Modbus+ support

Contents

- Brief Introduction to Modbus Protocol for Modbus Plus
- Feature of ModiconSV85 VME interface board
- Point to Point Message Transaction
- Interfacing the driver to EPICS records
- The driver (ModiconSV85.c)
- The device support (devModbusPlus.c)
- Status of utilization
- How to use

Modbus Protocol for Modbus Plus

Based on Master and Slave Query-response cycle scheme

- Typical Modbus command: Read Input Status

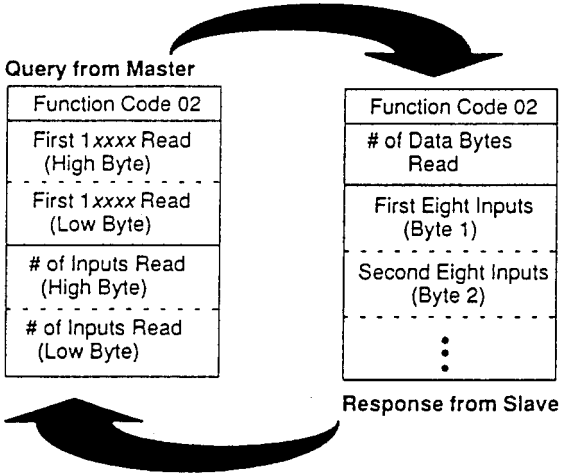


Figure 6 Query–Response Cycle for Modbus Function 02

Feature of ModiconSV85 VME board

- Modicon SV85 VME interface board can be
  - a master of 8 slave nodes
  - and a slave of 8 master nodes at the same time.
- Functions as an A24 or A16 slave only
- DMA capabilities are not supported
- VME bus interface consists of:
  - dual port RAM interface
  - control and status registers
  - VME bus interrupt interface

Block Diagram of SV85

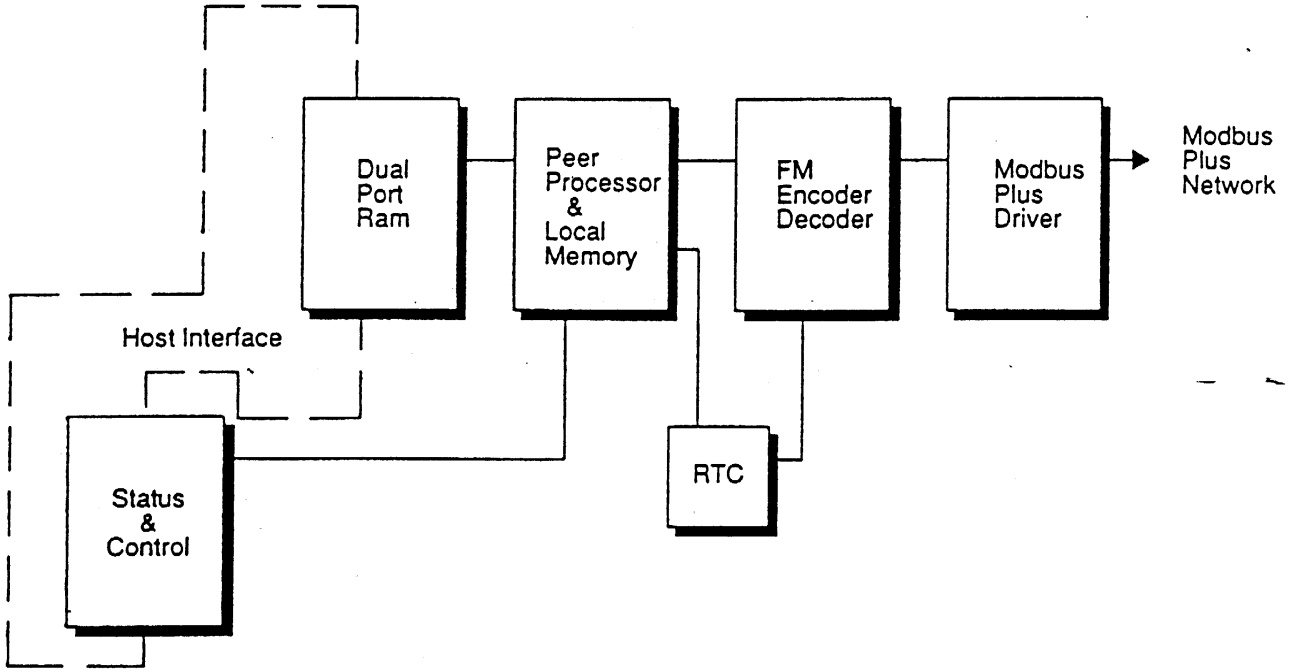


Figure 4-1 Hardware Block Diagram

Network wiring

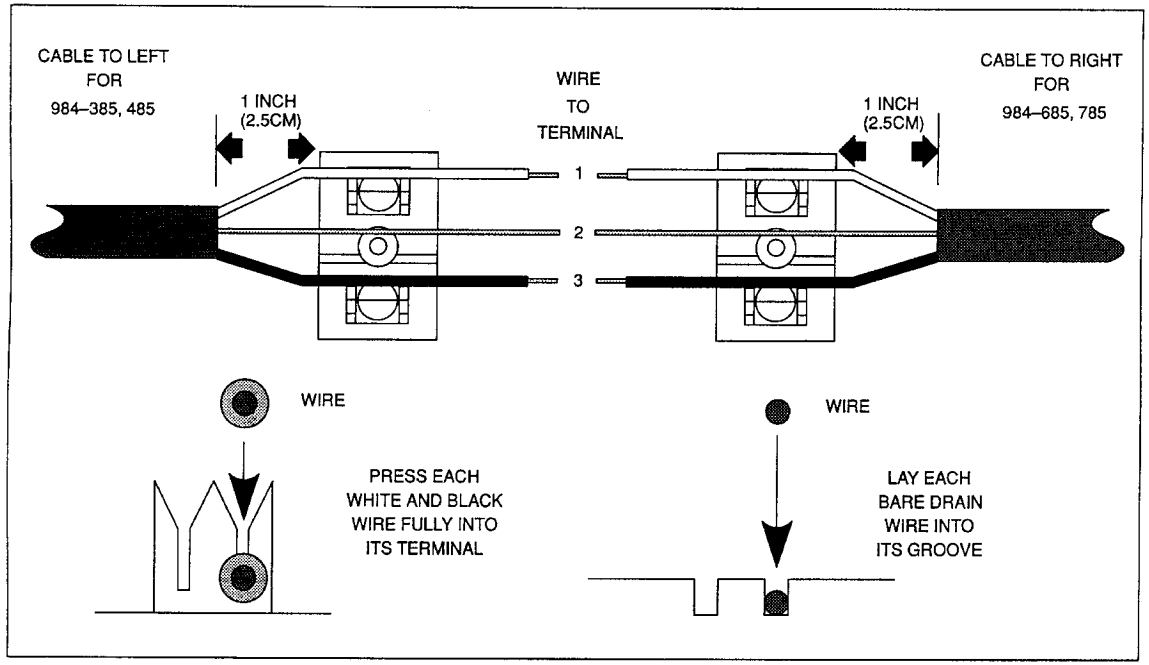
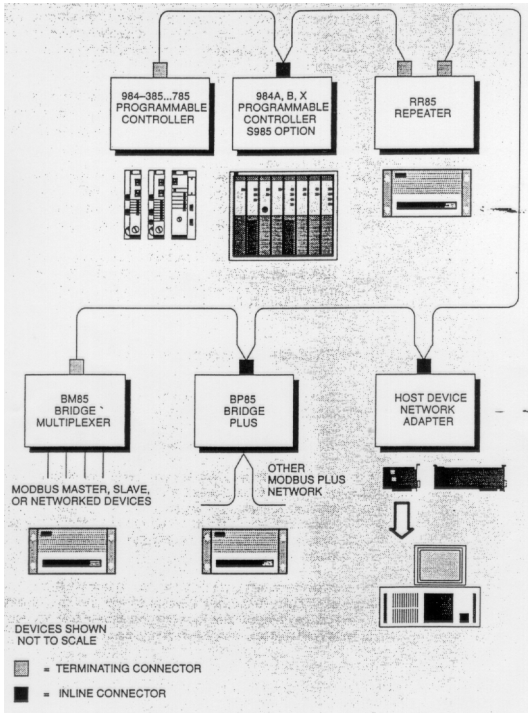


Figure 52 Connecting the Wires

Network Configuration



Point to Point Message Transaction

- The interface board has a peer processor.
- IOC can conduct the Modbus transaction by giving the peer processor a set of Interface Commands.
- An Interface Command(to the peer processor) consists of a Command Opcode and, if necessary, Command Data.
- The Command Data includes a Modbus command(to be sent to another node) as a part of it, if necessary.

Message Transaction (continued)

- An example: Sending a Modbus Command to a Slave Node &Getting a Modbus Response from the Slave Node.
- IOC gives the peer processor an Command Opcode ``Put Master Command to Output Path'', and then, a Command Data (..., routing path, ..., Modbus command).
- The peer processor sends the Modbus command to another node according to the routing path.
- When the peer processor gets a Modbus response from the remote node, the peer processor informs the IOC of the completion of the transaction by Host Service Request interrupt.

Message Transaction (continued)

- Before the IOC gets the response from the peer processor, the IOC must inspect which transaction did the peer processor complete.
- (the peer processor can be a master of 8 slave nodes at the same time)
- An interface command ``Get Host Service Request'' is used for the purpose.
- The IOC decides the transaction path to be processed, and issues ``Get Master Response from Output Path'' to get the response from the peer processor.

the driver - EPICS records intereaction

- In general, PLC devices hold a lot of signal points.
- A signal point or a set of signal points corresponds to a database record in EPICS.
- However, it is very inefficient to execute Modbus transaction for each signal point(with asynchronous I/O mechanism of device support).
- Whole signal points can be read by single Modbus transaction(if the total number of signal points does not exceed the max of 2000).

5. The driver (ModiconSV85.c)

- Supports Point to Point Message Transaction only.
- Global Database Transaction is not supported at present.
  - "Global" means that a node can broadcast the data to all the other nodes.
  - "Database" means that each node maintains a table of global data sent by every other node.
  - The global data is contained in the token frame.
  - PLCs can inform the IOC of an event by using this mechanism.
  - Support of the Global Database Transaction is possible future improvement.
- Supports Reading Data and Writing Data only.
- Programming PLCs from an IOC is not supported.
- The IOC can be only a master, can not be a slave of a PLC.

ModiconSV85.c (continued)

- Supports concurrent 8 transactions to be in process at the same time.
- Each transaction is managed by its own task.
- A SrqHandler task controls progress of those tasks.
- Can notify events to EPICS by the post_event() call.
  - If some changes are detected in the memory, the driver issues post_event() to inform EPICS of it.
  - Any change in the bit-maps results in the processing of all records in the relevant node (a post event number is given to a node).

The device support (devModbusPlus.c)

- Currently supports only Bi and MbbiDirect
- Uses VME_IO as the device type
- At device support initialization, the initialize routine look up a global symbol ``pModbusPlus''(it points a structure allocated at the driver initialization).

Status of utilization

- The ModiconSV85.c and devModbusPlus.c were utilized in KEK AR/BT operation in last March for the first time.
- Currently, it has been running stably.
- It will be utilized in magnet status monitoring system
- and safety control system of the KEK-Bfactory.

How to Use

Need to write a system description file.

plcConfig.h

```
int modbusPostEventNum = 1;
#define NUM_NODES        1
#define NUM_COMM         1
#define SOURCE_ADDR   0x02

static unsigned char routingPath[
NUM_NODES ][5] = { { 0x01, 0x00,
0x00, 0x00, 0x00 } };

static int numberOfCommands[
NUM_NODES ] = { 1 };
```

```
static unsigned char QueryFromMaster[
NUM_NODES ][ NUM_COMM ][5] = {
  {
    {
      0x02,   /* Read Input Status */
      0x00,   /* high */
      0x00,   /* low   from 0 */
      0x00,   /* high */
      0x36    /* low   54 bits */
    }
  }
};
```

Address specification

```
struct  vmeio {
        short  card;
        short  signal;
        char param[32];
 }
```

card : specifies node address

signal : specifies function code of the Modbus command

param : specifies address of point seen by ladder program

[, # of bits(mbbiDirect only)]

[: routing path, if necessary]

Address specification

Example:#47 S1 @127

- read status of #127 in node 47.
- The node 47 should be in the same network as the IOC.

Example:#47 S2 @5,4

- read input status of 4 bits from #5 in the node 47.

Example:#32 S2 @5,4:{ 24, 61, 32, 0, 0 }

- the destination is node 32.
- The Modbus command will be transmitted to node 32 through two repeater nodes, node 24 and node 61.