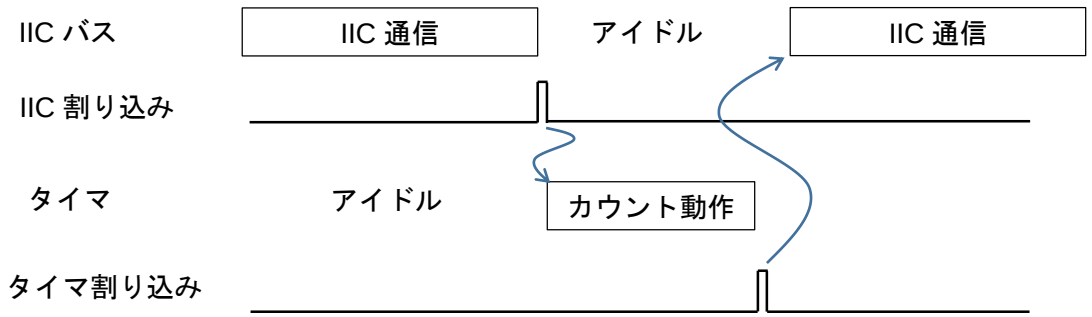
はじめに

I2C 通信の第 5 弾としては、RL78/G14 に内蔵された SAU の簡易 IIC 機能をポーリング方式で LCD モジュールを制御しています。（例によって、CA-78K0R からの移植です。）
なお、受信機能も作りこんでいますが、LCD モジュールは送信機能をサポートしていないと記載されていたので、マスタ受信機能のデバッグや動作確認は行っていません。

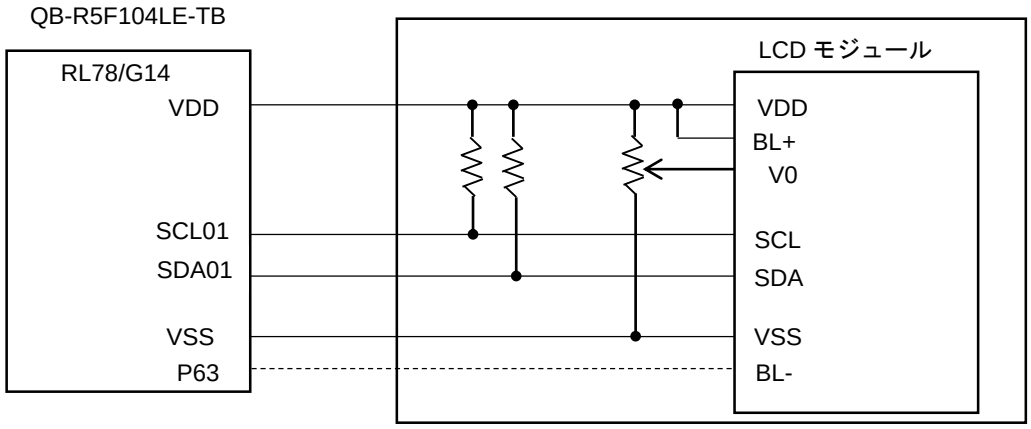
簡易 IIC では IICA0 を用いたマスタに比べて、機能が制限されています。スレーブとして使用できない以外に、スタートコンディションやストップコンディションの発行がハードウェアではなく、ソフトウェアで処理する必要がある点、スレーブと同期をとる機能（ウエイト機能）がサポートされていません。

ここで使用している制御対象の LCD モジュールでは、1 バイト分の通信完了から次の通信までにある程度の時間がないと正常に動作しません。
簡易 IIC での通信も同様に、スレーブ側での通信準備が完了するまでの時間をきちんとソフトウェアで確保する必要があります。IICA0 では、この部分はスレーブが SCL 信号をローレベルに引くことでマスタにウエイトをかけて対応できていました。しかし、簡易 IIC では、スレーブから（ハードウェアで）マスタにウエイトをかけることができないので、マスタ側がソフトウェアで対応する必要があります。つまり、1 バイト分の通信完了から次の通信開始までの時間待ち処理を追加する必要があります。時間を効率的にカウントするなら、TAU のワンカウント・モードを使用します。IIC 通信完了時にタイマをトリガして、待ち時間のカウントを開始します。IIC の制御ソフトウェアとしては通信を開始するときに、タイマがカウント完了していることを確認してから処理を開始するようにします。



ハードウェアの構成

基本的なハードウェア構成は変わりません。ここでは、SAU のチャネル 01 と TAU のチャネル 3 を使用して、I2C バスを制御します。




ソフトウェアの構成（IIC00 の制御ルーチン）

ここでは、IIC01 を割り込みベースで制御します。IICA0 と異なり、簡易 IIC では IIC バスの状態や通信状態を示すレジスタが存在しません。そのため、これらを示すためのフラグを準備する必要があります。このためのフラグとして、g_iic01_status を準備します。
このフラグの状態は以下のようになります。

| 設定値 | 状態 | 備考 |
|---|---|---|
| 0x00 | 正常終了状態 | 初期値 |
| 0x01 | マスタ送信状態 | |
| 0x81 | マスタ受信でアドレス送信中 | |
| 0x41 | マスタ受信中 | |
| 0xFF | 異常終了状態 | NACK 応答 |

通信が起動して、スタートコンディションを発行すると 0x01（送信）か 0x81（受信）になります。ただし、この段階では、IIC01 は送信モード（SCR01 の TXE01 は 1）になります。受信動作では、スレーブからのアドレス送信に ACK 応答があれば、SCR01 の TXE01 を 0、RXE01 を 1 にして受信モードにします。(そのようにプログラムしたはずですが、動作確認はしていません。)

プログラムとしては、汎用性も意識して作成していますが、使用している LCD モジュールに合わせた対応を追加しています。デバッグを進めながら、時間待ちをあちこちに入れているので、無駄があるかもしれませんが、目的の動作はしているので、OK としておきます。

IIC01 の基本的な動作の制御のために以下の関数で準備しています。

・R_IIC01_Master_Send()関数：送信を開始するための処理を行います。
・R_IIC01_Master_Receive()関数：受信を開始するための処理を行います。（未使用）
・R_IIC01_start_condition()関数：IIC01 で通信を開始するために、IIC バスにスタートコンディションを発行します。IIC01 は送信モードに設定されます。
・R_IIC01_StopCondition()関数：通信を終了する場合に I2C バスを開放するために、IIC バスにストップコンディションを発行します。
・R_IIC01_wait_comend()関数：IIC01 の通信完了を待ちます。
・r_iic01_interrupt()関数：IIC01 の通信完了割込み処理関数です。ここでは、TM03 を起動するだけです。
・r_inttm03_interrupt()関数：変数 g_iic01_status を使用して、実際の IIC01 の通信完了割込みの処理を行います。すべてのデータの通信処理が完了したら、結果を変数 g_iic01_status に設定して終了します。

ここでは、これらとは別に LCD モジュールを簡単に制御するために、コマンドやデータを設定する関数を準備しています。

・init_LCD()関数：タイミングを調整しながら、LCD モジュールを初期化します。
・set_command()関数：上記の関数群を使って、IIC01 を制御し、引数で渡されたバイト・データを LCD モジュールにコマンドとして送信します。
・set_data()関数：上記の関数群を使って、IIC01 を制御し、引数で渡されたバイト・データを LCD モジュールにデータとして送信します。
・wait_time()関数：引数で渡された回数 20μs 待ち関数を呼び出して、指定された時間を待ちます。
・wait_5us()関数：TM02 をワンカウント・モードで使用して 5μs の時間を待ちます。CA78K0R ではソフトタイマを使用していましたが、CC-RL では時間が大きく異なったために、TM02 を使った構成に変更しています。
・set_2digit()関数：引数で渡されたバイト・データを 2 ケタの ASCII コードに変化して LCD モジュールに送信（表示）します。（未使用）
・set_1digit()関数：引数で渡されたバイト・データの下 4 ビットを ASCII コードに変化して LCD モジュールに送信（表示）します。（未使用）

メイン処理は、12 ビットインターバルタイマを用いた 100ms 割込みをカウントして、500ms のタイミングを生成し、表示をスクロールします。
・r_it_interrupt()関数：100ms ごとの割込みをカウントして、500ms のタイミング（g_05s_status）を作成します。