

液晶搭載マイコン・モジュール,使いこなしていただけ ているでしょうか.店頭で『すぐに使える!液晶搭載マイ コン・モジュール』(以下,本書)を購入し,パソコンに USBケーブルを接続し,LEDが点滅することを確認でき たでしょうか.はんだ付け不要で,「とりあえず」動作する というコンセプトで,この液晶搭載マイコン・モジュール は作られています.そしてWebサイトで配布されている 開発ツールとサンプル・プログラムをダウンロードし, LCD表示やスイッチ入力,最後のブロック崩しのサンプ ルまで自力でコンパイル&ダウンロードできたでしょうか. ここまではCQ出版社および筆者の動作確認がとれている ものなので,誰でも動作できるようになっています.

そこからさらに一歩進んで、オリジナルのプログラムを 書いているでしょうか. せっかくの液晶搭載マイコンです ので、自分の好きな絵を描いてみたいと思いませんか? 今回はその方法について解説を行います.本書 第5章に掲 載されている液晶表示のサンプル・プログラムを改造して 自分で作成した任意のデータを表示できるようにします.

1. 搭載しているマイコンの特徴

本書にはセイコーエプソン社の S1C17702 マイコンと, 解像度 72 ドット× 32 ドットの液晶画面 (LCD) が搭載され た基板が付属しています.同マイコンはセイコーエプソン 社オリジナルの CPU コアである C17 マイコン・アーキテ クチャを採用しています.また同マイコンは CPU コアだ けでなく LCD コントローラを内蔵しているのが特徴です. そのため, CPU から出力されている信号線を液晶に直結 するだけで画面表示ができるという特徴があります.

今回はこの液晶画面の表示を行ってみましょう.

2. 下準備:プロジェクトのコピー

● 既存のプロジェクトをコピーして使う

液晶表示プログラムを0から作るのは大変なので,すで に動いているプログラムを元に作るのが得策です.プログ ラミングでは,このような「差分開発」によって工期を短縮 することができます.

幸い,今回はLCDの表示を行うLCD_Testがサンプ ル・プログラムとして公開されているので,これを元に自 分の好きな画面を表示してみましょう.

● LCD_Test の動作確認

まずは Web ページで公開されている,第5章のサンプ ル・プログラム LCD_Test を C:¥EPSON¥C17WBIF¥ eclipse¥workspace に展開し,ビルドします.zipファ イルを展開し,File → Import でインポートします.その 後,Project → Build Project でビルドした後,Run → Externel Tools → Externel Tools Configuration でマイコ ン基板へ書き込みます.このあたりの手順は本書で解説さ れている通りです.

● Eclipse でのプロジェクトのコピー方法

ここからは自分用のプロジェクトLCD_Test2を作りま す.元のLCD_Testは残しておきたいので,プロジェクト をコピーして名前を変更します.

そこで、ディレクトリ LCD_Test を LCD_Test2 という 名前でいったんデスクトップに丸ごとコピーして File → Import します. しかし、「Some projects were hidden because they exist in the workspace directory」という警 告が発生し、インポートできません. どうやらディレクト リ名を変えただけでは LCD_Test と同じプロジェクトと認 識されてインポートできないようです.

まずは自由自在に画面表示を行おう 液晶搭載マイコン・モジュール 活用テクニック

そこでテキスト・エディタでファイルを眺めると、ファ イル.projectでプロジェクト名を指定している 「<projectDescription> <name>LCD_Test </name>」という記述があります.これをLCD_Test2に 書き換えたところ、無事にインポートできました.この状 態でビルドは行えます.

次に Run → Externel Tools → Externel Tools Configuration としたところ, Externel Tools Configurations ウィン ドウには C17 Debbuger Launch for LCD_Test しか出てき ません. 今回コピーした LCD_Test2 がありません. 同じ ようにディレクトリを眺めて, 拡張子.launchのファイル C17 Debbuger Launch for LCD_Test.launch を C17 Debbuger Launch for LCD_Test2.launch にリ ネームすると Configurations ウィンドウに出てきました.

さらにこのファイルの中にワークスペースの場所を指定 する workspace_loc:/LCD_Test という部分があった ので,これも LCD_Test2 に変更します.この作業をしな いとデバッグ時にコピー前のファイルが参照されてハマり ます.

以上の変更で,無事新しいプロジェクトLCD_Test2の ビルド&実行ができました.

ここまでの作業をまとめると, Eclipse プロジェクトの コピーは, 以下の手順で行います.

1) .project の編集

 c main.c	×			
100	clearLine	e(3);		
101				
102	// LCD or	1		
103	lcdOn();			
104				
105	// draw checker			
106	drawLine(O);			
107	drawLine	(1);		
108	drawLine	(2);	ちポイ	ントの害力
(109	drawLine	(3);	////.	
ė110	for(i = 0); i < LOOF	_NUM;	i++);
Toggl <u>D</u> isab Break	e <u>B</u> reakpoint - { le Breakpoint point <u>P</u> roperties.	ブレークポー		の設定/解除)
<u>G</u> o to	Annotation	Otrl+1	JOH;	1++);
Add B Add]	Boo <u>k</u> mark [ask		arom	e.
 Show Show Foldir 	Quick Diff Line <u>N</u> umbers Ng	Ctrl+Shift+Q	·upt	i++);
Pre <u>f</u> ei	rences			
123	for(i = 0); i < LOOF	NUM;	i++);
124				

図1 ブレーク・ポイントの設定

- 2) <プロジェクト名>.launchを<新プロジェクト
 名>.launchにリネーム
- 3) <新プロジェクト名>.launchのworkspace_loc:/ を編集

3. 文字を書いてみる

● main.c を改変する

LCD_Test に含まれる main.cがメイン・ルーチンで す.initLcdPower()やinitLcd(),lcdOn()などの 関数を呼んで初期化を行っています.初期化が終わった後 に drawLine()などで描画を行っているようです.ソー ス・ウィンドウに表示されている main.cの行番号の左で 右クリックすると,メニューの中に「Toggle Break Point」 という選択肢があります.これを選択すると青色の丸が付 き,ここでプログラムを止められます(図1).実際にプロ グラムがどこまで実行されたか確認するには,このように ブレークポイントを設定して試していくのが一番です.

ソースを見ると, for ループで時間待ちをしているよう です.reverseLcdBW();の後の for ループあたりにプ ログラムを追加してみましょう.

● VRAM の構造

S1C17702の液晶画面は表示用ビデオRAM(以下 VRAM)を持っており、ここに値を書き込めば描画が行え ます.VRAMの構造は参考文献(1)の「22.5 表示メモリ」 の「図22.5.1 表示メモリマップ」に記載されています.これ を簡略化したものを図2に示します.なお、今回の基板で は1/32 デューティを選択しています.

縦8ドット×4行=32ドットで,横は72バイトです. これによりカタログ・スペックどおり72ドット×32バイ トになります.



図2 VRAMの構造



縦方向は図3のようになっています. 一番上のドットだ け黒くしたい場合には16進数で0x01を、一番下のドット だけ黒くしたい場合には0x80を書き込みます.すべて黒 くしたい場合は0xffです.たとえば、文字「A」の形で左上 に表示したい場合は図4のようなデータになります.これ をアドレス 0x80000 から順に書き込むと写真1のように表 示されます. このプログラムをリスト2に示します. リス ト1で示した「プログラムを追加する個所」にリスト2を挿 入すると、写真1のような表示が行われます.

応用例と注意

アドレスと書き込むデータがわかれば、後は好きな絵を 描くだけです.

一つだけ注意する点は、右端のアドレスと次の行の左端

<pre>* Main function. * * Main function. * * return: int Normality 0, Abnormality 1 * history: 2007/08/17 start */ int main(void) { long i; // loop counter long stimmer; psTimmer = &onesec</pre>	/*			
<pre>* * * * return: int Normality 0, Abormality 1 * history: 2007/08/17 start */ // int main(void) { long i; // loop counter long stimmer; psTimmer = &onesec</pre>	* Main function.	// initializes LCD interrupt		
<pre>* return: int Normality 0, Abnormality 1 * history: 2007/08/17 start */ int main(void) (long i; // loop counter long sTimmer; psTimmer = &onesec debugModeMisc(MISC_OIDBG_STOP); //misc stop in debug mode controlPsc(PSC_PRUN_RUN); //prescaler run debugModeMisc(MISC_OIDBG_STOP); //prescaler stop in debug mode controll(IGL_G_PCLKEN_ENA); //run clg controll(IGL_G_PCLKEN_ENA); //run clg controllt(ITC_CTL_ENA); //itc enable // initializes LCD // initializes LCD // clock is oSC3 startLcdClk(LCD_DLW_SELECT_OSC3, LCD_CLK_DIV_32); // clock is oSC3 startLcdClk(LCD_DLW_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdSBg(); cometation =</pre>	*	<pre>initLcdInt(0x1); // interrupt level is 1</pre>		
<pre>* history: 2007/08/17 start */ int main(void) { long i; // loop counter long sTimmer; psTimmer = &onesec debugModeMisc(MISC_OIDBG_STOP); //misc stop in debug mode controlPsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUN_STOP); //misc stop in debug mode setClockGear(CLG_CCLKGR_1); //set clock gear *defalt=0x0 controllClq(CLG_PCLKER_ENA); //run clg controllClq(CLG_PCLKER_ENA); //run clg controlltc(TTC_CTL_ENA); //it c enable // initializes LCD // prever booster of f, regulator on, heaby protection of f. initLcdPower (LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is oSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdSeg(); cmax</pre>	* return: int Normality 0, Abnormality 1			
<pre>*/ int main(void) { clearLine(0); clearLine(1); drawLine(1); drawLine(1); drawLine(1); drawLine(2); drawLine(3); // reverse LCD monochrome. reverseLdSW(1); for(i = 0; i < LOOP_NUM; i++); // initLed(LCD_DUPTy_32, LCD_CONTRAST_LV_14); reverseLcdSW(1); for(i = 0; i < LOOP_NUM; i++); (combatelyLine(Added); combatelyLine(Added); combatelyLine(Added); combatelyLine(Addeddddddddddddddddddddddddddddddddd</pre>	* history: 2007/08/17 start	// clear LCD		
<pre>int main(void) { long i; // loop counter long sTimmer; psTimmer = &onesec debugModeMisc(MISC_OIDBC_STOP); //misc stop in debug mode controlPsc(PSC_PRUND_STOP); //prescaler stop in debug mode controlCg(CLG_CCLKER_1); //prescaler stop in debug mode setClockGar(CLG_CCLKER_1); //prescaler stop in debug mode controlClg(CLG_CCLKER_ENA); //run clg controlClg(CLG_CCLKER_ENA); //run clg controlClg(CLG_CCLKER_ENA); //run clg controlClg(CLG_PCLKEN_ENA); //run clg controlClg(CLG_CCLKER_ENA); //run clg controlClg(CLG_FCKER_ENA); //run</pre>	*/	clearLine(0);		
<pre>long i; // loop counter long sTimmer; psTimmer; psTimmer = &onesec debugModeMisc(MISC_OIDEG_STOP); //misc stop in debug mode controlPsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUN_SUN); //prescaler stop in debug mode setClockGear(CLG_CCLKGR_1); //set clock gear *defalt=0x0 controlClg(CLG_PCLKEN_ENA); //run clg controlLtc(ITC_CTL_ENA); //run clg controlLtc(ITC_CTL_ENA); //itc enable // initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdEW(); reverseLcdEW(); reverseLcdEW(); combactylcmatocondectore reverseLcdEW(); combactylcmatocondectore reverseLcdEW(); combactylcmatocondectore reverseLcdEW(); combactylcmatocondectore reverseLcdEW(); combactylcmatocondectore reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // clock is OSC3 startLcd(lk(LDD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // clock is OSC3 startLcd(lk(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // re</pre>	int main(void) {	<pre>clearLine(1);</pre>		
<pre>long sTimmer; psTimmer ; psTimmer = &onesec debugModeMisc(MISC_OIDEG_STOP); //misc stop in debug mode controlPsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUN_RUN); //prescaler stop in debug mode setClockGear(CLG_CCLKGR_1); //tet clock gear *defalt=0x0 controlLtg(CLG_PCLKNE_NN); //ite enable // initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CUTX_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdSeg(); // cometable comet</pre>	long i; // loop counter	clearLine(2);		
<pre>psTimmer = &onesec // LCD on lcdon(); debugModeMisc(MISC_OIDBG_STOP); //misc stop in debug mode controlPsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUN_STOP); //prescaler run debugModePsc(PSC_PRUN_STOP); //prescaler run debugModePsc(PSC_PRUN_STOP); //prescaler run debugModePsc(DSC_PRUN_STOP); //prescaler stop in debug mode setClockGear(CLG_CLKER_1); //set clock gear *defalt=0x0 controllfc(ITC_CTL_ENA); //run clg controllfc(ITC_CTL_ENA); //run clg controlltc(ITC_CTL_ENA); //run</pre>	long sTimmer;	clearLine(3);		
psTimmer = &onesec debugModeMisc(MISC_OIDEG_STOP); //misc stop in debug mode controlPsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUN_STOP); //prescaler stop in debug mode setClockGear(CLG_CCLKRR_1); //set clock gear *defalt=0x0 controlLd(CLG_PCLKEN_ENA); //run clg controlItc(ITC_CTL_ENA); //ruc nable // initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdSm(); reverseLCdSm(); reverseLcdSm(); reverseLcdSm(); reverseLcdS				
<pre>debugModeMisc(MISC_O1DBG_STOP); //misc stop in debug mode controlPsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUND_STOP); //prescaler stop in debug mode setClockGear(CLG_CCLKGR_1); //set clock gear *defalt=0x0 controlItc(ITC_CTL_ENA); //run clg controlItc(ITC_CTL_ENA); //run clg controlItcLCD_CTL_ENA); //itc enable // initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CUK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd[LCD_DUTT_332, LCD_CONTRAST_LV_14]; reverseLcdSm(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdSm(); rev</pre>	psTimmer = &onesec	// LCD on		
debugModeMisc(MISC_01DEG_STOP); //misc stop in debug mode controlPsc(PSC_PRUND_RUN); //prescaler run debugModePsc(PSC_PRUND_STOP); //prescaler stop in debug mode setClockGear(CLG_CCLKGR_1); //set clock gear *defalt=0x0 controllg(CLG_PCLKEN_ENA); //run clg controlltc(ITC_CTL_ENA); //itc enable // initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is 0SC3 startLcdClk(LCD_CLK_SELECT_0SC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdSeg(); // clock is 0; // clock		lcdOn();		
controlPsc(PSC_PRUN_RUN); //prescaler run debugModePsc(PSC_PRUND_STOP); //prescaler stop in debug mode setClocKGear(CLG_CCLKGR_1); //set clock gear *defalt=0x0 controllg(CLG_PCLKEN_ENA); //run clg controlltc(ITC_CTL_ENA); //itc enable // initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdSeg(); // everseLcdSeg(); // clock is OSC3 // reverseLcdSeg(); // clock is OSC3 // clock is OSC3 // reverseLcdSeg(); // clock is OSC3 // clock is	debugModeMisc(MISC_01DBG_STOP); //misc stop in debug mode			
debugModePsc(PSC_PRUND_STOP); //prescaler stop in debug mode setClockGear(CLG_CCLKGR_1); //set clock gear *defalt=0x0 controlLg(CLG_PCLKEN_ENA); //run clg controlTtc(ITC_CTL_ENA); //reverse LCD monochrome. reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); // reverse LCD monochrome. reverseLcdEW(); for(i = 0; i < LOOP_NUM; i++); initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdEq(); reverseLcdEq(); /~ enverseLcdEq(); /~ envers	controlPsc(PSC_PRUN_RUN); //prescaler run	// draw checker		
setClockGear(CLG_CCLKGR_1); //set clock gear *defalt=0x0 controllg(CLG_PCLKEN_ENA); //run clg controlltc(ITC_CTL_ENA); //run clg controlltc(ITC_CTL_ENA); //itc enable // initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdSeg(); // set clock gear *defalt=0x0 drawLine(1); drawLine(1); drawLine(3); for(i = 0; i < LOOP_NUM; i++); // reverse LCD monochrome. reverseLcdBW(); for(i = 0; i < LOOP_NUM; i++); // reverseLCD monochrome. reverseLcdBW(); for(i = 0; i < LOOP_NUM; i++); // const.tylcmideseindesei	<pre>debugModePsc(PSC_PRUND_STOP); //prescaler stop in debug mode</pre>	drawLine(0);		
controllg(CLG_PCLKEN_ENA); //run clg controlltc(ITC_CTL_ENA); //run clg controlltc(ITC_CTL_ENA); //itc enable // initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdCom(); reverseLcdSeg(); ~~ ^略~	<pre>setClockGear(CLG_CCLKGR_1); //set clock gear *defalt=0x0</pre>	drawLine(1);		
controlltc(ITC_CTL_ENA); //itc enable drawLine(3); for(i = 0; i < LOOP_NUM; i++); // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdSeg(); // clock is OSC3 cele-	controlClg(CLG_PCLKEN_ENA); //run clg	drawLine(2);		
<pre>// initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLcdEw(); for(i = 0; i < LOOP_NUM; i++); // reverseLc</pre>	controlItc(ITC_CTL_ENA); //itc enable	drawLine(3);		
<pre>// initializes LCD // power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // reverse LCD monochrome. reverseLcdBW(); // reverse LCD monochrome. reverseLcdBW(); for(i = 0; i < LOOP_NUM; i++); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdSeg(); ~~ Note that the matching of the matching</pre>		for $(i = 0; i < LOOP_NUM; i++);$		
<pre>// power booster off, regulator on, heaby protection off. initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdSeg(); // reverse LCD monochrome. reverseLcdSeg(); // reverse LCD monochrome. reverseLcdBW(); for(i = 0; i < LOOP_NUM; i++); </pre>	// initializes LCD			
<pre>initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF); // clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdSeg(); ~~略~</pre>	<pre>// power booster off, regulator on, heaby protection off.</pre>	// reverse LCD monochrome.		
<pre>// clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdSeg();</pre>	initLcdPower(LCD_PWR_BOOSTER_OFF, LCD_HEAVY_PROTECT_OFF);	reverseLcdBW();		
<pre>// clock is OSC3 startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdSeg();</pre>		for $(i = 0; i < LOOP_NUM; i++);$		
startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32); // reverse LCD monochrome. reverseLcdBW(); // duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdSeg(); ~略~	// clock is OSC3			
<pre>// duty is 1/32, contrast is LV.14. initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdSeg();</pre>	<pre>startLcdClk(LCD_CLK_SELECT_OSC3, LCD_CLK_DIV_32);</pre>	// reverse LCD monochrome.		
<pre>// duty is 1/32, contrast is LV.14. initLcd(LcD_DUTY_32, LcD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdSeg();</pre>		reverseLcdBW();		
initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14); reverseLcdCom(); reverseLcdSeg(); ~略~	// duty is 1/32, contrast is LV.14.	<pre>for(i = 0; i < LOOP_NUM; i++);</pre>		
reverseLcdCom(); reverseLcdSeg(); ~略~	<pre>initLcd(LCD_DUTY_32, LCD_CONTRAST_LV_14);</pre>	→ このあたりに新しい描画を追加する		
reverseLcdSeg(); ~ ~略~	reverseLcdCom();			
	reverseLcdSeg();			

がつながっていないことです.たとえば、1行目の右端は 0x80047ですが、次の行の左端は0x80100であることに注 意してください.

0から71までの値を書き込むプログラムをリスト3に示 します.この実行結果は写真2のようになります.



● アドレスの計算

▶ 写真1 文字[A]を表示

画面へのパターン表示ができたので,次は任意の座標に 点を打つ関数を作成しましょう.

int pset(x,y); でx=0~71, y=0~31の範囲で 点を打ちます. 点を打つといっても, 結局はアドレスを計 算してそこにデータを書き込むだけです.

アドレスの計算式ですが、座標(0,0)のアドレスが 0x80000で、xが1増えるごとにアドレスも1増えます. 面

活用テクニック リスト2 文字「A」を書き込む unsigned char *p; clearLine(0); //画面消去 ~略~ clearLine(1); clearLine(2); clearLine(0); clearLine(3); A REPORT OF REPORT OF REPORT clearLine(1): clearLine(2); No to the test to the test to //文字 A のビット・パターン clearLine(3); *(unsigned char *)0x80000 = 0x7e; PERENEAR *(unsigned char *)0x80001 = 0x09; p = (unsigned char *)0x80000; *(unsigned char *)0x80002 = 0x09; for (i = 0; i < 72; i++) { *(unsigned char *)0x80003 = 0x09; *p++ = i;*(unsigned char *)0x80004 = 0x7e; p = (unsigned char *)0x80100;for(i = 0; i < LOOP_NUM; i++);</pre> 写真2 リスト3の実行結果 for (i = 0; i < 72; i++) { *p++ = 256-i; 3 p = (unsigned char *)0x80200;for (i = 0; i < 72; i++) { int pset(int x, int y) { *p++ = i;const unsigned char *vram=0x80000; 3 unsigned char *p; p = (unsigned char *)0x80300:unsigned char data=1; for (i = 0; i < 72; i++) { *p++ = 256-i: //範囲チェック 3 if ((x>72) || (y>32)){ ▶ リスト3 return (-1); //異常終了 0から71までの値を書き込む for(i = 0; i < LOOP_NUM; i++);</pre> プログラム p = vram+(y/8)*0x100+x; //アドレスの計算 data <<=(y % 8);</pre> // 書き込むデータ 倒なのが y の計算で, y が8 増えるごとに 0x100 増えます. *p = data;

そこで計算式は、ポインタ変数を(unsigned char *)p として、

p = vram + (y/8) * 0x100 + x;

になります.ここに書き込むデータは、y座標=0,8,16,24のときは0x01,y座標=1,9,17,25のときは0x02となります(図5).

ここではシフト演算子を使って,

unsigned char data=1;

data << =(y % 8); // 書き込むデータ
となります.ということで関数はリスト4のようになります.</pre>

入力値*x*と*y*の範囲チェックもちゃんと行ってます.こ れは意外と重要です.リスト5のようにこれを呼び出すと 写真3のように表示されます.

これで完成…と思ったのですが、実はバグがあります.



pset(0,0);

pset(0,2);

▶ リスト4

pset()関数

のように、同じアドレスへデータを書き込むと、前のドットが消えてしまいます(写真4).最初に書き込んだデータ 0x01が0x04で上書きされ、消えてしまうためです.これ を回避するためには、データの論理和(OR)をとって書き 込みます.この場合、最終的に書き込みたいデータは0x05 です.そこで、前に書き込んだデータとのORをとるよう に変更します.リスト6が変更後です.

return (0); //正常終了

まずは自由自在に画面表示を行おう 液晶搭載マイコン・モジュール

リスト5 pset()関数の呼び出し }





写真3 pset()関数の表示結果



「Interface」液晶搭載マイコン基板アプリケーション制作コンテストのお知らせ

Interface 編集部では、書籍『すぐに使える!液晶搭マイコン・ モジュール』に付属したマイコン基板を利用する「アプリケーショ ン制作コンテスト | を開催する予定です.

本基板には,液晶はもちろんのこと,スイッチや GPIO,タイ マなどが搭載されています.これらを活用した制御アプリケー ションや表示装置,ゲームなどさまざまな応用ができるでしょう.

この機会に実用的な作品,楽しい作品,ユニークな作品,教 育的な作品など,自慢の製作事例を持ち込んで,本コンテスト に参加してみませんか?

入賞者には、すてきな商品を差し上げる予定です.また、入 賞作品については、その製作レポートを本誌に掲載します.ア プリケーション制作コンテストの詳細は、次号(2010年4月号) の誌面および本誌 Webページ(図A)で発表します.

リスト6 pset()関数(OR版)

in	<pre>at pset(int x, int y){ const unsigned char *vram=0x80000; unsigned char *p; unsigned char data=1;</pre>
	//範囲チェック if ((x>72) (y>32)){ return (-1); //異常終了 }
	p = vram+(y/8)*0x100+x; //アドレスの計算 data <<=(y % 8); // 書き込むデータ *p = data; ← 変更した
}	return (0); //正常終了

これで**写真5**のように正常に表示されるようになりました.このように VRAM へ直接書き込むプログラムでは,前に描いたものを消さないために「重ね合わせ」の問題が発生します.今回はモノクロ液晶なので単純に OR で重ね合わせができましたが,これがフル・カラーになると,1ドットずつ確認していかないといけないので大変です.そのため,重ね合わせ機能をハードウェアで実装したグラフィックス・チップが世の中にたくさんあります.

* * *

ここまで理解できれば、もう好きな画像を描けるはずで す. Let's Try !

最近のパソコンは VRAM へ直接アクセスすることがで きず,どのような機構で画面に絵が描かれているのかを理 解するのは難しいと思います.しかし今回の液晶搭載マイ



図A 「うくに使える!液晶拾載マイコン・モンユール』 サホー ト・ページ http://www.cqpub.co.jp/interface/contents/speci al0912/



写真4 pset() 関数のテスト(OR なし版)



写真5 pset() 関数のテスト (OR あり版)

コン・モジュールのような素朴なシステムであれば,メモ リへのアクセスが画面への描画と直結していることが体で 理解できると思います.このような環境は貴重なので,ぜ ひ,自分の手を動かして試してみてください.

参考文献

(1) S1C17702 テクニカルマニュアル, セイコーエプソン.