ASCIIコードに 関して

L \ H	0	1	2	3	4	5	6	7
0	NUL	DE		0	@	P		p
1	SOH	D1	!	1	A	Q	a	q
2	STX	D2	″	2	B	R	b	r
3	ETX	D3	#	3	C	S	c	s
4	EOT	D4	$	4	D	T	d	t
5	ENQ	NAK	%	5	E	U	e	u
6	ACK	SN	&	6	F	V	f	v
7	BEL	ETB	’	7	G	W	g	w
8	BS	CN	(	8	H	X	h	x
9	HT	EM	)	9	I	Y	i	y
A	LF	SB	*	:	J	Z	j	z
B	HM	EC	+	;	K	[	k	{
C	CL		,	<	L	¥	l	|
D	CR		−	=	M	]	m	}
E	SO		.	>	N	^	n	~
F	SI		/	?	O	_	o	DEL

左のコード表が、ほぼ ASCIIコード表です。 ほぼと 付けたのは、日本で、文字の形を 局部的に変えている箇所があるからです。 ASCIIコードは、原案が アメリカで作成されたので、7bit のコード体系です。 この中で、00h ～ 1Fhは、データ通信における制御コードなので、文字フォントはありません。7Fhの DEL コードも 制御コードです。

この、ASCIIコード表の読み方は、左上の 肌色の Hと Lについて、Hは 一番上の 0 ～ 7の 数字を指しており、7bitコードの 上位 3bitを意味します。 そして、Lの下の 0 ～ Fの 16進数は、7bitコードの 下位 4bitを 指しています。

ちなみに、文字の形を変えているコードとは、5Chの ¥ コードです。アメリカでは 逆スラッシュ ＼ です。 ちなみに C言語で printf関数の フォーマット文字列の ¥ シーケンスが ありますが アメリカでは ＼シーケンスに なります。

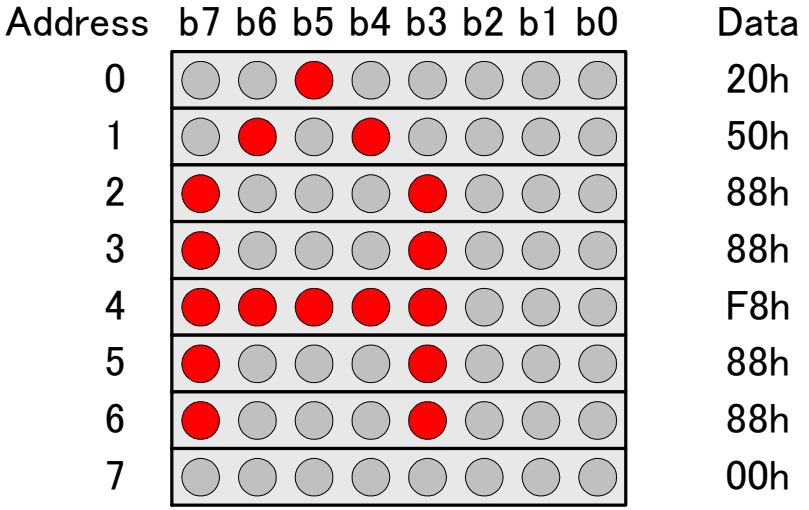
それと、テキストファイルでよく使う TABキーの コードは 09h HT (ホリゾンタルTAB)コードです。 そして、ENTERキーで改行しますが、MS-DOSでは、0Dhの CRコードと 0Ahの LFコードの 2つを組みにして 使用します。 (MS-Windowsの場合は、大体同じですが インターネットのファイルの場合、コード体系が異なる場合があります。) ちなみに Linuxのテキストファイルは、改行コードは、0ah LFコード だけです。

今回の ASCII文字フォントに 関して

今回の目的は、横 32 Dot、縦 8 Dot の 横長の Dot マトリクス LED表示器に 文字を 表示することなので、文字フォントは、5x7 Dotの文字フォントを使用します。 遥か昔の 8bitのパソコン時代は、普通に使われていたフォントでしたが、今はパソコンでは 殆ど使われなくなって しまいました。 ローエンドマイコンでは漢字を扱う事は まずないので 5x7のフォントは16桁2行の液晶の文字表示器などで まだ使われています。

今回は、32x8 Dotの DotマトリクスLED表示器に、ASCII文字を出したいので、文字のフォントデータ(文字の形のデータ)と、文字フォントを 表示するプログラムを 自前で マイコンに組み込む必要が あります。 因みに、文字サイズは、5x7 Dotですが、データサイズは、8x8 Dotサイズ(8 byte)で、持ちます。

表示フォント(A)

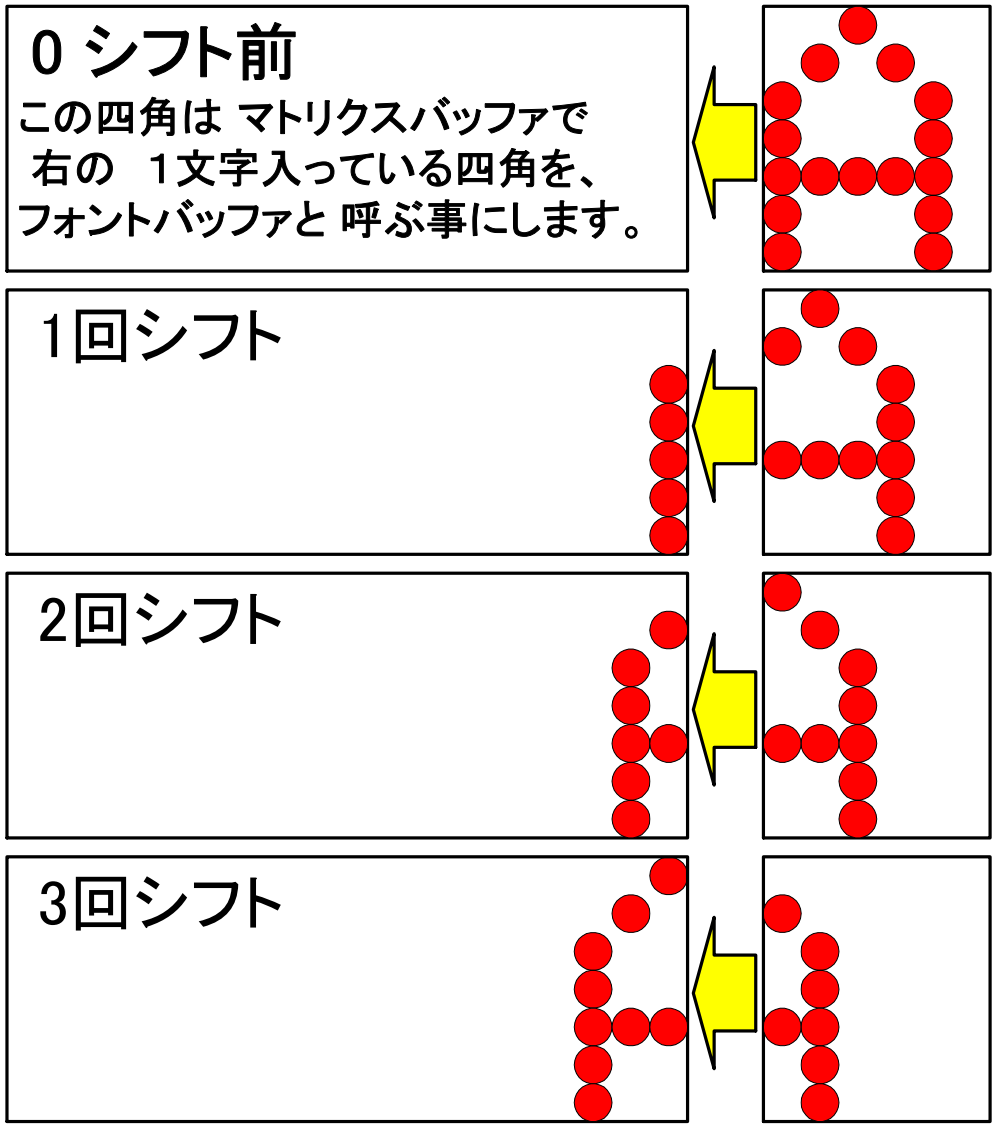


上記の図は、文字 A の フォントデータです。この場合 横 8bitの バイトデータ 8個なので 8 byteの配列データとして扱えます。 左のアドレスは、配列の添え字と見なす事が出来ます。各 byteデータの bit 並びは、左端が bit7 で、右端が bit 0 です。 5x7の 文字フォントは 左上に寄せた状態でデザインされています。フォントのDataは、上から 20h、50h、88h、88h、F8h、88h、88h、00h に、なっています。

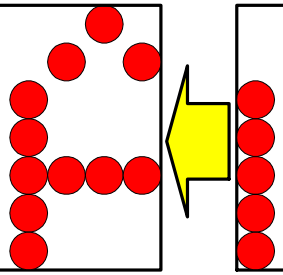
文字フォントの ビットシフトに関して

MAX7219の データ転送は、ビットシリアルで転送し、内部に 16bitのシフトレジスタを持っていて、更に隣のMAX7219と連結しているのでそのシフトレジスタを使って横に、右から左に流れるように文字フォントを表示出来そうな気がしますが出来ません。 何故かというと、下位 8 bit は 表示データですが、上位 8bit内の下位 4bitは MAX7219の レジスタアドレスだからです。 よってアドレス情報は、16bit中の b11 ～ b8 に きっちり合わせる形でシフトしなければならないためです。

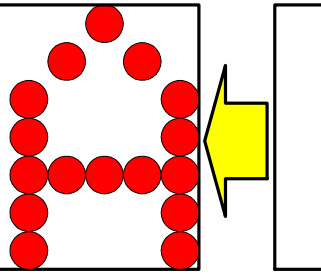
では、どのように 文字フォントを 右から左にドット単位で、流れるように表示するかというとマイコン内で、32x8 bitの 配列データを用意して、その中に表示したいビットデータを、配置してそのデータを 横方向バイト単位に、4バイト連続で転送する動作を 縦8回繰り返す事になります。 右に図で示します。



4回シフト

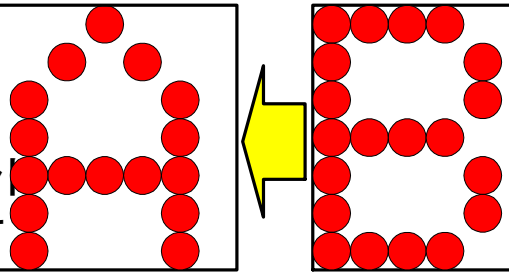


5回シフト

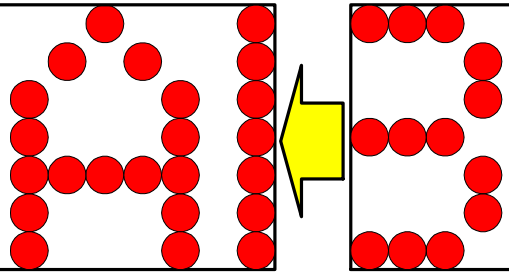


6回シフト

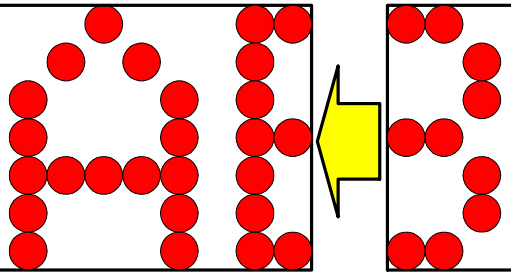
このタイミングで、Bの
文字フォントを　フォント
バッファにセットします



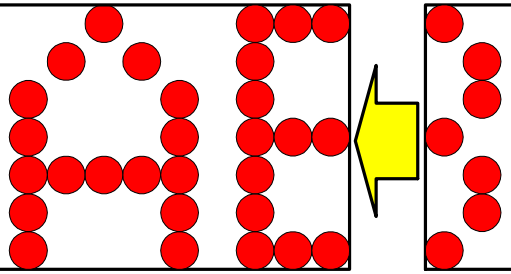
7回シフト



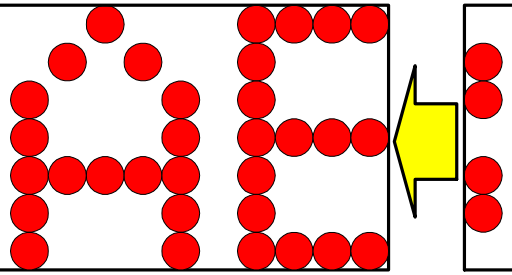
8回シフト



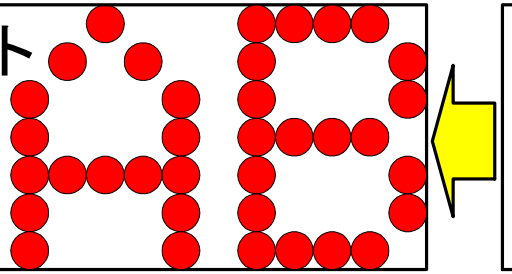
9回シフト

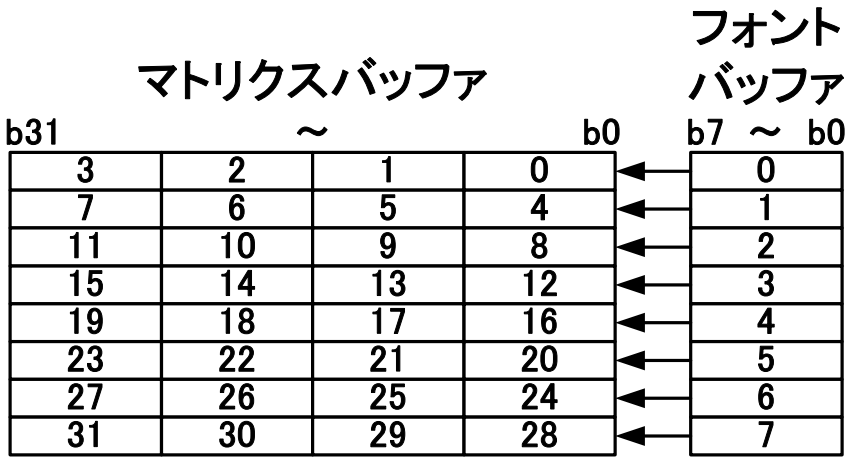


10回シフト



11回シフト





フォントバッファのバイト並びは、上から 0 ～ 7に なっている事は、先ほど説明しました。 左のマトリクスバッファの バイト並びは、右から左に向かい アドレスが増えて行きます。

これは、マトリクスバッファの 右端が b0 で左端が b31 の 32 bit シフトレジスタとして扱います。 データの元は、フォントバッファから bit 転送するので、計 40bit の シフトレジスタとしてシフト演算を行います。 マトリクスバッファのアドレス配置が ３２１０と なっているのは、実は、R8Cマイコンが 上位バイト 下位バイトの並びが、若い番地の方に下位バイトが来る リトルエンディアンだからです。

それと、MAX7219を使った 8x8LEDモジュールを ４つ連結しているモジュールも、右側にコネクタを持って来ると、右端のドットが b0 になり、左端のドットが b31 になります。

但し、マトリクスバッファの 横 ４個のバイトデータを転送する順序は、アドレス順は ３２１０ の順に転送する事になります。 表現を変えるとMAX7219に転送する データは
最初に転送するデータ: 一番左に表示する 8ドットデータです。
２番目に転送するデータ: 左から 2番目に表示する 8ドットデータです。
３番目に転送するデータ: 左から 3番目に表示する 8ドットデータです。
４番目に転送するデータ: 左から 4番目に表示する 8ドットデータです。