外の場合にもデータシーケンスが強制的にクリアされ、ユーザーが想定しているデータシーケンスと狂う場合がある。このような場合に備えて、データシーケンスを強制的にクリアすることが可能となっている。具体的には、１バイトのデータ０ｘＦＦ（１６進表記）をＵＡＲＴから入力することで行われる。このコードを受け取った場合、データシーケンスは１バイト目のデータ待ち状態に設定される。
・スプライト表示器を経由して背景表示器にデータを渡す場合、スプライト表示器の特殊レジスタ（Ｖ４であれば７５番レジスタ、Ｖ２であれば２７番レジスタが該当）に書き込んだデータが背景表示器に送信される。
＜例＞スプライト表示器Ｖ４を経由して、背景Ｘ座標＝５、Ｙ座標＝２にビットマップ番号＝７を設定する場合（つまり１バイト目が０ｘ２５＝３７、２バイト目が０ｘ０７＝７のデータを送る場合）
→送信側ＣＰＵからスプライト表示器に対して「７５」「３７」「７５」「７」の４バイトを送信することにより、背景表示器には「３７」「７」の２バイトが送信される。

（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。
・スプライト表示器を経由した場合は、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）
＜補足＞これは、スプライト表示器に２バイト送信する毎に背景表示器に１バイトのデータが送られることにより、実質の転送レートが５７．６ｋｂｐｓ相当となり、内部処理が間に合う計算になっているが、１１５．２ｋｂｐｓで直接連続したデータを送信し続けると内部処理が追いつかなくなってデータの読み飛ばしが生じる恐れがある。そもそもスプライト表示器のような垂直／水平同期信号を生成する機器経由を前提にしているのでこのような仕様となっているため、それ以外の方法で接続する場合には注意が必要。
・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３０．２μ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）同期信号と処理タイミングについて

・当表示器は、独自に同期信号（垂直／水平とも）を生成する機能を有してはおらず、接続する機器（スプライト表示器など）から垂直／水平同期のタイミングを受信することで同期タイミングを認識する仕組みとなっている。
・具体的には、ＰＤ２が立ち下がった時にラインカウンタを０とし、ＰＤ３が立ち上がった時はラインカウンタを１加算しつつそのラインカウンタ値に相当する映像信号１本分を出力するという仕組みになっている。
・よって、ＰＤ２に垂直同期信号を、ＰＤ３に水平同期信号を入力することで、接続機器との動作を同期させることが可能となる。
・スプライト表示器と接続する場合には、スプライト表示器のＰＤ５（垂直同期出力）をＰＤ２に、同ＰＤ４（水平同期出力）をＰＤ３に入力する。

４．使用方法

（１）初期化

・電源ＯＮ時もしくはリセット時に、内部のビデオＲＡＭを自動的に０クリアし、併せてデータシーケンスもクリアしている。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内部ビデオＲＡＭのクリアを行っておく必要がある。

（※データシーケンス：接続する機器から背景表示器へは「ＸＹ軸」＋「パターン番号」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ビデオＲＡＭ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。（ＸＹ座標の指定を待つ状態）
・データシーケンスのクリア、及び各座標に０を設定する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、Ｘ軸＝０～１５、Ｙ軸＝０～１１の全座標にパターン番号０を書き込む。
・もしくは、デフォルトのマップパターン０番にはＶＲＡＭ全体を０でクリアするデータが格納されているため、これを用いて全画面洗い替えを行っても良い。その場合は「０ｘＦＦ」「０ｘＦ０」「０」の順にデータを送ればよい。６章で説明するオリジナルマップデータ作成を行う場合は、このように全画面を０クリアするようなマップパターンを１つ用意しておくことを推奨。
・背景表示器が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３０．２μ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＋３０．２μ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。

（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、全画面の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信号が出力された状態になっている。

（３）背景制御

・送信用ＣＰＵから特殊レジスタ（例えば、スプライト表示器Ｖ４なら内部レジスタ７５番）にＸＹ座標とビットマップ番号を送信することで、背景の特定位置のビットマップを変更することができる。
・この送信データは２バイトで構成される。１バイト目は上位４ビットがＹ軸、下位４ビットがＸ軸。２バイト目は上位４ビットが常に０、下位４ビットがビット

マップ番号。詳しくは、３章（２）を参照。
・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）

```
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）マップ洗い替え

・Ｘ軸に０、Ｙ軸に１５を設定し、ビットマップ番号の代わりに背景マップ番号を指定することで、全画面洗い替えを行うことができる。詳しくは、３章（２）を参照。
・以下にＷＩＮＡＶＲ用のサンプルを示す。（背景マップ番号２で全画面洗い替えをする場合の例）
・なお、背景マップ番号２には、デフォルトでドットイートタイプのゲームの画面が収録されている。

```
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=0
    unsigned char y=15
    unsigned char p=2
    background(x,y,p);
    for(;;);
}
```

（５）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）に示すようなビットマップパターン１６件、及び背景マップパターン８件が収録されている。これらのデータを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スプライト表示器と同様、ＳＭＰＴＥ１７０Ｍの規格には厳密に準拠していないことに留意。（スプライト表示器のマニュアルを参照）
・１フィールドあたりの走査線数は２６２本を想定しているが、独自に走査線を生成しているわけではなく、接続する機器（主にスプライト表示器を想定）まかせとなっている。接続する機器側で２６２本以外の走査線数の同期信号を生成している場合、想定しない表示を行う恐れがあるため注意要。
（数本～１０本程度少ない走査線数であれば、大きな問題は生じないロジックとなっている。逆に多い場合は、下記（５）の補足のとおり想定外となるため厳禁。）

（２）インターレースについて

・スプライト表示器と同様に、１フィールドの走査線は２６２本を想定しているため、画像表示機器によっては表示などに支障が出る恐れがある。詳細はスプライト表示器のマニュアル参照。

（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

・ＳＭＰＴＥ１７０Ｍとの差異による注意点は、およそスプライト表示器の注意点と同様のためスプライト表示器のマニュアルを参照。
・当表示器とスプライト表示器を組み合わせて使用する場合、それぞれのＩＣに別々の発振器を接続して使うと、各発振器間で生じるズレ（製品誤差など）によって映像の同期が最大で１クロック分ズレる可能性がある。このため、スプライト画像に対して背景画像が１クロック分（＝０．０５μ秒…走査線１本の約０．０８％の時間に相当）の範囲で左右にブレることになる。このブレを押さえるためには、発振器に２０ＭＨｚの「オシレータ」を用いて２つのＩＣに同一のクロックを供給し、クロックを完全に同期させる必要がある。
・２章（５）のサンプル回路では２つのＩＣには別々のクロックが供給されており、ブレが生じる典型例である。
・このブレが生じるのは背景画像の範囲内だけに留まる。合成される側のスプライト信号やシンクロ信号には特に影響は及ばない。

（５）シンクロ信号について

・ここでいうシンクロ信号とは、コンポジット信号中のシンクロ信号ではなく、スプライト表示器から当表示器に送られる垂直／水平同期信号のことである。
・垂直同期期間中には、ＰＤ３に水平同期信号の立ち上がり信号入力が入力されて

はならない。
＜補足＞内部では、垂直同期信号の立ち下り（つまり１Ｈの一番最初の瞬間）からラインカウントを開始していて、かつ垂直同期期間９Ｈの間は同期信号が入力されないことが前提となっている。２６２本－９本＝２５３本となるので、内部のラインカウンタは１バイト（０～２５５）で足りることとなる。もし垂直同期期間に水平同期信号が入ると、１バイトで表現できる限界を超えてしまい、誤動作の原因となりうる。
・スプライト表示器のマニュアルで図示したようなシンクロ信号の入力を想定しているので、詳しくはスプライト表示器のマニュアルを参照のこと。

（６）出力電圧について

・スプライト表示器のマニュアルを参照。

（７）スクロールについて

・当表示器のバージョン２（Ｖ２）では、技術的容量的な問題などによりスクロール機能は実装していない。
・よってドット単位でのスクロールは当バージョンでは不可能だが、８ドット単位のスクロールであれば、以下の方法で一応実現可能。
（方法１）
表示中の背景マップと同じデータを別途送信側ＣＰＵでも保持しておき、適宜スクロール後のデータを作成→１画面分丸ごと送信する。
なお、その際に要する１回の転送時間は、約０．０３３３…秒（１１５．２ｋｂｐｓの速度で９６×２バイトの背景データをスプライト表示器経由で送信した場合）。この方法は転送側ＣＰＵに大きなメモリの確保が必要となり、また高速なスクロールの用途にも向かない。
（方法２）
あらかじめ、８枚の背景マップにパタパタ漫画式の背景を登録しておいて、必要に応じてこの８枚をパタパタと切り替える。この方法であれば、転送速度は気にとめるレベルでは無いと考えられる。一方、非常に単調な繰り返しの動きしか表現できない。

６．付録

（１）背景マップ／背景ビットマップ書き換えツールについて

・ＨＥＸファイル上の０ｘ２２４番地からの１２８バイトがビットマップデータ、その後ろ（０ｘ２Ａ４番地）から７６８バイトが背景マップデータとなっている。（ビットマップ１個分のデータが８バイト×１６個＝１２８バイトと、背景マップ１枚が９６バイト×８枚＝７６８バイト）
・ビットマップのデータ構造は、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バイトがビットマップ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。
・背景マップのデータ構造は、背景マップの最上ライン左上から右上に向けて２マスにつき１バイト、１６マスは計８バイトで１ラインを形成。以下、１ライン毎に８バイトで１２ライン分の計９６バイトが背景マップ１枚を構成する。なお２マス１バイトのうち上位４ビットが左側のマス、下位４ビットが右側のマスのキャラクターを示している。

・このＨＥＸファイル上の各データを直接書き換えることで、背景画像のビットマップや背景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、bg_changer.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。
・bg_changer.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、ビットマップや背景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・背景マップを組み込んだ表示が可能となる。
・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_bg2_origin.hex、ntsc_bg2_update.hex となっている。必要に応じて適宜修正されたい。
・ntsc_bg2_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

・ntsc_bg2_origin.hex の DATA 中に、デフォルトで格納されているビットマップ及び背景マップに相当するものが格納されている。
・DATA 中のビットマップの数値は０が黒、１が白を示す。背景マップ上の番号は０ｘ００～０ｘ０Ｆのビットマップ番号を示す。
・DATA 内容についてはデフォルトの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。