

CGRAM活用(キャラクタ表示LCD)

概要

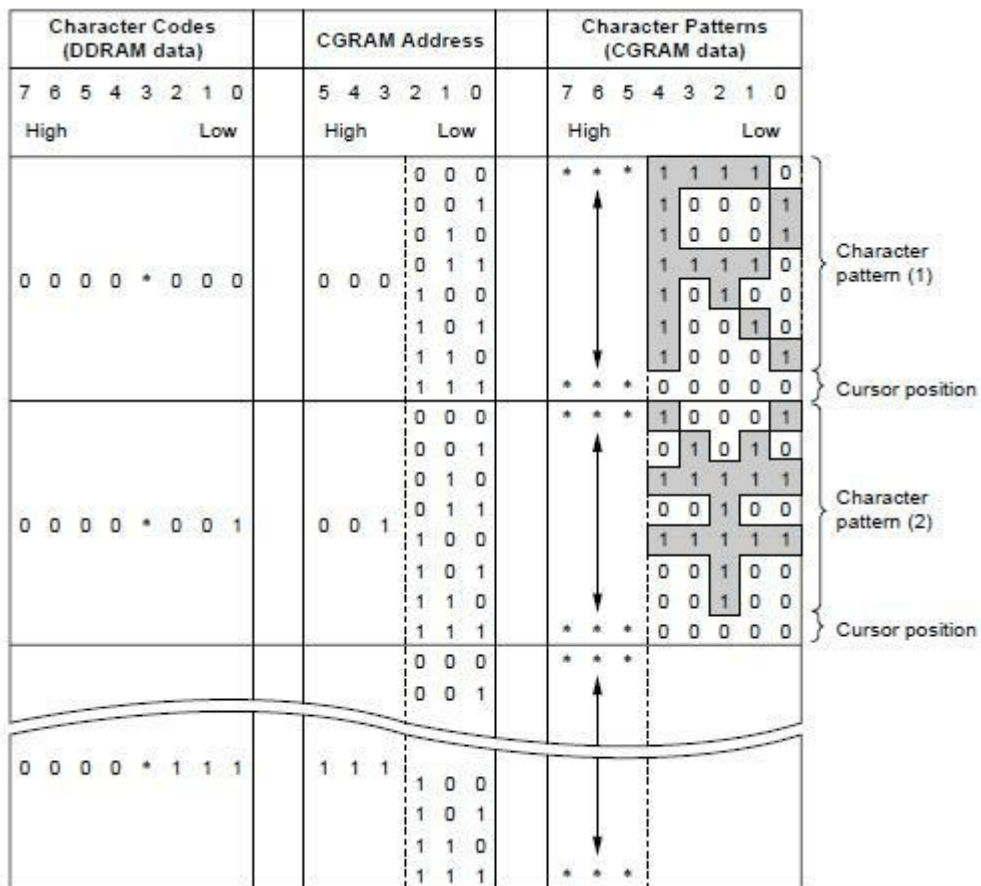
LCD(Liquid Crystal Display:液晶ディスプレイ)にはCGRAM(Character Generator RAM)が用意されており、これを使うことによって、独自のキャラクタを表示させることができます。今回は、7チャンネル分のアナログデータを、LCDにバー表示させてみました。

動作原理

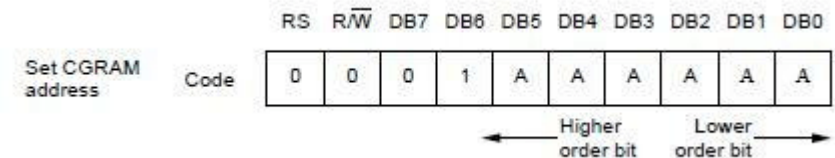
<キャラクタコードとキャラクタパターンの関係> 独自のキャラクタは、キャラクタコード(0x00~0x07)に8個分割り当てられています。

Lower 4 bits	Upper 4 bits	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
xxxx0000	CG RAM (1)			0	Q	P	`	P				-	タ	ミ	α	ρ	
xxxx0001	(2)		!	1	A	Q	a	q				。	ア	チ	△	ä	q
xxxx0010	(3)		"	2	B	R	b	r				「	イ	ツ	×	β	θ
xxxx0011	(4)		#	3	C	S	c	s				」	ウ	テ	モ	ε	∞
xxxx0100	(5)		\$	4	D	T	d	t				、	エ	ト	ヤ	μ	Ω
xxxx0101	(6)		%	5	E	U	e	u				・	オ	ナ	ユ	σ	Ü
xxxx0110	(7)		&	6	F	V	f	v				ヲ	カ	ニ	ヨ	ρ	Σ
xxxx0111	(8)		'	7	G	W	g	w				ア	キ	ヌ	ラ	g	π
xxxx1000	(1)		(8	H	X	h	x				ィ	ク	ネ	リ	フ	×
xxxx1001	(2))	9	I	Y	i	y				ウ	ケ	ル	レ	'	y
xxxx1010	(3)		*	:	J	Z	j	z				エ	コ	ハ	レ	j	チ
xxxx1011	(4)		+	;	K	[k	{				オ	サ	ヒ	ロ	*	斤
xxxx1100	(5)		,	<	L	¥	l	l				カ	シ	フ	ワ	φ	円
xxxx1101	(6)		-	=	M]	m	}				ユ	ヌ	ヘ	ン	も	÷
xxxx1110	(7)		.	>	N	^	n	→				ヨ	セ	ホ	ッ	ñ	
xxxx1111	(8)		/	?	O	_	o	←				ッ	ソ	マ	°	ö	■

<キャラクタコード(0x00~0x07)とCGRAMアドレスとCGRAMデータの関係> キャラクタデータは、8バイト×5個で一組となります。次の例では、キャラクタコード(0x00)に、“R”という文字、キャラクタコード(0x01)に、“ / ”という文字を登録した場合の内容です。



<制御コマンド> セットCGRAMアドレスコマンドで、CGRAMのアドレスを指定して、以降は順次キャラ

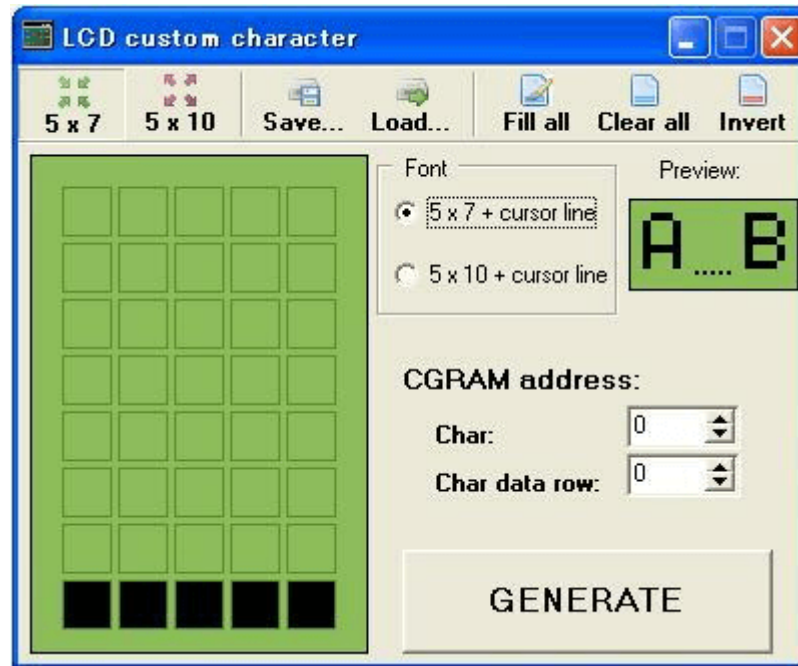


クタデータを書き込んでいきます。

<実際の表示> 通常の文字表示と同じです。

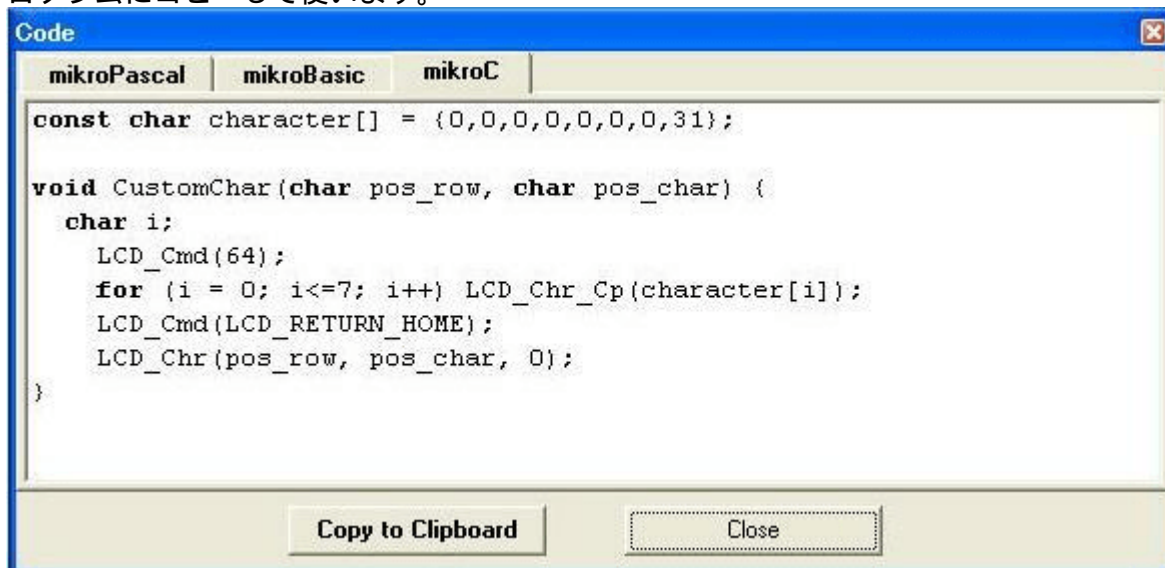
- 標準キャラクタの表示
LCD_Chr(1, 1, 'A');
LCD_Chr(1, 2, 0x30);
- 独自キャラクタの表示
LCD_Chr(2, 1, 0x00);
LCD_Chr(2, 2, 0x07);

<mikroC付属のキャラクタデータ生成ツール> 手計算でキャラクタデータを設定しても良いのですが、結構面倒くさい作業になります。このツールを使うことにより、視覚的にキャラクタデータを設定する



ことが出来るので便利です。

GENERATE(生成)ボタンを押すとキャラクタデータの登録&表示関数を自動的に生成します。これをプログラムにコピーして使います。

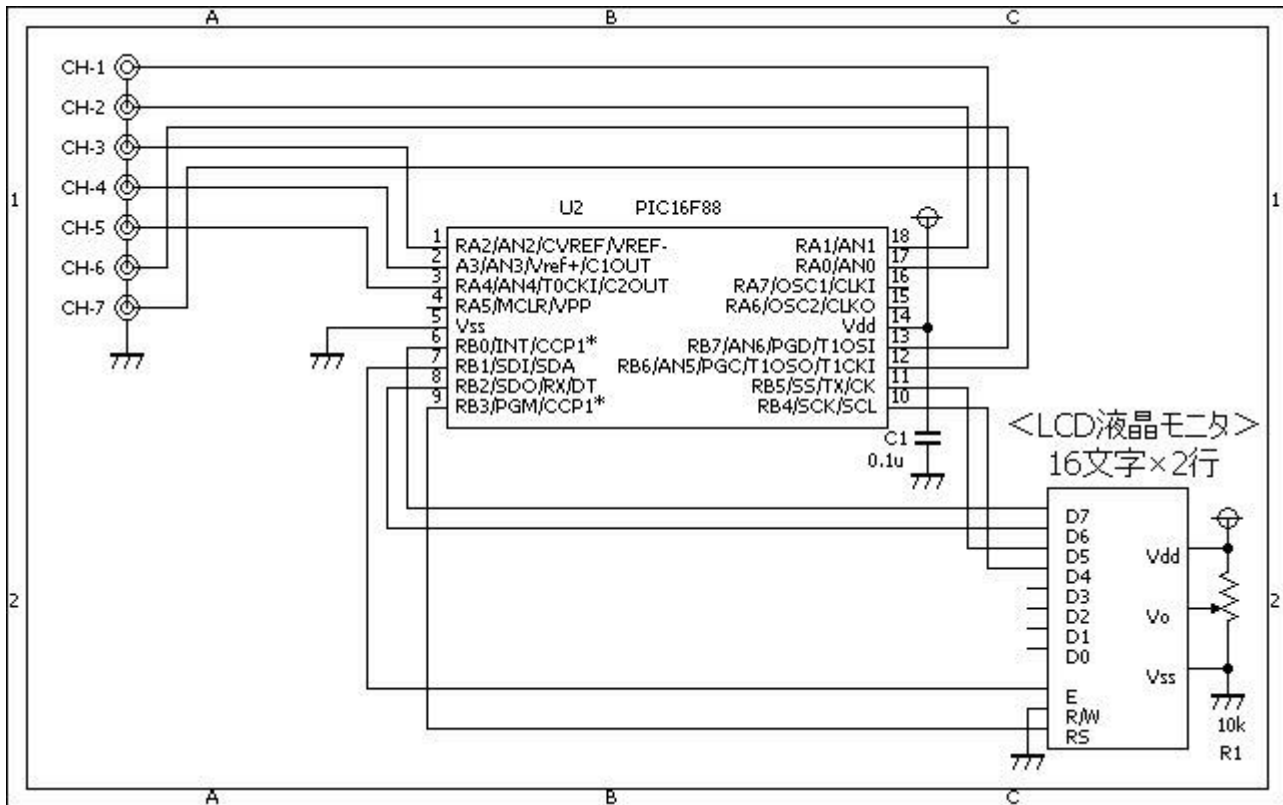


```
Code
mikroPascal  mikroBasic  mikroC
const char character[] = {0,0,0,0,0,0,0,31};

void CustomChar(char pos_row, char pos_char) {
    char i;
    LCD_Cmd(64);
    for (i = 0; i<=7; i++) LCD_Chr_Cp(character[i]);
    LCD_Cmd(LCD_RETURN_HOME);
    LCD_Chr(pos_row, pos_char, 0);
}
```

Copy to Clipboard Close

回路図



ソースコード

[cgramTest.c](#)

```
//*****
*
/*
□□□□□□□□活用（キャラクタ表示□□□□）
*/
//*****
*

const char character1[] = {0,0,0,0,0,0,0,31};
const char character2[] = {0,0,0,0,0,0,0,31,31};
const char character3[] = {0,0,0,0,0,0,31,31,31};
const char character4[] = {0,0,0,0,31,31,31,31};
const char character5[] = {0,0,0,31,31,31,31,31};
const char character6[] = {0,0,31,31,31,31,31,31};
const char character7[] = {0,31,31,31,31,31,31,31};
const char character8[] = {31,31,31,31,31,31,31,31};

void RegistCustomChar()
{
    char i;
    //
    LCD_Cmd(64);
    for (i = 0; i<=7; i++) {
```

```
    LCD_Chr_Cp(character1[i]);
}
for (i = 0; i<=7; i++) {
    LCD_Chr_Cp(character2[i]);
}
for (i = 0; i<=7; i++) {
    LCD_Chr_Cp(character3[i]);
}
for (i = 0; i<=7; i++) {
    LCD_Chr_Cp(character4[i]);
}
for (i = 0; i<=7; i++) {
    LCD_Chr_Cp(character5[i]);
}
for (i = 0; i<=7; i++) {
    LCD_Chr_Cp(character6[i]);
}
for (i = 0; i<=7; i++) {
    LCD_Chr_Cp(character7[i]);
}
for (i = 0; i<=7; i++) {
    LCD_Chr_Cp(character8[i]);
}
    LCD_Cmd(LCD_RETURN_HOME);
}

//*****
*

void BarDisp(char index, int dat)
{
    switch (dat / 64) {
    case 0:
        Lcd_Chr(1, index, ' ');
        Lcd_Chr(2, index, 0);
        break;
    case 1:
        Lcd_Chr(1, index, ' ');
        Lcd_Chr(2, index, 1);
        break;
    case 2:
        Lcd_Chr(1, index, ' ');
        Lcd_Chr(2, index, 2);
        break;
    case 3:
        Lcd_Chr(1, index, ' ');
        Lcd_Chr(2, index, 3);
        break;
    case 4:
        Lcd_Chr(1, index, ' ');
        Lcd_Chr(2, index, 4);

```

```
        break;
    case 5:
        Lcd_Chr(1, index, ' ');
        Lcd_Chr(2, index, 5);
        break;
    case 6:
        Lcd_Chr(1, index, ' ');
        Lcd_Chr(2, index, 6);
        break;
    case 7:
        Lcd_Chr(1, index, ' ');
        Lcd_Chr(2, index, 7);
        break;
    case 8:
        Lcd_Chr(1, index, 0);
        Lcd_Chr(2, index, 7);
        break;
    case 9:
        Lcd_Chr(1, index, 1);
        Lcd_Chr(2, index, 7);
        break;
    case 10:
        Lcd_Chr(1, index, 2);
        Lcd_Chr(2, index, 7);
        break;
    case 11:
        Lcd_Chr(1, index, 3);
        Lcd_Chr(2, index, 7);
        break;
    case 12:
        Lcd_Chr(1, index, 4);
        Lcd_Chr(2, index, 7);
        break;
    case 13:
        Lcd_Chr(1, index, 5);
        Lcd_Chr(2, index, 7);
        break;
    case 14:
        Lcd_Chr(1, index, 6);
        Lcd_Chr(2, index, 7);
        break;
    case 15:
        Lcd_Chr(1, index, 7);
        Lcd_Chr(2, index, 7);
        break;
    }
}

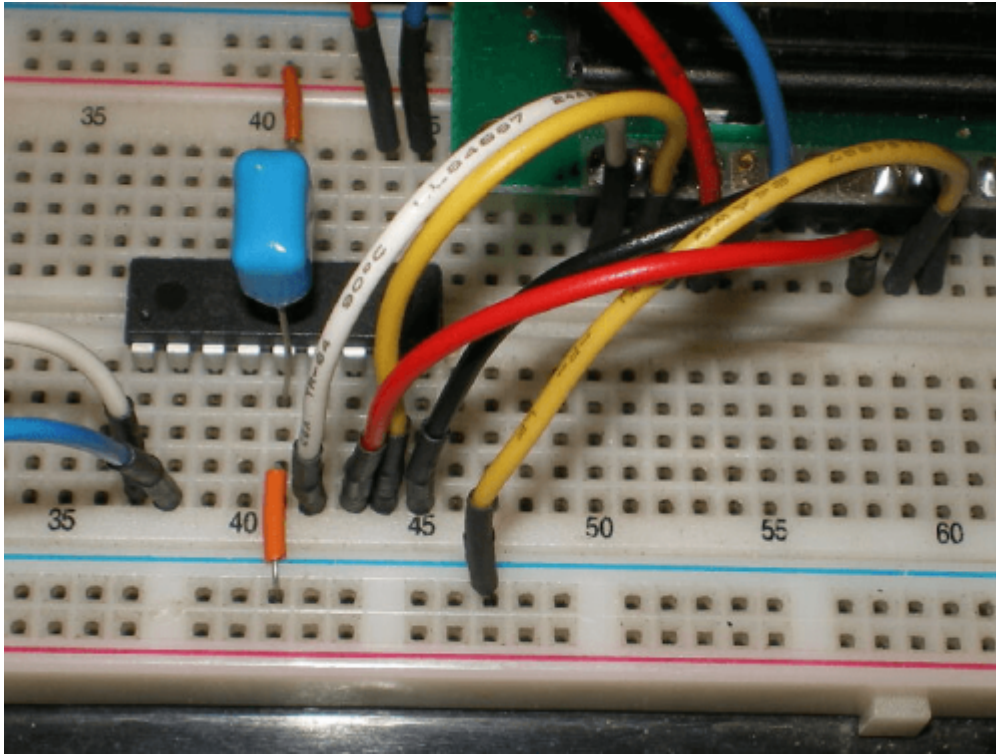
//*****
*
```

```
void main()
{
    int    dat0, dat1, dat2, dat3, dat4, dat5, dat6;
    char   buf[6];
    //
    OSCCON = 0b01110000;           // クロックは8Mhz
    CMCON  = 0b00000111;           // コンパレータは使用しない。
    ANSEL  = 0b01111111;           // □□□変換を使用する。
    TRISA  = 0b11111111;
    TRISB  = 0b11000000;
    //□□□の初期化
    Lcd_Config(&PORTB, 3, 1, 2, 0, 2, 5, 4);
    RegistCustomChar();
    Lcd_Cmd(LCD_CURSOR_OFF);
    Lcd_Cmd(LCD_CLEAR);
    //
    while (1) {
        BarDisp(1, dat0 = Adc_Read(0));
        BarDisp(2, dat1 = Adc_Read(1));
        BarDisp(3, dat2 = Adc_Read(2));
        BarDisp(4, dat3 = Adc_Read(3));
        BarDisp(5, dat4 = Adc_Read(4));
        BarDisp(6, dat5 = Adc_Read(5));
        BarDisp(7, dat6 = Adc_Read(6));
        Lcd_Out(1, 9, "1:");
        WordToStr(dat0 * 4.8828125, buf);
        Lcd_Out(1, 11, &buf[1]);
        Lcd_Out(1, 15, "mV");
        Lcd_Out(2, 9, "2:");
        WordToStr(dat1 * 4.8828125, buf);
        Lcd_Out(2, 11, &buf[1]);
        Lcd_Out(2, 15, "mV");
        Delay_ms(500);
        //
        BarDisp(1, dat0 = Adc_Read(0));
        BarDisp(2, dat1 = Adc_Read(1));
        BarDisp(3, dat2 = Adc_Read(2));
        BarDisp(4, dat3 = Adc_Read(3));
        BarDisp(5, dat4 = Adc_Read(4));
        BarDisp(6, dat5 = Adc_Read(5));
        BarDisp(7, dat6 = Adc_Read(6));
        Lcd_Out(1, 9, "3:");
        WordToStr(dat2 * 4.8828125, buf);
        Lcd_Out(1, 11, &buf[1]);
        Lcd_Out(1, 15, "mV");
        Lcd_Out(2, 9, "4:");
        WordToStr(dat3 * 4.8828125, buf);
        Lcd_Out(2, 11, &buf[1]);
        Lcd_Out(2, 15, "mV");
        Delay_ms(500);
    }
}
```

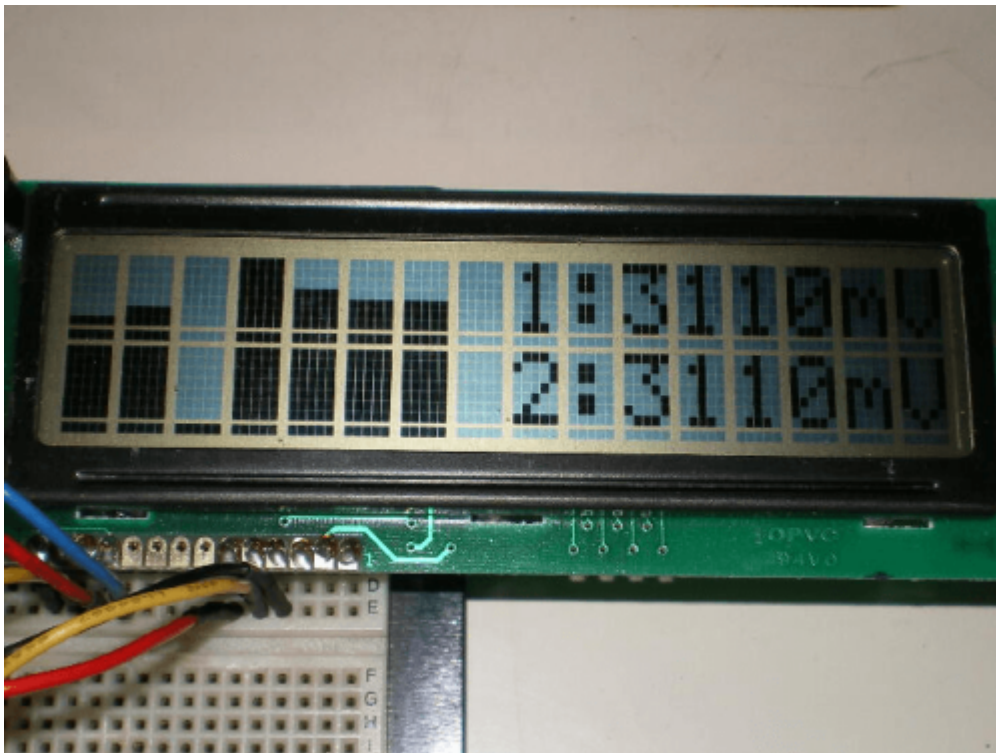
```
//
BarDisp(1, dat0 = Adc_Read(0));
BarDisp(2, dat1 = Adc_Read(1));
BarDisp(3, dat2 = Adc_Read(2));
BarDisp(4, dat3 = Adc_Read(3));
BarDisp(5, dat4 = Adc_Read(4));
BarDisp(6, dat5 = Adc_Read(5));
BarDisp(7, dat6 = Adc_Read(6));
Lcd_Out(1, 9, "5:");
WordToStr(dat4 * 4.8828125, buf);
Lcd_Out(1, 11, &buf[1]);
Lcd_Out(1, 15, "mV");
Lcd_Out(2, 9, "6:");
WordToStr(dat5 * 4.8828125, buf);
Lcd_Out(2, 11, &buf[1]);
Lcd_Out(2, 15, "mV");
Delay_ms(500);
//
BarDisp(1, dat0 = Adc_Read(0));
BarDisp(2, dat1 = Adc_Read(1));
BarDisp(3, dat2 = Adc_Read(2));
BarDisp(4, dat3 = Adc_Read(3));
BarDisp(5, dat4 = Adc_Read(4));
BarDisp(6, dat5 = Adc_Read(5));
BarDisp(7, dat6 = Adc_Read(6));
Lcd_Out(1, 9, "7:");
WordToStr(dat6 * 4.8828125, buf);
Lcd_Out(1, 11, &buf[1]);
Lcd_Out(1, 15, "mV");
Lcd_Out(2, 9, "      ");
Delay_ms(500);
//
}
} //~!

//*****
*
```

動作確認



左側:アナログデータのバー表示(CH-1~CH-7) 右側:アナログデータの数値表示(CH-1とCH-2)



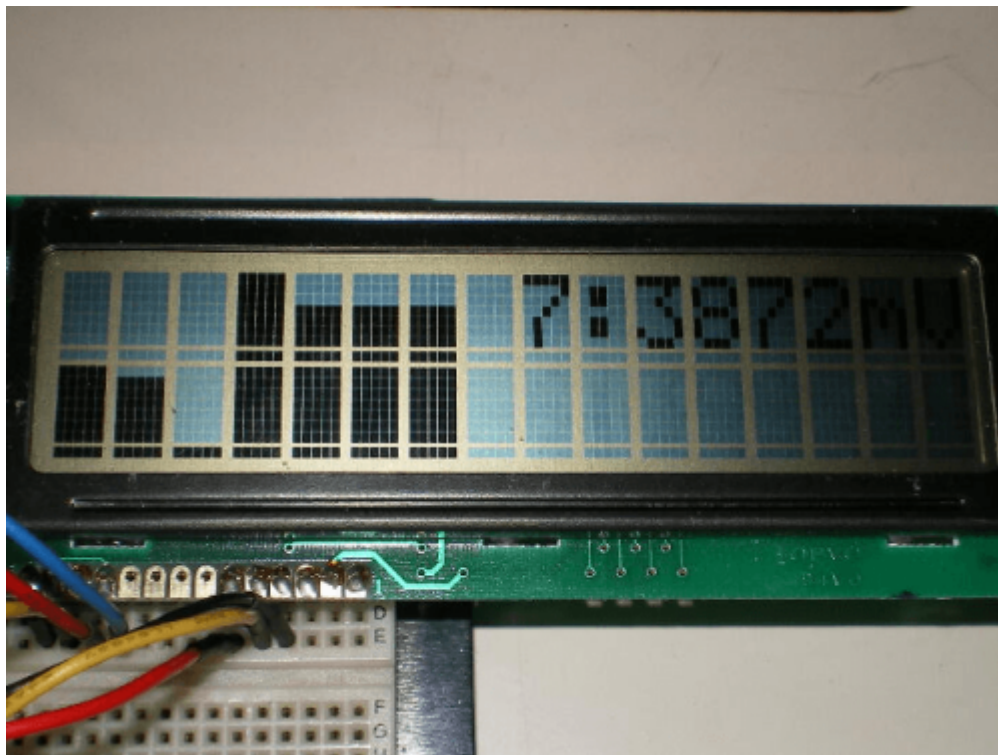
左側:アナログデータのバー表示(CH-1~CH-7) 右側:アナログデータの数値表示(CH-3とCH-4)



左側:アナログデータのバー表示(CH-1~CH-7) 右側:アナログデータの数値表示(CH-5とCH-6)



左側:アナログデータのバー表示(CH-1~CH-7) 右側:アナログデータの数値表示(CH-7)



From:
<http://www.deepsky.jp/wiki/> - うごくといいな

Permanent link:
<http://www.deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:103&rev=1588205260>

Last update: **2025/10/17 14:27**

