MODBUS RTU Programming—A MODBUS RTU Master/Slave Example Using Serial Connections

This is a simple MODBUS RTU master/slave example based on Connection Programming that utilizes MODBUS RTU functions. In this example, the MODBUS RTU master continuously sends MODBUS RTU requests of function code 3. The field for the number of registers, which ranges from 1 to 10, is also sent. The MODBUS RTU slave responds to each received MODBUS RTU request with simulated register values.

A. MODBUS RTU Slave Example

The slave uses the *mbrtu_packet_digest* function from within the dispatch callback function to get a MODBUS RTU request. Next, it parses the received request to generate a simulated MODBUS RTU response PDU and then uses the *mbrtu_packet_format* function to compose the corresponding MODBUS RTU response ADU. Finally, the response ADU is sent to the master.

```
/*  This user-defined function is called after the user program receives
    a data packet. It processes the content of the packet and then sends
    the simulated result back to the master */
static int
mbrtu_slave_dispatch (MHANDLE conp, void *private_data, DATAPKT *dpkt)
{
    unsigned char *buf = (unsigned char *) private_data;
    int consumed;
    unsigned char mbrtu_buf[256];
    int len;
    unsigned short slave_id;
    unsigned short Modbus_id = 10;

    len = mbrtu_packet_digest(dpkt->packet_data, dpkt->packet_size);
    if (len <= 0)
    {
        /* MODBUS ADU packet not complete */
        slave_id = dpkt->packet_data[0];
        if (slave_id == Modbus_id) /* wait */
            consumed = 0;
        else /* discard */
            consumed = dpkt->packet_size;
    }
    else
    {
        consumed = dpkt->packet_size;
        slave_id = dpkt->packet_data[0];
        if (slave_id == Modbus_id)
        {
            /* process MODBUS RTU request packet and get response PDU */
            len = mbrtu_parser(dpkt->packet_data, len, buf, sizeof(buf));
            /* format response ADU */
            len = mbrtu_packet_format(buf, len, mbrtu_buf);
            /* send MODBUS RTU response */
            if (len > 0)
                connection_send_data(conp, mbrtu_buf, len);
        }
    }
    dpkt->packet_consumed = consumed;

    return consumed;
}
```

B. MODBUS RTU Master Example

The master uses the *mbrtu_packet_digest* function from within the dispatch callback function to get a MODBUS RTU response. Next, it parses the received response and displays register values on the screen, and then uses the *mbrtu_packet_format* function to compose a new MODBUS RTU request ADU. Finally, the request ADU is sent to the slave.

```
/*  This user-defined function is called after the user program receives
    a data packet. It keeps sending a packet to the slave */
static int
mbrtu_master_dispatch (MHANDLE conp, void *private_data, DATAPKT *dpkt)
{
    int consumed;
    unsigned char *packet;
```

```
	int len;
	unsigned short count, d;
	int i;

	/* obtain a MODBUS RTU response packet */
	len = mbrtu_packet_digest(dpkt->packet_data, dpkt->packet_size);
	if (len <= 0)
	{
		/* MODBUS ADU packet not complete */
		consumed = 0;
	}
	else
	{
		consumed = dpkt->packet_size;
		packet = dpkt->packet_data;
		/* display response */
		packet++; /* skip slave ID */
		if (*packet > 0x80)
		{
			packet++;
			printf("exception=%x\n", *packet);
		}
		else
		{
			packet++;
			count = *packet;
			count >>= 1;
			packet++;
			for (i = 0; i < count; i++, packet +=2)
			{
				memcpy(&d, packet, 2);
				d = BSWAP16(d);
				printf("%d ", d);
			}
			printf("\n");
		}
#ifdef WIN32
		Sleep(1000);
#else
        sleep(1);
#endif
        send_request(conp);
    }
    dpkt->packet_consumed = consumed;

    return consumed;
}
```