

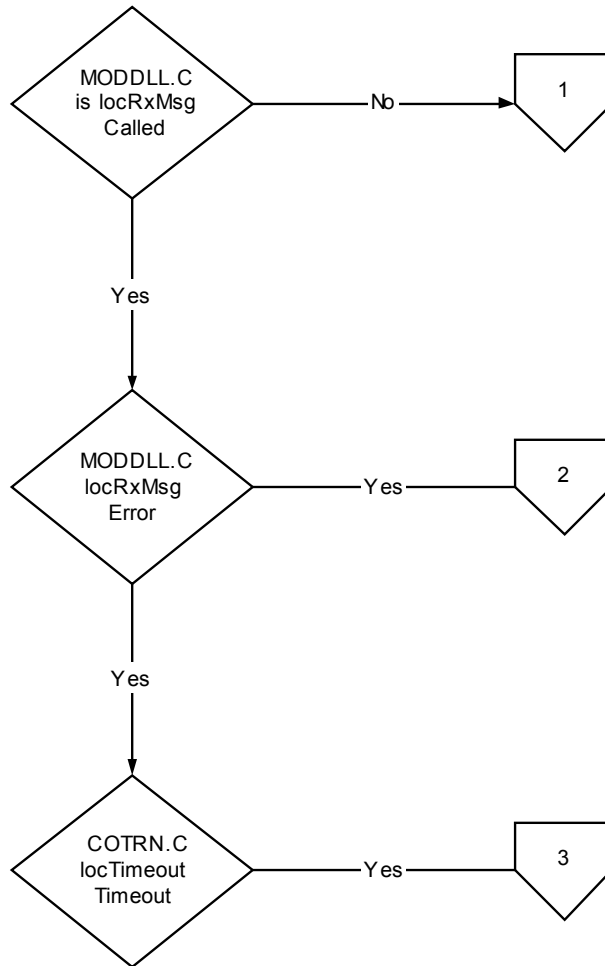


Applied Systems Engineering, Inc.

Technical Note #45
Debugging MODBUS Message Handling

About MODBUS message handling

When the MODPT is running in Master mode the MODPT will retransmit MODBUS messages if a proper response is not received from the remote device. This document describes test points that can be examined in the MODPT code to determine what has caused the problem. The following flowchart illustrates the MODPT modules and routines that need to be tested in determining the source of the communication problem.



The remainder of this document will describe what to look for in order to determine the source of the communication problem.

1 Routine locRxMsg in MODDLL.C is not called.

The first test is to determine if the routine locRxMsg in MODDLL.C is called. If this routine is called go to step 2, otherwise read this section. The routine locRxCompletion in MODDLL.C is called every time the COMM declares the event EVT_READ_COMPLETE. Routine locRxCompletion parses the new bytes received from COMM and determines if the data link message is assembled or more bytes need to be read from COMM. If the entire data link message is assembled the routine locRxMsg is called to process message security. If an error occurs at any step of the message parsing or the correct number bytes of the data link message are not received in sequence locRxCompletion will discard the

TechNote 45: Debugging MODBUS message handling

current message and attempt to process the next data link block in the receive stream.

Checking COMM status

Before parsing the incoming bytes `locRxCompletion` checks the status of the last COMM I/O. This status was returned to the MODPT through the COMM property `IOC_RXSTATUS`. The number of received bytes (property `IOC_RXLEN`) is also compared to the number of bytes the MODPT tried to read. If the number of bytes received from COMM does not match the number of bytes expected from COMM the current Message is discarded. The test on the integrity of the COMM data is as follows:

```
COPRT.C\locRxCompletion /* Called when EVT_READ_COMPLETE */

    If ( IOC_RXSTATUS < 0 )
        Then Send Error to MODDLL.C (pSok->Status < 0 )
    If ( IOC_RXLEN != CommRead length ) Then
        Send Error to MODDLL.C (pSok->Status < 0 )

MODDLL.C\locRxCompletion(, LPSOKIO pSok )

    If (pSok->Status < 0) Then
        Reject current Data Link Message
    Else
        Parse bytes received from COMM
        If message assembled call locRxMsg
```

If `pSok->Status < 0` then the user's COMM is returning an error on the last read posted by the MODPT. If `pSok->Status = 0` then the MODPT read was successfully performed by COMM. Since `locRxMsg` is not called the bytes returned by COMM must not be valid data bytes for a MODBUS message. The data link message being assembled is stored in `rx->Buf` by `locRxCompletion`. This buffer can be examined it should contain all of the assembled bytes in the MODBUS data link message.

2 Routine `locRxMsg` in `MODDLL.C` is called but returns with error.

If `locRxMsg` in `MODDLL.C` is called but returns an error read this section. If routine `locRxMsg` processes the data link message successfully go to step 3. The routine `locRxMsg` processes an assembled MODBUS data link message. It checks the security of the message. If the security check fails the message is discarded and `locRxMsg` returns with error. The buffer `rx->Buf` contains the fully-assembled MODBUS data link message. Examination of the buffer should reveal the incorrect data bytes in the message.

3 Routine `locTimeout` in `COTRN.C` is called and reports a timeout

While the data link is assembling the MODBUS response from a remote device the transaction manager is timing the transaction. If the transaction timer is small the transaction manager might time a MODBUS transaction out even while bytes are being assembled at the data link. Transaction timeout occurs in `locTimeout` in `COTRN.C` as follows:

```
if ( ! r->Timer )
{
    GPTTRACE1( GPTTOLINE(this), DEBUG_TRN | DEBUG_INFO,
```

TechNote 45: Debugging MODBUS message handling

```
        "TRN Timeout (id=%ld)\n", r->TranId );

    if ( ( r->Attempts <= r->Retry ) && ( r->Retry != 0 ) )
    {
        locSwitch( this, TRN_QUEUE_REQUEST, r->Item.Key, r );
        return locRequest( this );
    }

    r->Completion = GPT_ERROR_TIMEOUT;
    rc = (*this->m_Prop.Cfg.Confirm)( this->m_hPro, r );
    locEnd( this, r, rc );
    break;
}
```

If the completion status is set to timeout it might be that the transaction timer for the MODBUS protocol is set to small. The application timeout is controlled by the MODBUS property `IPROP_MODBUS_APPLICATION_TIMEOUT`. Increasing the timeout value might prevent the transaction from timing out.

Data Flow

Following is the data flow when a valid MODBUS response is received from a remote device.

```
COPRT.C\locRxCompletion    // Process COMM event EVT_READ_COMPLETE
MODDLL.C\locRxCompletion  // Assemble MODBUS message
MODDLL.C\locRxMsg         // Check Security of MODBUS message
COTRN.C\locIndicationCb   // Get next MODBUS application message
COTRN.C\locResume         // Resume transaction when response received
MODMASTER.C\locConfirm   // Parse the response received
```