KLCD_RL78 ： RL78 マイコン用ドライバ・キット　説明書

（1）概要

・漢字表示，キー入力（ローマ字かな変換含む）などのドライバ・キットです。
・printf 関数でグラフィック LCD に漢字表示ができます。
　秋月電子通商扱い 128x64 ドット LCD モジュール（SG12864A）などに好適です。
・gets 関数でキー入力を取得できます。かな変換や自動エコー表示に対応しています。
・比較的簡単なハードウエアで漢字表示，かな入力ができる回路図を添付しています。
・メロディ演奏，LED 調光，温度測定，時計，UART 通信，赤外通信も対応しています。
・ルネサス製マイコン RL78/G13 用（ツールは CS＋ for CA 用）です。
・マイコン，C 言語ともに初心者の人向けに解説書を同梱しています。
・ドライバ・ライブラリは購入前に試用できます。購入後はソース閲覧できます。

－－－　目次　－－－

（2）インストール／アンインストール

プログラム開発用の適当なフォルダに解凍して下さい。マイコンのポート割り付けを変更する場合や，RL78/G13(64～128pin)とは異なる RL78 を使用する場合は，ドライバのソース・プログラム（source.zip）も解凍します。これにはライセンス・コードが必要です。

アンインストールする場合は，解凍したファイルを削除してください。

（3）ファイル／フォルダ構成　（ □ がフォルダ名）

KLCD_RL78.pdf	本書
KLCD_RL78.mtpj	開発環境（CS＋ for CA）起動用のファイル
KLCD_RL78LIB.mtpj	ライブラリ作成環境（CS＋ for CA）起動用のファイル
*.dr	リンク・ディレクティブ
source.zip	ドライバのソース・プログラム（ライセンス・コード必要）

driver	
├ KLCD_RL78LIB.lib	ドライバ・ライブラリ
├ *.h	ドライバ・ライブラリを使用するためのヘッダ
└ *.c	必須指定ソース・プログラム（Font.c，HDWINIT.c，等）
main	解説書掲載のサンプル・プログラム（1～3 章用）
doc	
├ 解説書	「C から体感する RL78 マイコン」（1～3 章）
└ 回路図	回路図（PDF，D2CAD データ）

（４）ドライバの種類

①LCD 表示	128x64 ドット LCD に半角英数字や全角漢字を表示します。
②フォント	半角 95 文字，全角 1280 文字を実装しています。
③キー入力	5x5 マトリクスキー検出用です。
④キーコード変換	シフト状態に応じて ASCII／シフト JIS コードに変換します。
⑤メロディ制御	ブザー制御用です。曲の自動演奏にも対応しています。
⑥LED 調光	明るさ制御用です。PWM 波形を出力します。
⑦温度測定	A/D コンバータの読み取りと温度への変換計算をします。
⑧時計	リアルタイム・カウンタの設定／読み出し用です。
⑨UART（COM）通信	パソコンに接続して汎用ターミナル・ソフトと通信できます。
⑩赤外線通信	低速の赤外線データ転送用です（Ir-DA ではありません)。

（５）使用環境，変更方法

①開発環境（CS+ for CA）が必要です。ダウンロード方法，インストール方法が分らない場合は同梱の解説書（1 章）を参照してください。

②同梱の起動用ファイル（*. mtpj）をクリックして CS+ for CA を起動

③ポート，タイマ・チャネル，シリアル・チャネルの割り付けを変更する場合

driver¥PORT.h に割り付け定義があるので，これを変更してください。

driver¥HDWINIT.c にポート初期設定があるので，これも変更してください。

ドライバの再コンパイルも必要なので，変更該当のドライバ・ソース・ソースプログラムもプロジェクトのソースとして指定して下さい（ライブラリ指定があってもソース指定が優先されます)。

また，割り込みソースを変更した場合は，PORT.h だけではなく，その割り込みを使用しているソースの#pragma interrupt 記述も変更してください。

④本キットのサポート外の RL78 に変更する場合

ポートや内蔵周辺機能が異なる場合は③と同様に変更してください。

・ポート変更

同じピン数の品種でもポートの種類が異なったり，同じポート名であっても機能や特性（特に TTL 入力対応）が異なることがあるので，十分注意してください。

・クロック系

発振方法や周波数帯（10MHz 以下／10MHz 超）が異なる場合は， HDWINIT.c のクロック系設定を変更します。クロック周波数を変更する場合は HDWINIT.h の周波数も変更してください。

⑤フォントの差し替え方法

・半角フォントは，Font.c の中にデータで埋め込まれています。

・全角フォントは，driver¥Font.inc にデータがあります。実装する文字の種類だけ変更するにはフォント・データ生成ユーティリティ（Font_78K0R）を使用してください。

⑥スタートアップ・ルーチンの差し替え

独自のスタートアップ・ルーチン（driver¥RESET.asm）を使用しています。

標準のスタートアップ・ルーチンを使用する場合は，main 関数の先頭でドライバ初期化関数 drvinit()を呼び出してください。

（6）ドライバ関数

具体的な使用例は，同梱の解説書を参考にしてください。

①LCD ドライバ（LCD.h)

・void LCD_open(void)；　　LCD の初期化（静的変数初期化前でも呼び出し可能）
標準出力ストリーム stdout(63 バイト)を定義しています。

・int LCD_display(FILE *fp)；　LCD 表示（出力ストリームから 1 バイトを取り出して表示。全角は 2 回目の呼び出しで表示)
引数　：FILE *fp;　　出力ストリームへのポインタ。
戻り値: int;　　出力文字。ただしストリーム空時は EOF。

②フォント・ドライバ（Font.h)　（注：driver¥Font.c をソース指定してください)

・__far char *get_font_ascii(unsigned char c)；　　半角英数フォントの取得
引数　：unsigned char c;　1 バイト文字コード
戻り値: __far char*;　　フォント・データ(14 バイト)へのポインタ
エラー時はダミーフォントへのポインタ

・_far char *get_font_kanji(unsigned int c)；　　全角フォントの取得
引数　：unsigned int c;　2 バイト文字コード(上位が最初のコード)
戻り値: __far char*;　　フォント・データ(28 バイト)へのポインタ
エラー時はダミーフォントへのポインタ

③キー入力，コード変換ドライバ（KEY.h，FEP.h)

・int KEY_get(void)；　　キー入力ドライバ（0.01 秒周期で呼び出しのこと）
戻り値：int;　　キー番号(1～25)。ただしキーが離された時 0，変化が無いときは EOF。

・int KEY_input(void)；　　最新のキー番号を取得。
戻り値：int;　　キー番号(0～25，0 はキーオフ)。EOF は取得済。

・void KEY_code(int d);　　キー番号を ASCII コードまたはシフト JIS に変換。
引数　: int d;　　キー番号
変換結果を，キーエコー用ストリーム stdecho（15 バイト）および編集バッファ（31 バイト）に格納します。編集バッファが満杯の場合は無視します。
さらに引数がキー番号 1（ENT キー，ASCII コード'¥n'に対応）であれば，編集バッファの内容を標準入力ストリーム stdin（63 バイト）へ移動します。
キー番号と文字の対応は，解説書 1.4.2 項を参照してください。

④入出力関数（stdio.h）

C 言語標準ライブラリ関数と，追加の関数を宣言しています。下記の関数はツール添付の標準ライブラリ関数を差し替えるため，必ずソースとして指定して下さい（デフォルトのプロジェクト・ファイルでは指定済みです)。

・int putchar(int c);　　標準出力ストリームへの 1 バイト出力。
引数　: int c;　　出力文字。
戻り値: int;　　出力文字。注意:デフォルトでは，ストリーム満杯時は待ち状態になります。main 関数およびそこから呼ばれる関数内で使用するときは問題ありませんが，タスク関数（ドライバなど）から呼ぶときは待ち状態にならないように，あらかじめ後述の FILE_space 関数で，ストリームに空きがあることを確認して下さい。putchar 関数は printf 関数からも呼ばれるため，printf 関数をタスク関数内で使用するときは，出力文字数分の空きがあることを確認してから呼んでください。

・int getchar(void);　　標準入力ストリームから 1 バイト入力。
戻り値: int;　　入力文字。ただしストリーム空時は EOF。

・int putc(int c, FILE *fp);　指定ストリームへの 1 バイト出力。
引数 1 : int c;　　出力文字。
引数 2 : FILE *fp;　　出力ストリームへのポインタ。
戻り値: int;　　出力文字。ただしストリーム満杯時は EOF。

・int getc(FILE *fp);　　指定ストリームから 1 バイト入力。
引数　: FILE *fp;　　入力ストリームへのポインタ。
戻り値: int;　　入力文字。ただしストリーム空時は EOF。

・int puts(char *s);　　標準出力ストリームへの文字列出力。
　引数　 : char *s;　　出力文字列へのポインタまたは文字列リテラル。
　戻り値: int;　　成功時 0。失敗時 EOF（出力ストリームに引数 1 の文字列長分の空きが無い場合）

・int fputs(char *s, FILE *fp);　指定ストリームへの文字列出力。
　引数 1 : char *s;　　出力文字列へのポインタまたは文字列リテラル。
　引数 2 : FILE *fp;　　出力ストリームへのポインタ。
　戻り値: int;　　成功時 0。失敗時 EOF（出力ストリームに引数 1 の文字列長分の空きが無い場合）

・char *gets(char *s);　　標準入力ストリームから文字列入力。
　引数　: char *s;　　入力文字列を格納する配列へのポインタ。
　戻り値: char *;　　成功時は引数 1 の値。ストリームに残っているデータの中に改行（¥n）がなければ NULL。
　　注意: デフォルトではバックスペース(¥b)による編集を行った結果を返すようにしてあります。

・char *fgets(char *s, int n, FILE *fp);　指定ストリームから文字列入力。
　引数 1 : char *s;　　入力文字列を格納する配列へのポインタ。
　引数 2 : int n;　　入力文字列を格納する配列のサイズ。
　引数 3 : FILE *fp;　　入力ストリームへのポインタ。
　戻り値: char *;　　成功時は引数 1 の値。ストリームに残っているデータ数が n-1 バイト以下で，かつその中に改行（¥n）がなければ NULL。
　　注意: デフォルトではバックスペース(¥b)による編集を行った結果を返すようにしてあります。

・FILE *fopen (char *name, char *mode);　外部ファイル（物理装置含む）を開く。
　引数 1 : char *name;　　本版では，UART 通信の場合”COM:”，赤外線通信の場合”IR:”を指定。
　引数 2 : char *mode;　　本版では，送信側は”wb”，受信側は”rb”を指定。
　戻り値: FILE *;　　成功時はストリームへのポインタ。失敗時は NULL。

・int fclose (FILE* fp);　　外部ファイル（物理装置含む）を閉じる。
　引数　: FILE *fp;　　閉じようとするストリームへのポインタ。
　戻り値: int;　　成功時 0。失敗時は EOF。

注意:デフォルトの fopen/fclose を使うと COM,IR 両方のドライバが組み込まれます。
片方だけ組み込みたい場合は，それぞれの通信ドライバの open/close 関数を直接呼び出すか，driver_list.h から不要なドライバのインクルードを削除して fopen/fclose を再コンパイルして下さい。

⑤時間関係（time.h, RTC.h）

・void BTIMER_open(void); 定周期（0.01 秒）タイマの初期設定。呼び出すと，0.01 秒間隔で自動的に BTIMER_int 関数（各種ドライバ呼び出し関数）を起動します。

・clock_t clock(void); プロセッサ時間の取得（C 言語標準関数）
戻り値: clock_t; この値を定数マクロ CLOCKS_PER_SEC で割ると 1 秒単位の経過時間となります。割り込み禁止が解除されます。

・void wait_clock(float t); 短時間の待ち。
引数 : float t; マイクロ秒[μs]単位の値を指定します。この時間何もせずに待ちますが，割り込み処理は受付可能です。
リエントラント可能。

・time_t time(time_t *timer); 暦時刻の取得。
引数 : time_t *timer; 戻り値のコピー先ポインタ。通常は NULL を指定。
戻り値: time_t; 本版では 0 時,12 時からの秒数。失敗時は EOF。

・struct tm *localtime(time_t *t); 本版では，RTC から時刻を読み取って，要素別の時刻に変換。
引数 : time_t *t; 暦時刻へのポインタ。本版ではダミー。
戻り値: struct tm *; 要素別の時刻構造体へのポインタ。

・int settime(struct tm *t); 要素別の時刻の RTC への書き込み。
引数 : struct tm *t; 要素別の時刻構造体へのポインタ。
戻り値: int; 成功時は 0，失敗時は EOF。

・void RTC_open(int c); RTC 初期設定用マクロ。
引数 : int c; RTC 制御レジスタ RTCC0 への書込み値。

⑥ブザー制御，メロディ制御（BEEP.h）

・void BEEP_timer_start(unsigned int c);　　方形波出力の開始，再開
　引数　：unsigned int c;　　625kHz / 出力周波数 - 1 を設定。

・void BEEP_timer_pause(void);　　方形波出力の一時停止

・void BEEP_timer_stop(void);　　方形波出力の停止

・void BEEP_check(void);　　メロディ定周期処理。以下の演奏関係関数を使用する場合は 0.01 秒周期で呼び出しが必要です。

・int BEEP_play(ONPU *p, int n, int t, int i);　　メロディ演奏要求
　引数 1 ： ONPU *p;　　音符データを格納した配列へのポインタ。
　引数 2 ： int n;　　配列 p の演奏を開始する要素番号。
　引数 3 ： int t;　　演奏テンポ。1 分間の四分音符の個数で指定。
　引数 3 ： int i;　　音程変更。半音単位で±の数値を指定。
　戻り値: int;　　成功時は 0 以上，失敗時（演奏中）は EOF。

・int BEEP_stop(void);　　演奏停止要求。
　戻り値: int;　　演奏中の音符配列の位置（0 以上）。停止中は EOF。

・int BEEP_refer(void);　　演奏状態の取得。
　戻り値: int;　　演奏状態コード。詳細はヘッダ参照。

⑦LED 調光（PWM.h）

・int PWM_start(int k);　　LED 調光開始
　引数　：int k;　　輝度 0～100%を 0～10000 の数値で指定。
　戻り値: int;　　成功時 0，失敗時 EOF（タイマが他で使用されている）。

・int PWM_stop(void);　　LED 調光停止
　戻り値: int;　　成功時 0，失敗時 EOF(未使用時)。

⑧温度測定（AD.h）

・unsigned int AD_read(int c)； A/D コンバータ読み出し
引数 ： int c; A/D チャネル指定。
戻り値: unsigned int; 上位 10 ビットに読み出し値が格納されます。

・int TEMPERATURE(unsigned int x)； A/D コンバータ読み出し値の温度への変換
引数 ： unsigned int x； A/D コンバータ読み出し値。
戻り値: int; 温度。1℃単位。
注意：5V 電源で使用する場合は，コンパイル・オプションの定義に VDD5V を追加してください。デフォルトのプロジェクト・ファイルには指定してありません。

⑨UART（COM ポート）通信ドライバ（UART.h）

・void UART_TX_check(void)； 送信フロー制御。0.01 秒以下の周期で呼び出し必要。

・void UART_RX_check(void)； 受信フロー制御。0.01 秒以下の周期で呼び出し必要。

・FILE *UART_TX_open(void)； 送信 UART オープン。fopen 関数から呼ばれます。
戻り値: FILE *; 成功時はストリームへのポインタ。失敗時は NULL。

・FILE *UART_RX_open(void)； 受信 UART オープン。fopen 関数から呼ばれます。
戻り値: FILE *; 成功時はストリームへのポインタ。失敗時は NULL。

・int UART_TX_close(void)； 送信 UART クローズ。fclose 関数から呼ばれます。
戻り値: int; 成功時は 0，失敗時は EOF。送信未了のデータがストリームに残っている場合はクローズせずに，UART_CLOSE_RETRY を返します。

・int UART_RX_close(void)； 受信 UART クローズ。fclose 関数から呼ばれます。
戻り値: int; 成功時は 0，失敗時は EOF。

・int UART_set_baud(void)； ボーレート設定変更。オープン前に呼び出しが必要。
戻り値: int; 成功時は 0，失敗時は EOF。
注：ボーレートは UART.h の UART_SPEED 定義値に書く必要があります。

⑩赤外線通信ドライバ（IR.h）

・void IR_TX_check(void);　　送信シンボル組立。0.01 秒以下の周期で呼び出し必要。

・void IR_RX_check(void);　　受信データ組立。0.01 秒以下の周期で呼び出し必要。

・FILE *IR_TX_open(void);　　赤外線送信オープン。fopen 関数から呼ばれます。
戻り値: FILE *;　　成功時はストリームへのポインタ。失敗時は NULL。

・FILE *IR_RX_open(void);　　赤外線受信オープン。fopen 関数から呼ばれます。
戻り値: FILE *;　　成功時はストリームへのポインタ。失敗時は NULL。

・int IR_TX_close(void);　　赤外線送信クローズ。fclose 関数から呼ばれます。
戻り値: int;　　成功時は 0, 失敗時は EOF。送信未了のデータが
ストリームに残っている場合はクローズせずに,
UART_CLOSE_RETRY を返します。

・int IR_RX_close(void);　　赤外線受信クローズ。fclose 関数から呼ばれます。
戻り値: int;　　成功時は 0, 失敗時は EOF。

（７）仕様，制限事項，注意事項，等

①対象開発ツール

ルネサス エレクトロニクス製の統合環境 CS+ for CA。動作確認は，V4.0。

②ドライバ・キットの使用許諾条件（これに同意できない場合は使用許諾しません）

・シェアウエア料金支払い前に本キットを使用したプログラムを開示・頒布することを禁じます。

・シェアウエア料金支払い後は，下記 a～d の開示・頒布は制限しません。

ａ：実行形式ファイル（*.lmf，*.hex）の開示・頒布（商用／非商用を問わず）

ｂ：実行形式ファイルを書きこんだマイコンや機器の頒布（商用／非商用を問わず）

ｃ：オブジェクト・ライブラリ（*.lib）およびヘッダ（*.h）をオリジナル・プログラムに同梱しての開示・頒布（商用／非商用を問わず）。ただし出処を明記願います。

ｄ：非営利（無償で，かつ原稿料・印税・広告料などの間接収入も無し）において，driver フォルダに格納してあるソース･プログラム(Font.c,BTIMER.c,HDWINIT.c)あるいはサンプル・プログラム（main_*.c）について，一部または全部をオリジナル・プログラムに同梱しての開示・頒布。ただし出処を明記願います。

・シェアウエア料金支払いにかかわらずドライバ・ソース・プログラム（ライセンス・コードによって解凍されたファイル）は，一部および全部の開示・頒布を禁じます。

・本キットは無保証です。本キットの使用によるいかなる損害も補償しません。

③解説書の 4 章

作成は未定です。

④メモリ・カード，音声録音再生のドライバ

本キットに含む予定は今のところ未定です。

（８）重要な更新情報

R1.05	2017/8/15	HDWINIT.c	PMC レジスタ設定漏れ修正
		Font.c	２ページ目以上の漢字テーブルの位置計算修正
		LCD.c	不具合(戻り値の無い return)修正
		getc.c, putc.c	境界をまたぐ場合の計算不具合修正
		UART.c	受信エラー処理追加
R1.04	2012/8/24	公開（解説書３章まで）	
R1.01	2012/6/17	暫定公開（解説書２章まで）	

（９）シェアウエア料金，支払い方法

ベクターのシェアレジで支払い願います。購入するとライセンス・コードが送られてきますので，それをパスワードとして KLCD_RL78.zip を解凍してください。パスワード付き zip ファイルの解凍は Lhaplus などのフリー・ソフトで出来ます。

バージョンアップでも同一のライセンス・コードを使用できます。

（１０）サポート

①問い合わせ先：100-softsupportlltq@@memoad.jp　（注：@@を@に変えて下さい）

できるだけタイトル先頭に【サポート依頼】を付けて下さい。通常は３日以内に返答しますが，見落としや旅行・入院などで返答できない場合があります。

FAQ や追加情報がある場合は，下記サイトのソフトウエア・サポートのページに掲載します。

http://www2u.biglobe.ne.jp/~tequila/

②ビルド・エラー，動作不具合の問い合わせ

問い合わせ時にプロジェクト・ファイル（*.mtpj）を添付してください。

エラーなどのメッセージが出た場合は，コピー&ペーストして添付してください。

2017 年 8 月 15 日　てきーらサンドム