UART1 のコード生成されたコードのシミュレータでの動作確認

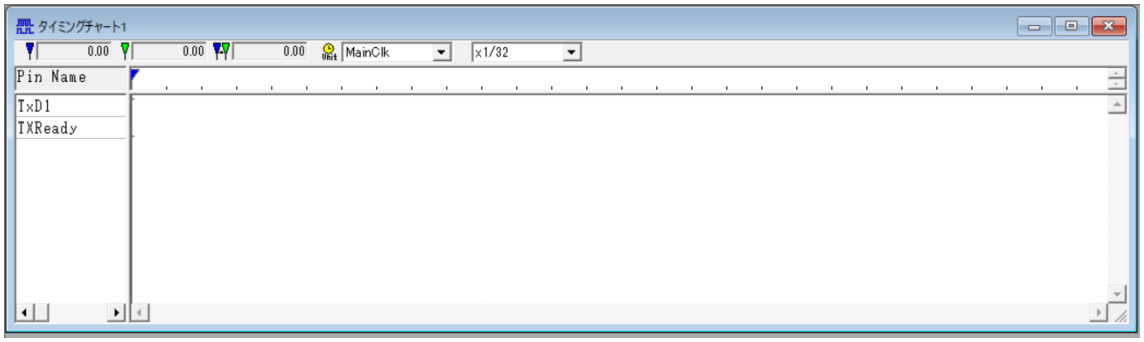
コード生成された通信用 API の問題点の一つが、通信完了の処理は callback 関数でユーザーが処理する必要があることです。普通は通信状態を示すフラグを準備しておくのですが、生成されたコードにはこのハンドリング処理が全くありません。普通であれば、通信を開始したら、フラグをクリアし、通信が完了（指定されたデータが全て送信完了）したら、フラグをセットすることになります。しかし、callback だけではフラグのセットしかできません。そこで、フラグのクリアは main 処理のレベルで実行することになります。

普通は、フラグは変数として、メモリ上に確保するのですが、今回のスレッドでは「8 バイト送信したいのに 1,2 バイトしか送信できない」となっていたので、メモリではなく、SFR（具体的に出力ポート）を使って、外部から確認できるようにしてみました。出力ポートの初期状態は 0 なので、それをそのまま使うことにします。ここでは、P30 を使います。20pin の RL78/G13 は持っていないので、ここではシミュレータのタイミングチャート機能を使って確認することにします。

測定に使用する main プログラムを以下に示します。

```
70                  void main(void)
71                  {
72      0019e           R_MAIN_UserInit();
73                      /* Start user code. Do not edit comment generated here */
74                      {
75
76                  //      uint8_t work;
77
78      001a1           P3_bit.no0 = 0x00;                  /* 送信完了フラグをクリア   */
79
80                      while( 1 )
81                      {
82
83      001b0               R_UART1_Send( (uint8_t *)g_tx_buff, 0x08 );     /* 送信処理開始  */
84      001a7               while( 0 == P3_bit.no0 )            /* 送信完了待ち             */
85                          {
86      001a6                   NOP();
87                          }
88
89
90      001ac               NOP();
91
92      001ad               P3_bit.no0 = 0x00;                  /* 送信完了フラグをクリア   */
93
94                      }
95                  }
96                  /* End user code. Do not edit comment generated here */
97              }
```

90 行目の NOP();はブレークポイントを設定するために準備したものです。下に GUI のタイミングチャート部分を示します。上が TxD1 信号で、下側の信号が P30 です。



r_cg_serial_user.c については、r_uart1_callback_sendend 関数の所に次に示すように、

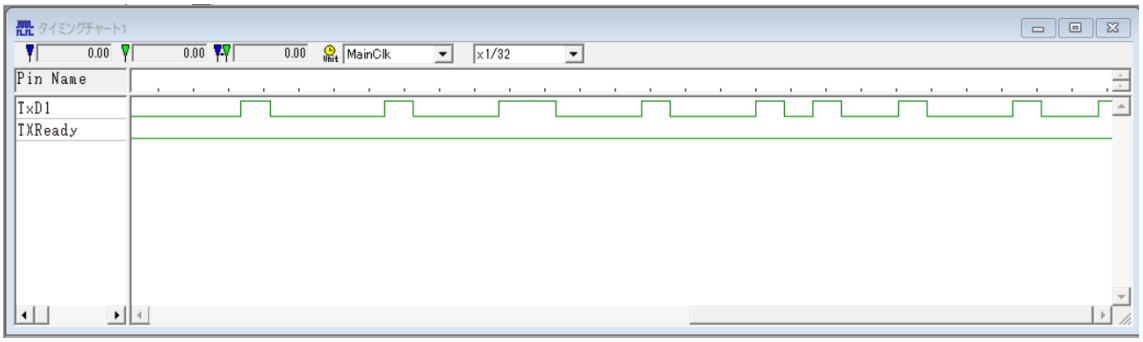
P3_bit.no0 = 1;を追加するだけです。

```
84 static void r_uart1_callback_sendend(void)↓
85 {↓
86     /* Start user code. Do not edit comment generated here */↓
87      P3_bit.no0 = 1;↓
88     /* End user code. Do not edit comment generated here */↓
89 }↓
```

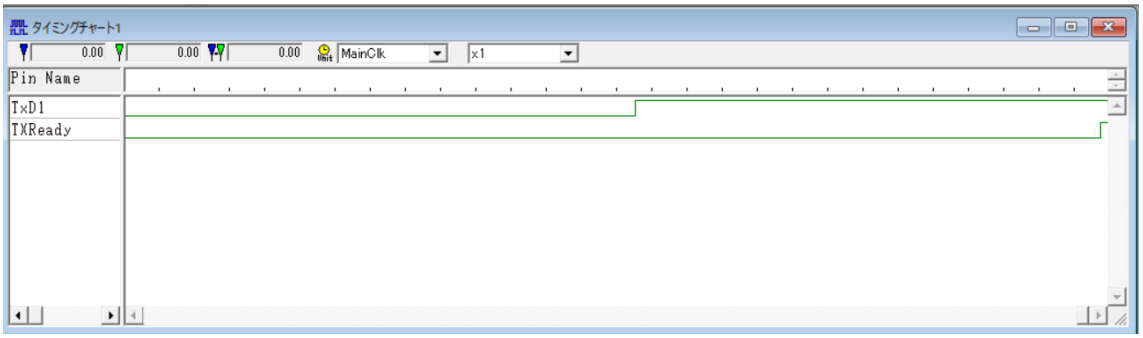
また、R_MAIN_UserInit 関数は R_UART1_Start()関数を呼び出して UART1 を起動しています。

```
105 void R_MAIN_UserInit(void)↓
106 {↓
107     /* Start user code. Do not edit comment generated here */↓
108     R_UART1_Start();              /* UART1動作起動              */↓
109     EI();↓
110     /* End user code. Do not edit comment generated here */↓
111 }↓
```

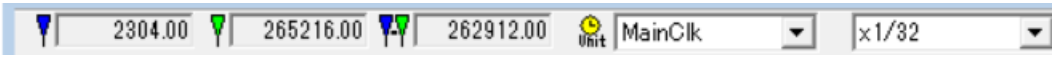
シミュレータにダウンロードして、ブレークポイントありで実行させると以下のようになります。



最後の所を拡大したのが下の図です。下側の信号が最後に立ち上がっているのが分かります。



最初のスタートビットから最後のストップビットまでの時間は以下のようになります。



このように、1,2 バイトでなく 8 バイト分が送信されていることが確認できます。