Technical Note No. 115

Master-Slave Networks

Summary

Library functions for master-slave 9th-bit binary communication using a two-wire, half-duplex RS-485 connection are available from Z-World. This protocol is only supported on Z180 port 1, which is configured for RS-485 communication for the BL1000, BL1100, BL1200, BL1400, BL1500, BL1600, PK2100, PK2200, PK2300, and PK2400. Any of these boards can be a master or slave. There should only be one master, with board address 0. Each slave should have its own distinct identification number, from 1 to 255.

Functional support for master-slave serial communication consists of the following:

- initialization of the Z180 port 1 for RS-485 communication
- the master sending an inquiry and waiting for a response from a slave
- the slaves monitoring for their own address during the 9th-bit transmission, and the targeted slave replying to the master

The binary command protocol adopted by Z-World is similar to that used for the Opto 22 binary protocol. A master message follows this structure:

```
[address of slave] [length] [ ] [ ]
  . . . . [ ] [crc hi][crc lo]
```

A slave response follows this structure:

```
[length] [ ] [ ] . . . . [ ]
 [crc hi] [crc lo]
```

The term **[length]** gives the number of bytes following.

During the transfer from the master, the **[address of slave]** byte is transferred with the 9th-bit set. Only the slave residing at this address will open up to listen to the rest of the message. The body of the message is sent with the 9th bit clear.

Hardware Connection

Typical connections for the two-wire RS-485 network are shown in Figure 1 below. Any of Z-World's controllers can be either a master or a slave. Each network can have only one master, but up to 255 slaves.

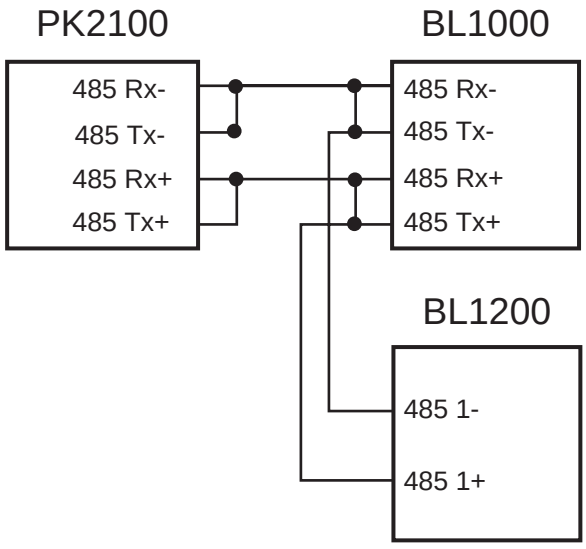

Figure 1. Typical RS-485 Network Connections

The BL1200 and PK2200 have half-duplex RS-485 ports. The BL1000, BL1100, and PK2100 have full-duplex RS-485 ports. To configure a full-duplex RS-485 port to half-duplex, connect RX+ to TX+ and RX- to TX-. Biasing and termination resistors should be removed from all slave units.

Software Support

- **`void op_init_z1( char baud, char* rbuf, char address )`**

  Initializes the Z180 port 1 for RS-485 9th-bit binary communication. The data format defaults to 8 bits, no parity, and 1 stop bit.

  PARAMETER 1: **`baud`** selects the baud rate in multiples of 1200 (for example, 16 specifies 19,200 baud).

  PARAMETER 2: **`rbuf`** is the pointer to the receive buffer.

  PARAMETER 3: **`address`** is the network address of the board: 0 for the master board, 1–255 for the slaves.

- **`int check_opto_command()`**

  Checks for a valid and completed command (or reply) in the receive buffer.

  RETURN VALUE: 0 if there is no completed command or message available; 1 if there is a completed command or message available; –2 if the completed command or reply has a bad cyclic redundancy check (CRC) value.

- **`int sendOp22( char dest, char* message, char len, int delays )`**

  This function is used by the master to send a message to the slave and then wait for a reply. This function will format the message according to the required protocol, computing its CRC value.

  PARAMETER 1: **`dest`** is the slave identity (1–255).

  PARAMETER 2: **`message`** is the pointer to the message to be sent to the slave.

  PARAMETER 3: **`len`** is the length of the message. Maximum message length is 251 bytes.

  PARAMETER 4: **`delay`** is the number of 50-millisecond delays to wait for the slave reply. If the real-time kernel is in use, then software executes the delay using the suspend function. Before each delay, the receive buffer is checked for a completed reply.

  RETURN VALUE: –1 if there is no reply from the slave; –2 if a completed reply has a bad CRC value; 1 if there is a completed reply with a proper CRC value.

  The reply is stored in **`rbuf`** after being received according to the required protocol.

- **`void replyOpto22( char* reply, char count, int delay )`**

  A slave uses this function to reply to the master's message. This function will format the reply according to the required protocol.

  PARAMETER 1: **`reply`** is the pointer to the reply. This could be a character string or it could be a binary data array or structure.

  PARAMETER 2: **`count`** is the length of the reply in bytes. Maximum reply length is 252 bytes.

  PARAMETER 3: **`delay`** is the number of 50-millisecond delays before the reply is sent. If the real-time kernel is in use, then software executes the delay using the **`suspend`** function.

Miscellaneous Functions

- **`void misticware( char* tbuf, char count )`**

  This function is a "gateway" for RS-485 9th-bit binary communication. Sets up the transmit buffer and initializes interrupt-driven transmission.

  PARAMETER 1: **`tbuf`** is the pointer to the message to be transmitted. This message should already be formatted, with its CRC, in the 9th-bit protocol.

  PARAMETER 2: **`count`** is the total number of bytes to be transmitted.

- **`void optodelay()`**

  Creates a 50-millisecond software delay. If the real-time kernel is in use, then software executes the delay using the **`suspend`** function.

- **`int rbuf_there()`**

  Monitors receive buffer for a completed command or reply.

  RETURN VALUE: 1 if a completed command or reply is available; 0 if no completed reply or command is available.

- **`void op_send_z1( char* tbuf, char count )`**

  This function is called by **`misticware`** to initiate transmission of data.

- **`void op_rec_z1()`**

  This function is called by **`misticware`** to reset and to ready the receiver for data reception.

- **`void op_kill_z1()`**

  Kill Z180 port 1. The RS-485 driver is also disabled.

- **`void z1_op_int()`**

  This is an interrupt service routine for the Z180 port 1 used in the master-slave networking.

Libraries

The following support libraries are available:

`network.lib`: Half-duplex RS-485 communication drivers

`modem232.lib`: Accessory library

Sample Programs

The following programs are available in **`samples\network`**:

`RS-485.c`: Simple slave RS-485 program. Talks back to master board running **`rs232.c`**.

`sremote.c`: Diagnostic Port via the RS-485 linkage. The master has to be running one of the following programs: **`z0rem.c`**, **`uartrem.c`**, **`cz0rem.c`**, **`cuartrem.c`**, **`s0rem.c`** , **`com1rem.c`**, **`or com2rem.c`**.

`csremote.c`: Same as **`sremote.c`**, but will only run on the PK2100 or thePK2200.

Sample Master-Slave Programs

Figures 1-4 (beginning on the next page) are sample master-slave network programs.

```
/* master.c ************************************************************
*  The master sends a string of message to the slave.
*  The slave replies and the master prints the reply to the STDIO.
**********************************************************************/
   char rbuf[255];          // receive buffer
   char reply[40];          // reply buffer
   char message[40];        // message buffer
main(){
   int i, j, ercode;
#if (BOARD_TYPE==CPLC_BOARD) + ( BOARD_TYPE==L_STAR)
   uplc_init();
#endif
   op_init_z1( 16,rbuf,0);  // initialize at 19200 baud, receive buffer
                            // and as master (0)
   j = 1;
   for(;;){
   // wait 10000 counts before polling
      for( i = 0; i < 10000; i++ ) runwatch();
      sprintf( message,"Message # %d",j++ );
   // send message to slave 1 and wait for reply
      ercode = TalkToSlave(1,message,strlen(message),3,reply);
      if( ercode == 1 ){
         printf("%s\n", reply);    // reply is valid
      }else{
         printf("Link Failure or Bad Link\n");
      }
   }
}
```

Figure 1. Send Message and Print Reply

```
/*************************************************************************
int TalkToSlave( int slave, char* query, int len, int delay, char* reply ){
*************************************************************************/
   int i, ercode;
// sends message and waits for reply
   ercode = sendOp22( slave, query, len, delay);
// if reply is valid, copy the ASCII reply to reply buffer
   if( ercode == 1){
      for( i = 0; i < rbuf[0] - 2; i++ ){
         reply[i] = rbuf[1+i];
      }
      reply[i] = '\0';
   }
   return ercode;
}
```

Figure 2. Send Message and Wait for Reply

```
/* slave.c ***************************************************************
*  The slave waits for a message from the master.  The slave prints the
*  master's message to the  STDIO and also replies to the master.
************************************************************************/
   char rbuf[255];          // receive buffer
   char query[40];          // query buffer
   char reply[40];          // reply buffer
main(){
   int j;
#if (BOARD_TYPE==CPLC_BOARD) + (BOARD_TYPE==L_STAR)
   uplc_init();
#endif
// initialize for 19200 baud, receive buffer, slave 1
   op_init_z1( 16,rbuf,1 );
   j = 1;
   for(;;){
      runwatch();
      if( masterquery(query) ){     // check for master query
         sprintf(reply,"%s, Slave Reply %d",query,j++);
      // sends back reply to the master
         replyOpto22(reply,strlen(reply),0);
      }
    }
}
```

Figure 3. Slave Waits for Message, Prints, and Replies

```
/************************************************************************
*  Check for master query.
*  Copy the message from the master to query if it is valid.
************************************************************************/
int masterquery( char* query ){
   int i;
   if( check_opto_command() != 1 ) return 0;
   for( i = 0; i< rbuf[1] - 2; i++ ){
      query[i] = rbuf[2 + i];
   }
   query[i] = '\0';        // put an end-of-string
   return 1;
}
```

Figure 4. Check for Master Query

Notes:

sendOp22 sets up the message for transfer using the 9th-bit protocol; it ***does not*** poll for the completion of the transfer. Transmission and reception of messages are done in the background interrupt routine.

While **network.lib** is based loosely on the **misticware** protocol from Opto 22, it is not compatible.

Z-World Corporate Headquarters
2900 Spafford Street
Davis, California 95616 U.S.A.
530.757.3737 • Fax: 530.753.5141

http://www.zworld.com