

—目次—

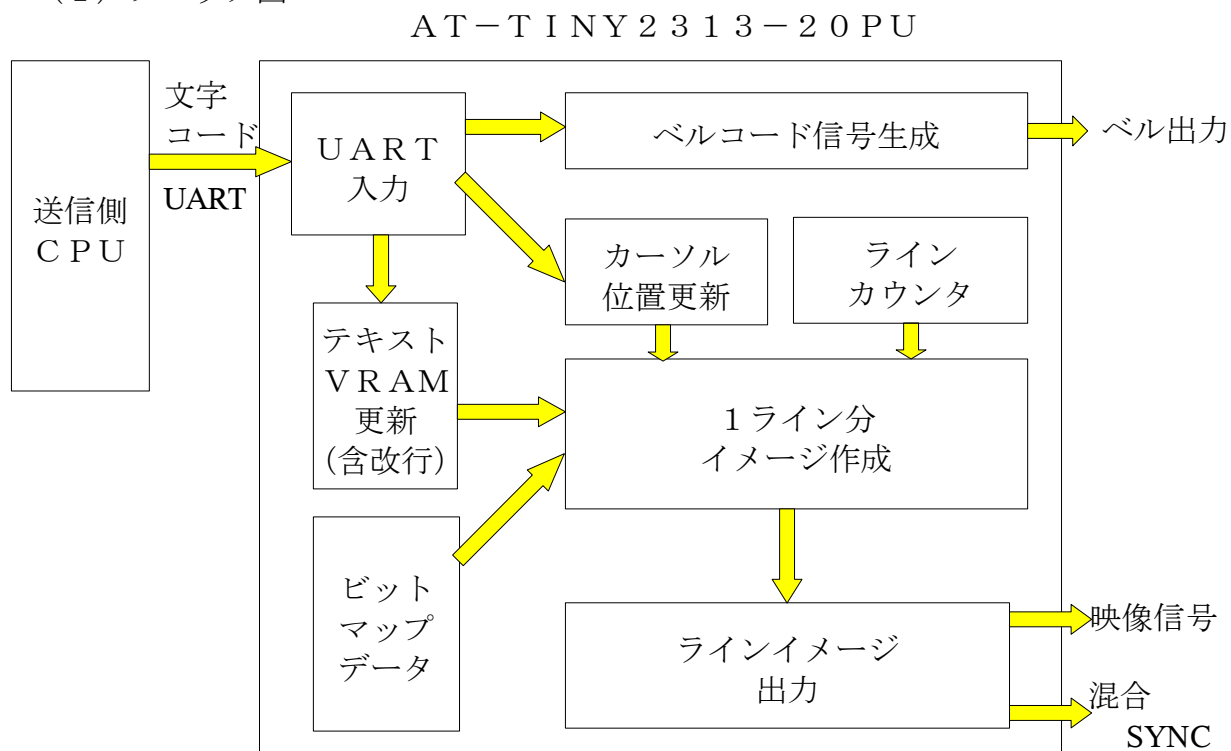
1. 機能概要
2. 動作環境
  - (1) ブロック図
  - (2) 使用回路図
  - (3) 抵抗値について
  - (4) それ以外の外部回路
  - (5) サンプル回路図
3. 仕様
  - (1) 表示領域
  - (2) 表示可能文字数
  - (3) 特殊文字コード
  - (4) U A R T接続仕様
4. 使用方法
  - (1) 初期化
  - (2) 文字表示、およびカーソル制御
5. 注意点など
  - (1) 表示方式・信号について (NTSCとの差異など)
  - (2) インターレースについて
  - (3) カラー信号について
  - (4) 信号のタイミングについて
  - (5) 出力電圧について
6. 付録
  - (1) キャラクターコード表

## 1. 機能概要

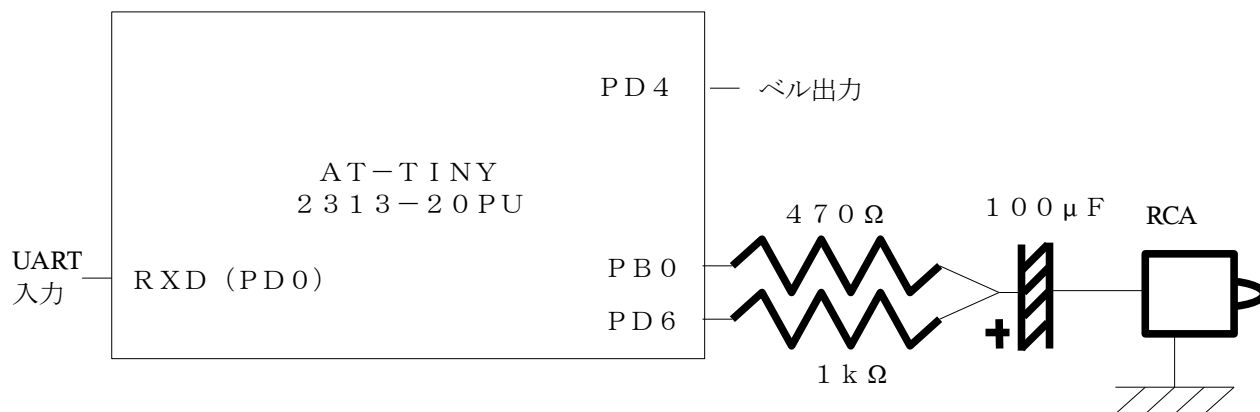
- U A R T信号から文字キャラクターの A S C I Iコードを入力し、N T S Cコンポジット信号（の類似信号）を生成、出力する。
- 文字1個のサイズは5ドット×6ドットのオリジナルフォント。白黒のみの表示。アルファベット、英数字、その他記号を合わせて70種のビットマップをプログラムROM上に格納。
- 横13文字×縦7行の合計91文字を表示。1文字表示するごとにカーソルが自動的に移動。適宜改行やスクロールが行われる。
- 座標系は、左上（0，0）～右下（12，6）の範囲。カーソル座標を直接指定してカーソル移動することが可能。
- 全画面クリア、ベルコード、改行コードの特殊コードを搭載。
- カーソル位置にはカーソルが点滅する。（非表示には出来ない）

## 2. 動作環境

### (1) ブロック図



### (2) 使用回路図

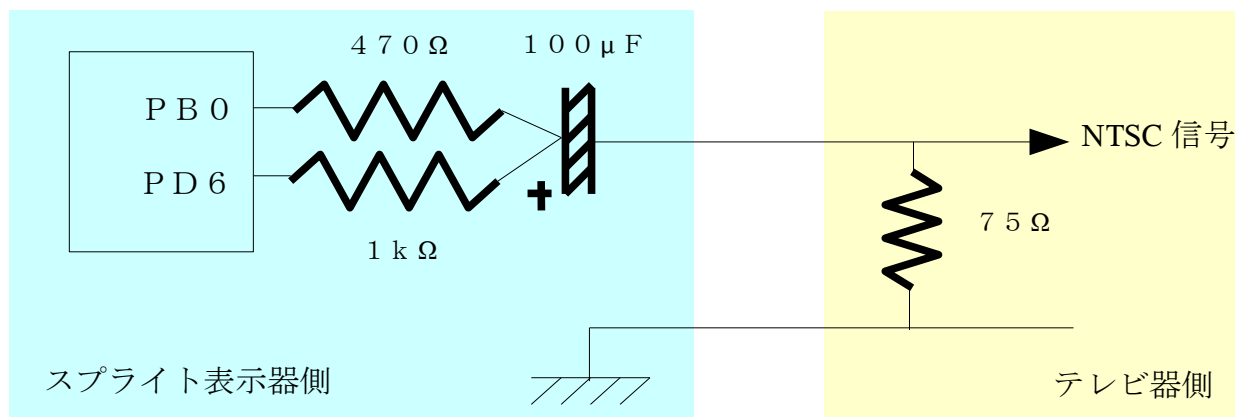


- ・ UART制御信号入力 : RXD端子 (PD0)
- ・ 映像信号 : PB0端子
- ・ 合成SYNC : PD6端子
- ・ ベル出力 : PD4端子

(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



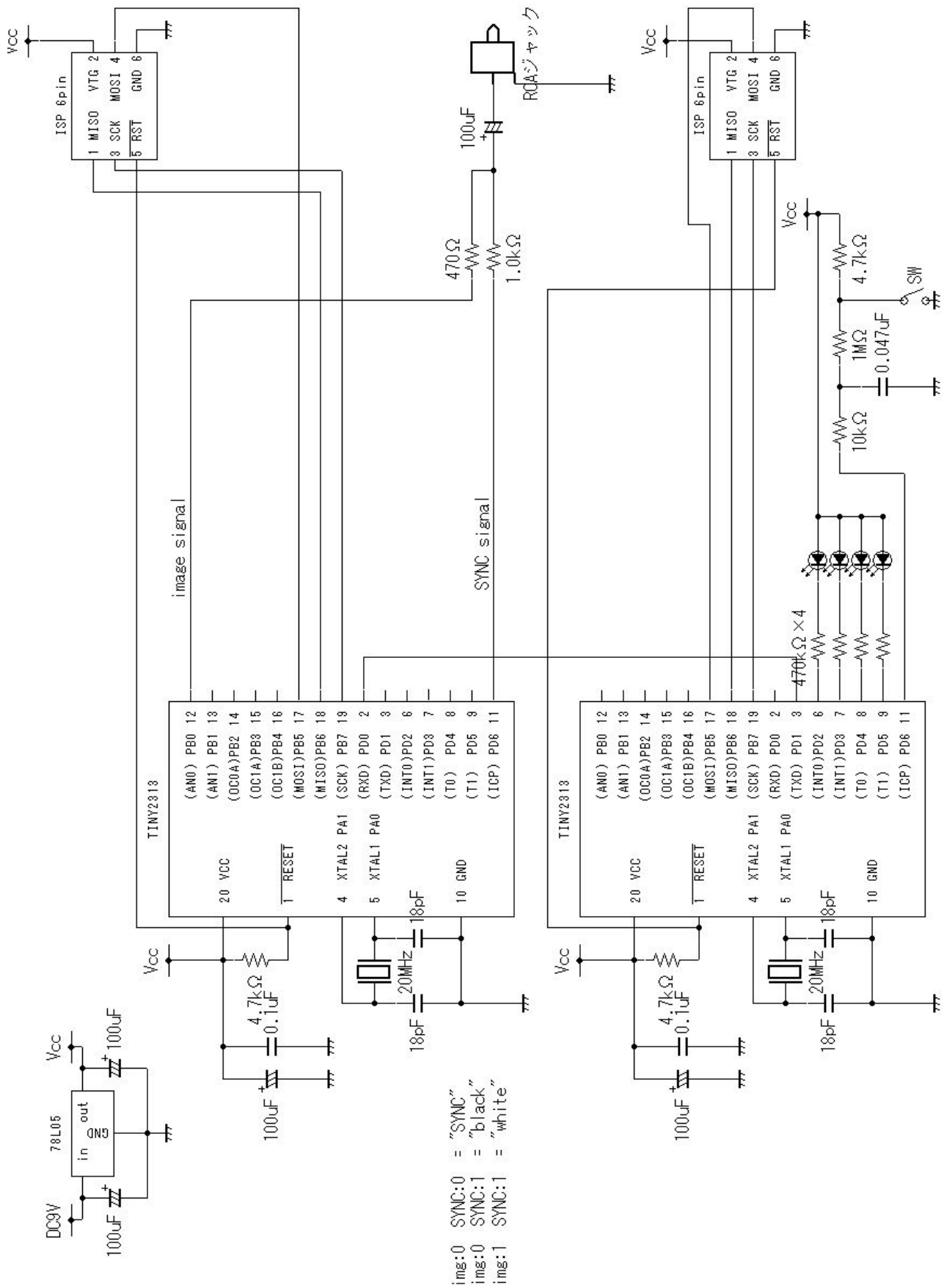
- SMPTE-170Mで規定された電圧（SYNC信号から白までが1Vppで、SYNC信号から黒までが0.3Vpp）に合わせるために、5Vの電圧を抵抗で分圧して実現する。その際の抵抗は上図のとおり470Ω、1kΩ、75Ωの3つを利用する。出力電圧は以下のとおり。（合成抵抗値と電圧の計算式は割愛する）

	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）

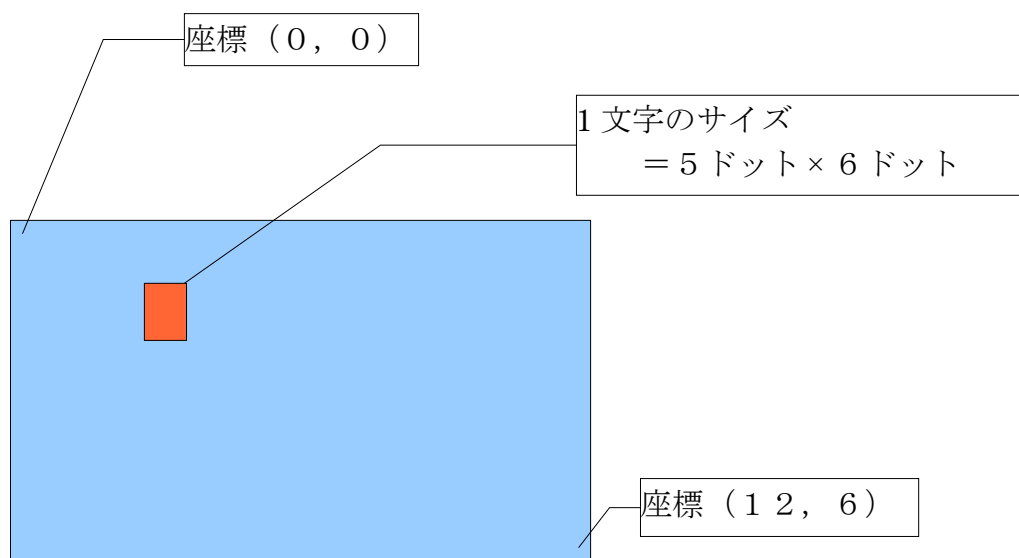
(5) サンプル回路図



### 3. 仕様

#### (1) 表示可能領域

- 文字が配置可能な座標は、左上 (0, 0) ~ 右下 (12, 6) の範囲。
- カーソルが画面右端にある時に表示可能文字コードを受信すると、その位置に文字を表示した後にカーソルは1行下の左端に移動する。
- 最下行の最右端 = (12, 6) にカーソルがあるときに表示可能文字コードを受信すると、その位置に文字を表示した直後にスクロールする。よって、事実上 (12, 6) の位置に文字が表示されることは無い。なおスクロール処理の詳細内容は (3) 特殊文字コードによるスクロールと同じなので、そちらを参照。
- 白黒表示のみ対応。
- カーソル位置には常にカーソルが点滅する。カーソルを常時消灯することは出来ない。



- 文字1個のサイズは5ドット×6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。(2ラインで1ドットを表現している)



#### (2) 使用可能文字種数

- アルファベット、英数字、その他記号を合わせて70種のビットマップをプログラムROM上に格納。(ただし、アルファベットの小文字フォントは搭載しておらず、大文字に変換して表示している。)

- それ以外に特殊文字コードが存在する。特殊文字コードについては（3）を参照。
- 文字コード表に存在しないコードを受信しても無視される。（何も起こらない）
- 文字コードの一覧については、項番6付録（1）キャラクターコード表を参照。

### （3）特殊文字コード

- ベルコード（0 x 07）：PD4端子に約0.5秒間だけ5Vを出力する。（この0.5秒の間にさらにベルコードを受信した場合、2回目のベルコード受信から再度0.5秒をカウントしなおす。この際PD4端子の出力は途切れない。以降、0.5秒以内にベルコードを再受信するたびにカウンター残量は0.5秒に巻き戻される。）なお、ベルコード出力だけではカーソル位置は影響を受けない（画面右端でも改行しないし、画面最下行右端でもスクロールしない）。
- 画面クリア（0 x 0C）：画面全体を空白コードで埋める。
- 改行コード（0 x 0D）：1行下の行の左端にカーソルを移動する（カーソル移動を行うだけで、表示内容には変更がない）。カーソルが最下行にある時（Y軸＝6の時）に改行コードを受信すると、1行分上方向にスクロールし、カーソルの位置は（0，6）に移動する。スクロールの際、最下行の表示内容はクリアされる。
- X軸指定コード（0 x 80～8C）：カーソルのX軸を直接指定する。
- Y軸指定コード（0 x 90～97）：カーソルのY軸を直接指定する。ただし、0 x 97が指定された場合は強制的にスクロールが行われると同時に、Y軸は6にセットされる。この場合のスクロールの際、X軸は変化しない。（これは、通常X軸指定→Y軸指定の順でカーソルを位置付けてから文字表示することに配慮…カーソル指定とスクロールを一度に行えることを実現）

### （4）UART接続仕様

- 1200bps、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バイト送信する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40.15μ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. 使用方法

### (1) 初期化

- ・電源ON時もしくはリセット時に、テキストVRAMに立ち上げメッセージを自動的に表示する。カーソル位置は(0, 0)に初期化される。適宜、ユーザープログラム側から画面クリアコード(0x0C)を送信して立ち上げメッセージを消去する。
- ・また、立ち上げ時の回路上のノイズがUARTに乗って不要データが取り込まれる恐れがある場合は、ユーザープログラムの先頭で画面クリアを行うことによってクリアすること。
- ・テキスト画面表示器表示器が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40.15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+40.15μ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) 文字表示、およびカーソル制御

- ・表示可能文字のコードをUARTにて送ることで、カーソル位置にその文字コードが表示される。表示可能な文字コードの一覧は項番6付録(1)を参照。アルファベットの小文字については、内部で大文字に変換されて表示されることに留意。
- ・カーソルを特定位置に移動するには、X軸、Y軸を指定することで実現する。X軸の指定は0x80~0x8C、Y軸の指定は0x90~0x97を送信することで実現する。ただし、0x97は強制的にスクロールするために確保されたコード。
- ・全画面をクリアするには0x0Cを、改行するには0x0Dを、ベル出力は0x07を送信する。最下行で改行を行うと自動的にスクロールする。
- ・以下にアセンブラ用のサンプルを示す。

```
*****
***** 定義類 *****
*****

***** 定義ファイル *****

.include "tn2313def.inc"

***** レジスタ *****

.def data1 = R16           ;超局所用途の汎用レジスタ
.def out_data = R17       ;UART出力用データ(サブ引渡し用)
.def l_count1 = R18      ;汎用カウンタ1
.def l_count2 = R19      ;汎用カウンタ2
.def l_count3 = R20      ;汎用カウンタ3
```



```

;***** 定数 *****

;*****
;***** リセット処理 *****
;*****

.org 0x00                                ;リセット時のスタート番地
rjmp  RESET                               ;リセットがかかったときに0番地からRESETに相対ジャンプ

RESET:

;*****
;***** 初期化処理 *****
;*****

;***** スタックポインタ初期設定 *****

ldi      data1, RAMEND                    ;data1にスタックポインタのアドレスをセット
out      SPL, data1                       ;スタックポインタにdata1をセット

;***** 初期化処理呼び出し *****

rcall   INITIALIZE

MAIN_LOOP:

;*****
;***** メインループ *****
;*****

;***** 送信処理 *****

ldi      out_data, 0x31 ;1
rcall   PUSH_AND_OUT

ldi      out_data, 0x41 ;A
rcall   PUSH_AND_OUT

ldi      out_data, 0x26 ;&
rcall   PUSH_AND_OUT

ldi      out_data, 0x0c ;cls
rcall   PUSH_AND_OUT

ldi      out_data, 0x20 ;space
rcall   PUSH_AND_OUT

ldi      l_count1, 0
MA_LOOP1:
mov      out_data, l_count1
rcall   PUSH_AND_OUT
inc      l_count1
cpi     l_count1, 0x80

```

```

brne    MA_LOOP1                ;0 から 0x7f まで続けて出力

ldi     out_data, 0x32          ;2
rcall   PUSH_AND_OUT

ldi     out_data, 0x42          ;B
rcall   PUSH_AND_OUT

ldi     out_data, 0x24          ;$
rcall   PUSH_AND_OUT

ldi     out_data, 0x0c          ;cls
rcall   PUSH_AND_OUT

ldi     l_count1, 0
MA_LOOP2:
ldi     out_data, 0x30
rcall   UART_OUT
inc     l_count1
cpi     l_count1, 91
brne    MA_LOOP2                ;91 文字連続して数字の 0 を出力

ldi     l_count1, 0
MA_LOOP3:
ldi     out_data, 0x31
rcall   UART_OUT
inc     l_count1
cpi     l_count1, 91
brne    MA_LOOP3                ;91 文字連続して数字の 1 を出力

ldi     l_count1, 0
MA_LOOP4:
ldi     out_data, 0x32
rcall   UART_OUT
inc     l_count1
cpi     l_count1, 91
brne    MA_LOOP4                ;91 文字連続して数字の 2 を出力

ldi     l_count1, 0
MA_LOOP5:
ldi     out_data, 0x33
rcall   UART_OUT
inc     l_count1
cpi     l_count1, 91
brne    MA_LOOP5                ;91 文字連続して数字の 3 を出力

ldi     l_count1, 0
MA_LOOP6:
ldi     out_data, 0x34
rcall   UART_OUT
inc     l_count1
cpi     l_count1, 91
brne    MA_LOOP6                ;91 文字連続して数字の 4 を出力

ldi     out_data, 0x8B          ;X軸を 11 に
rcall   PUSH_AND_OUT

ldi     out_data, 0x5A          ;Z

```

```

rcall    PUSH_AND_OUT

ldi      out_data, 0x94    ; Y軸を4に
rcall    PUSH_AND_OUT

ldi      out_data, 0x52    ; R
rcall    PUSH_AND_OUT

ldi      out_data, 0x0d    ; crlf
rcall    PUSH_AND_OUT

ldi      out_data, 0x41    ; a
rcall    PUSH_AND_OUT

ldi      out_data, 0x0d    ; crlf
rcall    PUSH_AND_OUT

ldi      out_data, 0x41    ; a
rcall    PUSH_AND_OUT

ldi      out_data, 0x0d    ; crlf
rcall    PUSH_AND_OUT

ldi      out_data, 0x41    ; a
rcall    PUSH_AND_OUT

ldi      out_data, 0x0d    ; crlf
rcall    PUSH_AND_OUT

ldi      out_data, 0x41    ; a
rcall    PUSH_AND_OUT

ldi      out_data, 0x0d    ; crlf
rcall    PUSH_AND_OUT

ldi      out_data, 0x41    ; a
rcall    PUSH_AND_OUT

ldi      out_data, 0x0d    ; crlf
rcall    PUSH_AND_OUT

ldi      out_data, 0x41    ; a
rcall    PUSH_AND_OUT

ldi      out_data, 0x80    ; X軸を0に
rcall    UART_OUT
ldi      out_data, 0x90    ; Y軸を0に
rcall    UART_OUT
ldi      out_data, 0x2A    ; *
rcall    UART_OUT
rcall    BUTTON_PUSH

ldi      out_data, 0x81    ; X軸を1に
rcall    UART_OUT
ldi      out_data, 0x91    ; Y軸を1に
rcall    UART_OUT
ldi      out_data, 0x2A    ; *
rcall    UART_OUT
rcall    BUTTON_PUSH

ldi      out_data, 0x82    ; X軸を2に

```

```

rcall  UART_OUT
ldi     out_data, 0x92  ;Y軸を2に
rcall  UART_OUT
ldi     out_data, 0x2A  ;*
rcall  UART_OUT
rcall  BUTTON_PUSH

ldi     out_data, 0x83  ;X軸を3に
rcall  UART_OUT
ldi     out_data, 0x93  ;Y軸を3に
rcall  UART_OUT
ldi     out_data, 0x2A  ;*
rcall  UART_OUT
rcall  BUTTON_PUSH

ldi     out_data, 0x84  ;X軸を4に
rcall  UART_OUT
ldi     out_data, 0x94  ;Y軸を4に
rcall  UART_OUT
ldi     out_data, 0x2A  ;*
rcall  UART_OUT
rcall  BUTTON_PUSH

ldi     out_data, 0x85  ;X軸を5に
rcall  UART_OUT
ldi     out_data, 0x95  ;Y軸を5に
rcall  UART_OUT
ldi     out_data, 0x2A  ;*
rcall  UART_OUT
rcall  BUTTON_PUSH

ldi     out_data, 0x86  ;X軸を6に
rcall  UART_OUT
ldi     out_data, 0x96  ;Y軸を6に
rcall  UART_OUT
ldi     out_data, 0x2A  ;*
rcall  UART_OUT
rcall  BUTTON_PUSH

ldi     out_data, 0x0c  ;cls
rcall  PUSH_AND_OUT

ldi     l_count1, 0
MA_LOOP7:
mov     out_data, l_count1
rcall  UART_OUT
inc     l_count1
cpi     l_count1, 0x80
brne   MA_LOOP7      ;0 から 0x7f まで続けて出力
rcall  BUTTON_PUSH

ldi     out_data, 0x0c  ;cls
rcall  PUSH_AND_OUT

rjmp   MAIN_LOOP      ;LOOP に相対ジャンプして無限ループ

```

```

;*****
;***** サブルーチン *****
;*****

;*****          *****
;***** <初期化処理> *****
;*****          *****
INITIALIZE:

;***** 入出力ポートの設定 *****

ldi          data1, 0b10111111      ;data1 を第6ビット以外立てる
out          DDRD, data1            ;P D 6 以外すべて出力にセット

ldi          data1, 0b11111111      ;data1 をすべて立てる
out          DDRB, data1            ;P B をすべて出力にセット

;***** 初期値の設定 *****

;***** RS232C 初期化 *****

ldi          data1, 0
out          UCSRB, data1           ;まず、一旦ディゼーブルに設定
out          UCSRA, data1           ;すべてクリア

ldi          data1, ((1<<UCSZ1) | (1<<UCSZ0))
out          UCSRC, data1           ;8ビット、ノンパリ、ストップ1

ldi          data1, 0x03
out          UBRRH, data1           ;上位4ビット
ldi          data1, 0xfc
out          UBRL, data1           ;下位8ビット
;UBRR = 0x3f7 = 1015 (=1230BAUD @20MHz)
;UBRR = 0x411 = 1041 (=1200BAUD @20MHz)

ldi          data1, (1<<TXEN)
out          UCSRB, data1           ;設定完了後に、受信イネーブルを設定

;***** 立ち上げまでにタイムラグを作る *****

ldi          l_count1, 255
IN_LOOP1:
ldi          l_count2, 255
IN_LOOP2:
nop
nop
nop
nop
nop

```

```

dec          l_count2
cpi          l_count2, 0
brne        IN_LOOP2

dec          l_count1
cpi          l_count1, 0
brne        IN_LOOP1          ;1/100 程度のタイムラグ

ret          ;初期化終わり

;*****          *****
;***** <キー押下& o f f チェック> *****
;*****          *****
BUTTON_PUSH:
;ボタンが押下されるまで待ち、
;その後離されるまで待つ

;***** <step1> *****

BP_LOOP1:
in          data1, PIND
andi       data1, 0b01000000    ;6 ビット目が立っているかチェック
brne      BP_LOOP1             ;ボタンが押されてなければ繰り返す
;ボタンは負理論：押されなければ1が入力

;***** <step2> *****

sbi        PORTD, 2

BP_LOOP2:
in          data1, PIND
andi       data1, 0b01000000    ;6 ビット目が立っているかチェック
breq      BP_LOOP2             ;ボタンが押されていれば繰り返す
;ボタンは負理論：押されていれば0が入力

cbi        PORTD, 2

ret          ;戻る

;*****          *****
;***** <UART 出力サブ> *****
;*****          *****
UART_OUT:
;UDRE が立つまで待ってからデータを出力

UO_LOOP1:
in          data1, UCSRA          ;UCSRA を読み込む
andi       data1, (1<<UDRE)      ;UDRE が立っているかチェック
breq      UO_LOOP1             ;立つまで繰り返す

out        UDR, out_data        ;データを出力

ret          ;戻る

```

```

;*****          *****
;***** < ボタン+出力 > *****
;*****          *****
PUSH_AND_OUT:
; ボタン押下を待ってからデータ出力

rcall  BUTTON_PUSH      ; ボタンチェック 呼び出し
rcall  UART_OUT         ; データ出力 呼び出し

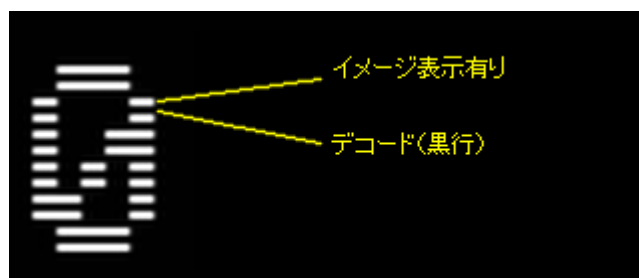
ret                    ; 戻る

```

## 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。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 $\mu$ 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ベルコードの出力時間計測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）ように設計しており、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) 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) キャラクターコード表

- ・以下にキャラクターコード表を示す。

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	-	-	sp	0	@	P	'	p	x0	y0	-	-	-	-	-	-
1	-	-	!	1	A	Q	a	q	x1	y1	-	-	-	-	-	-
2	-	-	"	2	B	R	b	r	x2	y2	-	-	-	-	-	-
3	-	-	#	3	C	S	c	s	x3	y3	-	-	-	-	-	-
4	-	-	\$	4	D	T	d	t	x4	y4	-	-	-	-	-	-
5	-	-	%	5	E	U	e	u	x5	y5	-	-	-	-	-	-
6	-	-	&	6	F	V	f	v	x6	y6	-	-	-	-	-	-
7	BL	-	'	7	G	W	g	w	x7	y7	-	-	-	-	-	-
8	-	-	(	8	H	X	h	x	x8	-	-	-	-	-	-	-
9	-	-	)	9	I	Y	i	y	x9	-	-	-	-	-	-	-
A	-	-	*	:	J	Z	j	z	x10	-	-	-	-	-	-	-
B	-	-	+	;	K	[	k	{	x11	-	-	-	-	-	-	-
C	CL	-	,	<	L	¥	l		x12	-	-	-	-	-	-	-
D	CR	-	-	=	M	]	m	}	-	-	-	-	-	-	-	-
E	-	-	.	>	N	^	n	→	-	-	-	-	-	-	-	-
F	-	-	/	?	O		o	←	-	-	-	-	-	-	-	-

- : 表示可能文字 (spはスペース)
- : 制御文字 (y 7 : 改行用)
- : 表示可能文字 (大文字に変換して表示)