VMS (VAX/AXP)

Modbus Plus

Interface Library

Programmer's Manual

Release: 1.9

Technical Writer:

Earl D. Lakia
Senior Staff Engineer
IPACT Inc.

file: k:ipcmv3::\ipccommon\mbplus\user_doc\library.doc

Chapter 1
MODBUS Plus VMS Interface Library

1.1 Introduction

This document describes the Modbus Plus interface library, and its distribution with the Modbus Plus Device Driver for VAXVMS for the SQ85 (licensed by Modicon) or the Modbus Plus Device Drdiver for AXPVMS for the SA85 (licensed by IPACT). This document also provides some integration help in using this software along with some actual PLC rungs that provide examples of communication logic. The third chapter documents some utilities that were gathered from the driver distribution and written as tools for this software. These tools will help you diagnose system faults and verify system operations.

1.2 Interface Library

The Modbus Plus VMS interface library provides a set of callable routines that reduces the effort required to communicate with devices on the Modbus Plus network. This software is implemented as a shared vectored library that is linked into each application that desires to communicate with the devices on the network. Because it is vectored, upgrades to the interface library will not require the relinking or recompiling of application programs.

This software assumes the presence of the Modbus Plus device driver and the SQ85 or SA85 Modbus Plus gateway hardware. The user should also reference the Modicon document: "Modicon DEC Host Based Devices User's Guide", publication number: "GM-HBDS-002 Rev. B" as an additional source of information.

1.3 Shareable Image Entry Points

The MBP interface calls and error status messages are all prefixed with "MBP_". The MBP_SHARE sharable image contains the following entry points:

MBP_CLOSE_NET- Close all channels and deallocate all paths to the Modbus Plus Network.

MBP_FORCE_SINGLE_COIL- Force a single coil within a PLC on the Modbus Plus Network.

MBP_GET_DRIVER_STATISTICS- Return current driver statistics.

MBP_GET_NETWORK_STATISTICS- Return current statistics from a network controller on the Modbus Plus Network.

MBP_HOST_WRITEABLE_REGION_V- Provides writeable region for each path on a Host SQ85 adapter writeable by a remote master.

MBP_OPEN_NET- Open channels and create paths for the process to access Modbus Plus Network.

MBP_PRESET_SINGLE_REGISTER- Set a single holding register within a PLC on the Modbus Plus Network.

MBP_READ_GLOBAL_DATA- Read the global data from the Host's Modbus Plus controller.

MBP_READ_REGISTERS- Read registers or coils from a PLC on the Modbus Plus Network.

MBP_READ_UNSOLICITED- Dequeue a message from the unsolicited mailbox.

MBP_REGISTER_UNSOLICITED- Register for unsolicited slave messages from PLCs or other VAXs on the Modbus Plus Network. Allocate a mailbox and initiate the actual posting and functioning of the slave data path reads and responses.

MBP_REGISTER_UNSOLICITED_V- Register for unsolicited slave messages from PLCs or other VAXs on the Modbus Plus Network. Allocate a mailbox for each path of the SQ85 and initiate the actual posting and functioning of the slave data path reads and responses.

MBP_WRITE_GLOBAL_DATA- Write to the host controller's global data.

MBP_WRITE_REGISTERS- Write a group of registers or coils in a PLC on the Modbus Plus Network.

MBP_READ_EXTENDED- Read registers from extended memory files in PLC.

MBP_SUSPEND_UNSOLICITED- Suspend a process from receiving any further modbus slave messages for a particular path on a SQ85 host adapter.

MBP_RESUME_UNSOLICITED- Resume a process to receive further modbus slave messages for a particular path on a SQ85 host adapter.

MBP_WRITE_EXTENDED- Write registers to extended memory files in PLC.

Each of the above mentioned calls are documented in Chapter Two. Certain calls may be called to wait for the communication to complete by specifying a valid event flag and using the “W” call (e.g. MBP_WRITE_REGISTERSW). Most calls allow only a single function on a path to be active at a time. This is due to the design of the Modbus Plus network protocol.

1.3.1 Installation

This software is implemented as a shared executable library (a linkable object library is also supplied but usage should be discouraged). This method provides the ability to install new releases of the software without requiring the applications that use them to be recompiled or relinked. This software and the VMS device driver for the SQ85 or SA85 network controller is installed via VMSINSTAL. A sample VMSINSTAL session is included in the appendix. After installation, the logical "MBPLUS_" points to the directory of the installed software. The command file "MBP_STARTUP.COM" should be chained to by the system or application startup command procedures. MBP_STARTUP will define the logical names and install the shared library. The VMSINSTAL kit uses the current version and revision numbers to create a unique software distribution directory (e.g. SYS$COMMON:[MBP017]). The software is placed in the root of SYS$COMMON or SYS$SYSDEVICE.

1.3.2 Compile Requirements

The following text libraries are available in the kit directory. These libraries contain the Modbus Plus I/O codes, and structures required for accessing the Modbus Plus application library. These libraries are:

MBPLUS_:MBP_C.TLB- C header definitions
MBPLUS_:MBP_FOR.TLB- FORTRAN structure definitions

The following are contained in these libraries:

- MBPPEX- The process expanded region created by the MBP_OPEN_NET service.
- MBPSHARE_MSG- Definition of all error codes returned by the Interface Communications library.
- MP_FUNCS- Definition of the VMS I/O function codes for the Modbus Device Driver
- MBP_DEF- DEC C Function prototypes for all of the MBP_ functions (available only on OPEN VMS/AXP)
- MPDRIVER_MSG- Definition of all error codes returned by the MPDRIVER for OPEN VMS/AXP for theSA85 device.

For “C” programmers, the user should compile and link with the following command (assuming source module is: “test.c”):

$CC test+mbplus_:mbp_c.tlb/library

For FORTRAN programmers the user may specify the location of the text library in the include statement within the source module.

1.3.3 Linking Requirements

The user has a choice of linking to either a shared library or an object library. To link to the object library the linker commands are:

```
$LINK/EXE=test.exe SYS$INPUT/OPTIONS
    test
    mbplus_:mbp.olb/include=(mbpshare_msg,mpdriver_msg)
    mbplus_:mbp.olb/lib
    sys$library:vaxcrtl/lib
```

To link to the shared library, the following linker commmands should be used:

```
$LINK/EXE=test.exe SYS$INPUT/OPTIONS
    test
    mbplus_:mbp_share/share
    mbplus_:mbp.olb/include=(mbpshare_msg,mpdriver_msg)
    mbplus_:mbp.olb/lib
    sys$library:vaxcrtl/lib
```

The logical "MBPLUS_" is initialized by the command file created by VMSINSTAL kit (MBP_STARTUP.COM).

The object library method is provided to allow the programmer to link to an object library or a debug object library. The debug library allows the programmer to isolate faults that might be occurring within the shared library, and assist in reporting bugs to the developers. These two libraries are: MBP.OLB and MBP_DBG.OLB.

1.3.4 Example Programs

The subdirectory "MBPLUS_:[.Examples]" contains example programs that may be used to test the network or as a basis to code user applications. A command file, “MENU.COM” provides easy test of these routines. The user is cautioned that these example programs will write to the target Modbus Plus node causing undesirable results if the target node is controlling a process.

1.3.4.1 Process Privileges

Users of the Modbus Plus device driver must have the following privileges:

SYSNAME- To allocate a permanent mailbox for data messages from the PLCs.
PHY_IO- Physical I/O privilege to access the Modbus Plus device driver.

1.3.4.2 Process Quotas

This software does not require any abnormal quotas to function. However, the following are provided for fault analysis, and determining if normal quotas are not present.

AST Quota- For each possible I/O an AST entry is required. An I/O is present for each slave and master path that the caller allocates.

PGFLQUOTA- Page file quota. The process expanded region is allocated from the system page file. This is typically three pages plus one page for each path that is allocated.

WSQUOTA- See PGFLQUOTA

1.4 Modbus Network

This software differs in the handling of the Modbus Plus Network addressing as compared to the MODCOM III software library (this MODCOM III library is a predecessor of this software, and the calls available in the MODCOM III library have been replaced by these services).

This software assumes that the user maintains his own mechanism of cross referencing a particular PLC and the route path to a particular PLC. The route array is a five byte array that specifies the navigation to the target PLC. The last non zero byte in the array must be the Modbus Plus node address. For further explanation of the route array, consult the Modicon DEC Host Based Devices User's Guide.

Possible methods of maintaining the routing database are program data statements, include definition files, macro source code, or RMS keyed files.

1.5 PLC Registers and Coil Numbering

All of the calls provided by the Modbus Application Library use the same numbering system as the actual PLC itself. The user need not bias or normalize any of the registers or coil number before passing the register address to the subroutine. This allows the PLC programmer and the VAX programmer the ability to converse using identical register and coil numbers. The software examines the address, such as 40001, and knows that this is the first holding register, or 00001 is the first coil status.

1.6 PLC MSTR Blocks

The PLC MSTR requires a routing path, and a Modbus Plus function code (along with other parameters). The routing path specifies the target slave node that can be another PLC or the VAX. If the target is a VAX, then the PLC must also specify in the next route register the host data slave path the message is to be sent to.

The preferred method for mapping the slave data paths and VAX Host node addresses is to have the VAX hosts specify these parameters to the PLC in its holding registers or global data area. By using this methodology, the PLC does not need to maintain this part of the link management. This has the following benefits:

- The programmers of the PLC need not worry about communicating changes to the programmers of the VAX unless agreed upon locations of the link management registers are changed.
- The ability to develop redundant VAX hosts without changing logic in the PLC.
- The ability for the VAX programmer to partition and even distribute functionality to multiple programs running on either the same or different VAXs.

1.6.1 PLC MSTR Example

Assuming that there are eight PLCs (Modbus Plus addresses 1 to 8) and two VAXs (Modbus Plus addresses 20 and 21). Each of the PLCs has two MSTR blocks. The first MSTR block is responsible for sending a transaction containing completed counts and statistics about a product just made to the "Production Management" VAX. The second MSTR block is responsible for uploading significant event and current process history to a process "Process History" VAX. In the event that either of the VAX are unavailable, the other VAX assumes the functions of the failed VAX. In each PLC, the following holding registers are defined:

1. MSTR block one route byte 1 (node)
2. MSTR block one route byte 2 (data path)
3. MSTR block two route byte 1 (node)
4. MSTR block two route byte 2 (data path)

In normal operation the Production Management VAX writes its Modbus Plus address in the 40001 holding register for all of the PLCs. The Production Management VAX allocates five paths and writes the returned path numbers from the MBP_OPEN_NET call (See: MBPPEX structure) in the 40002 holding registers of the eight PLCs (assigning two PLCs per data path).

In a similar fashion, the Process History VAX would write its node address into the 40003 of the PLCs and the allocated data paths into the 40004 holding registers. In the event of a failure of one of the VAXs, the failed application would begin execution on the other VAX (we assume a clustered or shared type of VAX

environment). In this case, the appropriate holding registers would be loaded to direct all of the MSTR blocks to the correct VAX.

During the time between when a VAX failed and the application was restarted on the BACKUP VAX, the MSTR blocks would terminate unsuccessfully. User written logic might choose to zero the VAX node and data path bytes and not execute the MSTR block until both are non-zero again. The zeroing of the node and data path in the MSTR route holding registers may be triggered by five or ten consecutive errors from the MSTR block. The MSTR would be reenabled when the backup VAX wrote the new node and path to the holding registers.

A sample PLC program is shown in the appendix.

1.7 Paths

Communication that actually requires datagrams over the Modbus Plus network require a path. Depending on the type of node, the number of paths differ. A path can be considered like an end to end socket that two Modbus Plus nodes may communicate through. For most Modbus Plus nodes, the paths are simply a managed resource and the first path available on a node is used when needed. For the VMS environment, slave data paths are addressed uniquely, and their path number must be specified by the master Modbus Node as the last route parameter. Master paths in the VMS environment are allocated and managed by the device driver.

The paths allocated by the MBP_OPEN_NET are allocated from a free pool (of which there are eight) maintained by the driver (see IO$_ALLOC function in the Modicon "DEC Host Based Devices User's Guide"). The actual allocated master and slave paths are stored in the process expanded region and returned to the caller by the MBP_OPEN_NET call. Because the driver allocates the paths, the paths may not be allocated sequentially!

All the routines documented in this manual that specify a path reference the path index. Typically, the path index will be equal to the actual path in a single program access system, but not always. This has little effect except for slave paths since the route array specifies the target PLC and there is no advantage to selecting a particular master path over another.

1.8 Master Paths

Each SQ85 or SA85 supports up to eight master paths. The MBP_OPEN_NET allocates available master paths in sequential order. A master path is used when the VAX is the master and the PLC or other remote Modbus Plus node is the slave. Most functioning by the VAX uses the Master paths.

1.9 Slave Paths

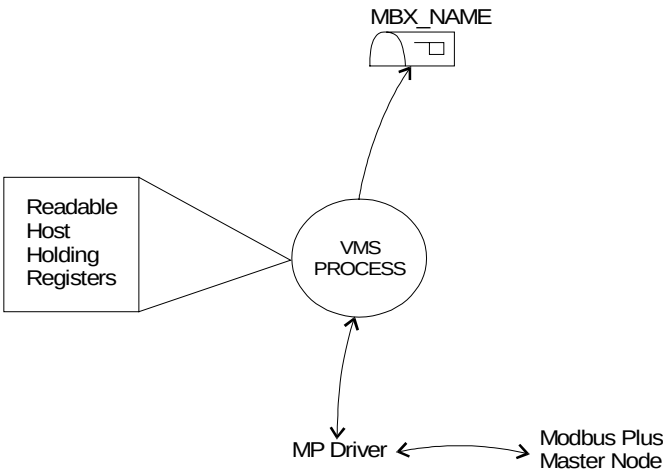
Each SQ85 or SA85 supports up to eight slave paths. The MBP_OPEN_NET allocates available slave paths in sequential order. A slave path is used when the VAX is the slave and the PLC or another Modbus Plus node is the master. The PLC uses the MSTR instruction to read or write to or from the VAX. The PLC writes are directed to one of the following (only one of the following methods is supported at the same time by the same VMS process):

1. A single VAX mailbox for all paths with optional readonly slave region (MBP_REGISTER_UNSOLICITED)

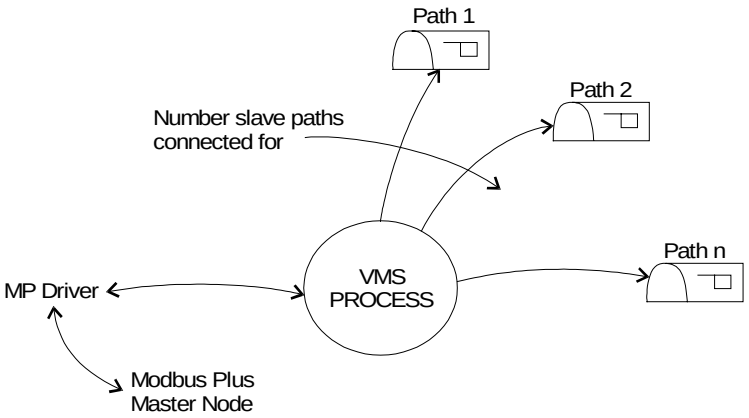
2. A VAX mailbox by path (MBP_REGISTER_UNSOLICITED_V)

3. An array or region by path for read or write access (MBP_MBP_HOST_WRITEABLE_REGION_V)

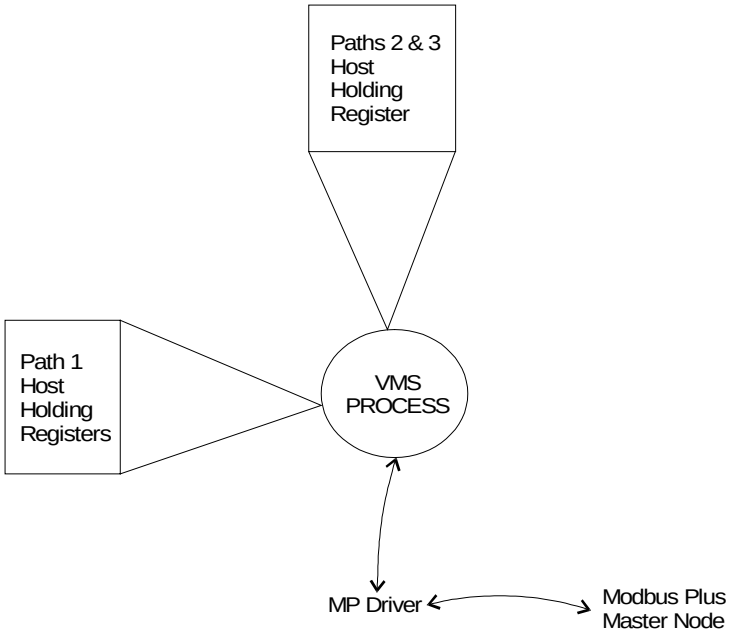
Each of these three options are shown in the following diagrams.



MBP_REGISTER__UNSOLICITED



MBP_REGISTER__UNSOLICITED_V



MBP_HOST_WRITEABLE_REGION_V

The PLC specifies which slave path is to be used in the second route parameter of the MSTR instruction. If the mailbox option is used and the mailbox becomes full, then the PLC is returned an error.

If the PLC wishes, it may read from any of the paths from a single region if it is defined. In this case the VAX emulates holding registers similar to a PLC with the first location addressed as register is 40001. This option is only supported with the MBP_REGISTER_UNSOLICITED routine call.

Since the PLC (Programmable Logic Controller) MSTR instruction specifies the actual path as part of its routing information, the VMS host must either tell the PLC which path is to receive the message (using technique described earlier) or the VMS host must allocate known slave paths such that the PLC can be programmed to write to known slave data paths in the VMS host. Defined slave paths are done by specifying an optional slave path vector array, CTRL_PATHS, in the MBP_OPEN_NET call.

1.10 Non PLC Modbus Nodes

It is possible to have Modbus Nodes which are not programmable Controllers. Typical non-Modbus nodes are either other VMS hosts, or operator displays. A typical application is where the VMS host provides supervisory information such as order information, customer names, product descriptions, and other data not needed for the actual operation or control of the operations. IPACT has used this feature with both UNICEL© marketed by Modicon, Factory Link© by USData, and InTouch© by Wonderware. Operator screens can be made that display information seamlessly without regard to the source of the information.

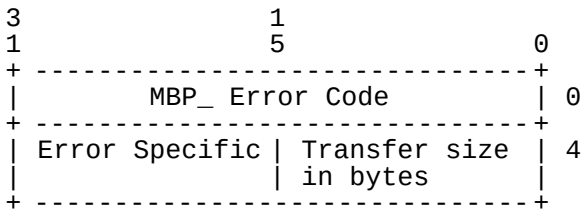
1.11 VAX and PLC Data Byte Order

The Modicon PLCs and the VAX store data bytes in different order. For data read or written to registers in the PLC, this software will automatically swap the data for the register data. For example, if a holding register in the PLC contained a simple counter and it currently read "2011" (hex), the same sixteen bit data word read by the VAX would have been "1120" (hex) if this software did not swap the data. This byte swapping is only performed on register data (MBP_READ_REGISTERS, MBP_WRITE_REGISTERS, MBP_WRITE_EXTENDED, MBP_READ_EXTENDED).

The unsolicited data from the PLC is placed into the mailbox as it is received from the PLC on the Modbus Plus Network. This software is unable to determine if the data should be byte swapped when it is received from the PLC. Therefore, the responsibility for this determination must be resolved by the programmer of the PLC and the VAX programmer.

1.12 Application Status Returns

All of the calls to the Application Library return status to the caller as the value of the function call similar to normal VMS standards. If the a status block is returned, it's format is as follows:



The following definition is contained in the MBP_C and MBP_FOR text libraries as: MBP_STATUS.

1.13 Contention and Synchronization Issues

This software assumes that the caller maintains all control and access to the PLC registers. If there are multiple writers, then these writers must develop locking techniques that will prevent one writer from overwriting another's output. This should be considered if the MBP_WRITE_REGISTERS calls are used by more than one process on one or more nodes.

If host holding register are to be used, the synchronization of the reads and writes to the host holding registers are the responsibility of the user. VMS locking and synchronization techniques can be used for this purpose. The process that allocates the slave paths provides all access for remote Modbus Plus master nodes via AST routines within the Modbus Plus Interface Library. Controlling the AST recognition of this process will control access to the host holding registers by the Modbus Plus master nodes.

> Note: See VMS documentation on process AST states, lock manager, event flags, and other methods of process synchronization. To disable AST recognition, see the SYS$SETAST VMS system service. Additionally, a process can have only one AST active which in itself can be used for synchronization (If changes to the region only occur at AST level, a consistent view of the region can be provided for remote Modbus Plus nodes).

1.14 Process Expanded Region

Each process that uses the Modbus Plus Application Interface library has a group global section created when the MBP_OPEN_NET is called. This region is used by the Modbus Plus Process Monitor utility (see utilities chapter). It also provides the ability for another process to enable or disable the reception of slave messages from a PLC master (see: MBP_RESUME_UNSOLICITED and MBP_SUSPEND_UNSOLICITED).

Chapter 2
MBP Application Library Calls

2.1 MBP_CLOSE_NET

Disconnect Modbus Plus Network.

FORMAT **MBP_CLOSE_NET (MBPPEX)**

RETURNS
VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS
MBPPEX
type: structure
access: read/write
mechanism: by reference

The structure that was returned by the MBP_OPEN_NET call.

DESCRIPTION This subroutine will deassign the channels and deallocate the paths that were used for communicating with the Modbus Plus Network by the calling process. The process expanded P0 space that was allocated is deleted along with the group global section. This routine does not need to be called if the process is exiting, as the normal VMS I/O rundown will release all paths and channels that were allocated by the process.

CONDITION VALUES RETURNED	MBP_NOTINITIALIZED	MBPPEX structure is invalid
	SYSTEM SERVICE	Errors from (SYS$DEASSIGN, SYS$QIO, SYS$DELTVA)

2.2 MBP_FORCE_SINGLE_COIL

Force a single output coil.

FORMAT **MBP_FORCE_SINGLE_COIL (MBPPEX,PATH,ROUTE,COIL_NUMBER, STATE,STATUS,EFN[,MTMO])**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below). The final status on a normal successful call is the status returned from the SYS$QIO call to the Modbus Device driver.

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

The structure that was returned by the MBP_OPEN_NET call.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is to be directed to. Must be between one and the number allocated by the MBP_OPEN_NET call.

ROUTE
type: five byte array
access: readonly
mechanism: by reference

This is an array filled in by the caller that specifies the route path to the particular PLC. The content of this array requires knowledge of the connected Modbus Plus network devices and their addresses.

COIL_NUMBER
type: word
access: readonly
mechanism: by value

The coil number to force in the addressed PLC.

STATE
type: word
access: readonly
mechanism: by value

New state of the coil (0 or 1).

STATUS
type: structure MBP_STATUS
access: write
mechanism: by reference

Final completion status as returned by the Modbus Plus device driver. See QIO condition codes in Modicon DEC Host Based Devices User's Manual.

EFN
type: long word
access: read
mechanism: by value

Event flag to set when I/O is complete.

MTMO
type: long word
access: read
mechanism: by value

Modbus Read Master Response timeout value in seconds.
Must be greater than two. The

DESCRIPTION

This subroutine sets a single output register using the specified path. The event flag is used for I/O synchronization and if none is specified, event flag zero is used. The event flag will be set when all of the registers requested to be read have been read. To satisfy the request, this routine may do more than a single Modbus transfer. This routine uses ASTs to process all I/O completions, and therefore must have adequate AST quota.

CONDITION VALUES RETURNED	MBP_NOTINITIALIZED	MBPPEX structure not initialized
	MBP_IONOTCOMP	I/O not complete on channel
	MBP_NOCHANNEL	No channel open on path
	MBP_PATHINUSE	Path in use
	MBP_INVALIDEFN	Invalid event flag specified
	MBP_BADSTARTADDR	Invalid starting address
	MBP_BADPATH	Path specified not allocated or out of range
	SYSTEM SERVICE	From SYS$QIO call.

2.3 MBP_GET_DRIVER_STATISTICS

Return the driver statistics.

FORMAT **MBP_GET_DRIVER_STATISTICS(MBPPEX,BUFFER)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MBP_ error status codes.

ARGUMENTS

MEPPEX
type: structure
access: readonly
mechanism: by reference

The structure that was returned by the MBP_OPEN_NET call.

BUFFER
type: 316 Byte Array
access: Write
mechanism: By reference

Buffer to receive the driver status and statistics. See DEC Host Based Devices User's Guide for a description of the returned record.

DESCRIPTION

This subroutine requests the local device driver to return the driver statistics to the user supplied buffer. The layout of the 316 byte buffer is described in the Modicon DEC Host Based Devices User's Guide (see IO$_GET_SS Get Driver Status and Statistics). This routine completes synchronously. This call is done to the first channel that was allocated by the MBP_OPEN_NET service. This function does not require a path. It always uses the first master path allocated.

CONDITION VALUES RETURNED

MBP_NOTINITIALIZED	MBPPEX returned by MBP_OPEN_NET is corrupt or invalid.
MBP_NOCHANNEL	No channel open on path
SYSTEM SERVICE	VMS System Service Errors
Modbus Errors	See Modicon DEC Host Based Devices User’s Guide and the appendix

2.4 MBP_GET_NETWORK_STATISTICS

Get Modbus Plus counters and statistics.

FORMAT **MBP_GET_NETWORK_STATISTICS (MBPPEX,PATH,ROUTE,BUFFER,CLEAR,STATUS,EFN)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MBP_ error status codes.

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

Structure returned by MBP_OPEN_NET.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is to be directed to. Must be between one and the number allocated by the MBP_OPEN_NET call.

ROUTE
type: five byte array
access: readonly
mechanism: by reference

This is an array filled in by the caller that specifies the route path to the particular PLC. The content of this array requires knowledge of the connected Modbus Plus network devices and their addresses. This subroutine does not validate the route array.

BUFFER
type: 110 byte array
access: write
mechanism: by reference

Buffer to receive the Modbus Plus status and statistics from the SQ85.

CLEAR
type: word
access: readonly
mechanism: by value

If true (1) read and clear the statistics, else only read.

STATUS
type: structure MBP_STATUS
access: write
mechanism: by reference

Final status.

EFN
type: long word
mechanism: by value

Event flag for I/O synchronization.

DESCRIPTION

This routine asks the SQ85 to return its Modbus Plus counters and statistics. By setting the "Clear" flag to true, the counters are optionally cleared. This entry executes synchronously.

CONDITION VALUES RETURNED

MBP_NOTINITIALIZED	MBPPEX structure not initialized
MBP_BADPATH	Path specified not allocated or out of range
MBP_PATHINUSE	Path is in use
MBP_NOCHANNEL	No channel open on path
MBP_ INVALIDEFN	No channel open on path Invalid event flag specified
MPB_IONOTCOMP	I/O not complete on channel
SYSTEM SERVICE	VMS System Service Errors from QIO call.

2.5 MBP_GET_SLAVE_ID

Modbus Plus node to return its id.

FORMAT **MBP_GET_SLAVE_ID (MBPPEX,PATH,ROUTE,BUFFER,EFN)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MBP_ error status codes.

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

Structure returned by MBP_OPEN_NET.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is to be directed to. Must be between one and the number allocated by the MBP_OPEN_NET call.

ROUTE
type: five byte array
access: readonly
mechanism: by reference

This is an array filled in by the caller that specifies the route path to the particular slave PLC. The content of this array requires knowledge of the connected Modbus Plus network devices and their addresses. This subroutine does not validate the route array.

BUFFER
type: 9 byte array
access: write
mechanism: by reference

Buffer to receive the Modbus Plus nodes slave id and status. The format of the nine byte buffer is:

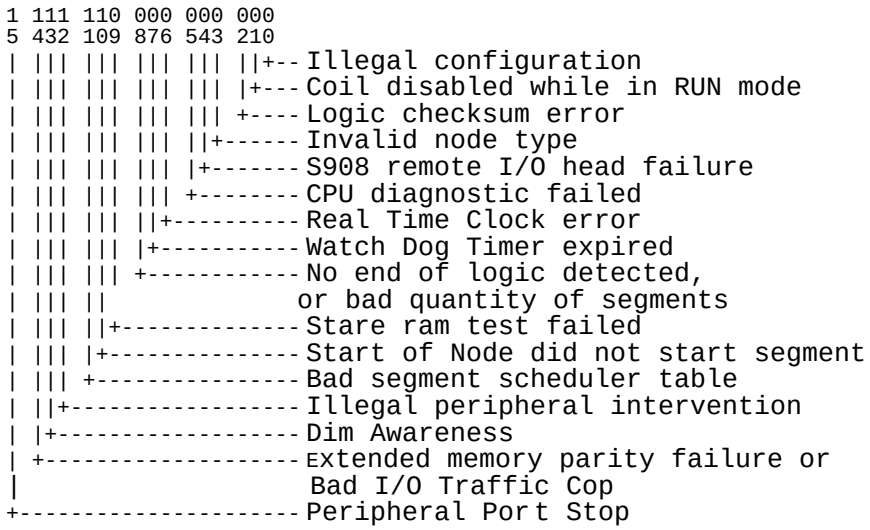
BYTE SLAVE_ID
BYTE LED
BYTE PAGE0
BYTE PAGEF
BYTE SEGMENTS
WORD CONTROLLER STATUS
WORD STOP CODE

DESCRIPTION This routine asks a particular Modbus Plus node to return its id. The user buffer should 9 bytes long. For more information, see the Modbus Plus Device Drivers Manual under function code 11. This call is synchronous.

The format of the controller status word is:

```
1 111 110 000 000 000
5 432 109 876 543 210
+ +++ +|| ||| ||+ ++|
   |   || ||| ||  | +- Memory downsized (0=no, 1= yes)
   |   || ||| ||  +--- Unassigned
   |   || ||| |+------ Battery Status (0=ok, 1= not ok)
   |   || ||| +------- Memory Protect (0= on, 1= off)
   |   || ||+--------- Run indicator (0= on, 1= off)
   |   || |+---------- Power on (1= on, 0= off)
   |   || +----------- Processor size (1= 16 bit node, 0= 24 bit node)
   |   |+------------- Single sweep status (0= off, 1=on)
   |   +-------------- Constant sweep status (0= off, 1= on)
   +------------------ Unassigned
```

The format of the stop code word is:



EFN
type: long word
mechanism:by value

Event flag for I/O synchronization.

CONDITION VALUES RETURNED	MBP_NOTINITIALIZED	MBPPEX returned by MBP_OPEN_NET is corrupt or invalid.
	MBP_BADPATH	The specified path is less than or greater than the number of master paths allocated by the MBP_OPEN_NET call.
	MBP_PATHINUSE	A request to do a second read or write to the master path, but the path is already active with a previous I/O request.
	MBP_NOCHANNEL	There is not channel or path for the passed path argument. The MBP_OPEN_NET most likely failed to allocate all of the desired paths. Check status of the MBP_OPEN_NET call.

2.6 MBP_HOST_WRITEABLE_REGION_V

Define VMS regions for Master writes by Programmable Controllers by path (VMS slave paths)

FORMAT **MBP_HOST_WRITEABLE_REGION_V (MBPPEX,REGION_ARRAY)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: write
mechanism: by reference

The structure to receive starting and ending address of the process specific data area used by the Modbus Plus interface subroutines (see MBPPEX in MBPlus.TLB).

REGION ARRAY
type: array of Quad word descriptors
access: readonly
mechanism: by reference

This structure should be dimensioned by eight (number of modbus slave paths). An entry should be filled in for each slave path allocated in the MBP_OPEN_NET call. Each quad word entry contains a standard VAX/VMS descriptor (only length and address are used).

DESCRIPTION

This subroutine allows the caller to set up a region to receive slave messages from devices on the Modbus Plus network. Each path can be assigned to a unique region. The caller must have already allocated slave paths via the MPB_OPEN_NET call. The user passes a descriptor (see VMS documenatation for this datatype) for each slave path the caller wants to be mapped to a region.

The regions may be identical. The service probes to ensure that the reason is accessible for write in the access mode of the caller. The passed addresses for the regions may the either process private, group global, or system global regions. Additionally, they may all map the same address space if desired. No locking is provided to protect the region from master writes or reads by the remote Modbus Plus nodes.

The service determines the length of the region from the descriptor and uses this to determine if a master read or write request from the remote Modbus Plus node is valid. A Modbus Plus execption is sent to the remote Modbus Plus Master node if the requested holding register is out of range. All regions are mapped as if the first register was 40001. Only the read and write of holding registers is supported.

The caller must determine if the slave paths allocated by the current process are to be written to a global region or it the data should be sent to a mailbox. A process can only support one method for responding to the Master Writes. If both functionality is desired, a second process must be present to service the other method. See the MBP_REGISTER_UNSOLICITED call as well.

The byte order and synchronization of data within the Host Holding Register regions are the responsibility of the user. The VMS process that calls MBP_HOST_WRITEABLE_REGION supports the actual data transfer between the Modbus Plus device driver at AST level. Therefore, disabling the AST recognition of this VMS process while the region is being modified will guarantee atomic and consistent data for Modbus Plus Master nodes reading data from the Host's holding registers. The method used must ensure that the process will be able to respond to the Master Modbus Plus node prior to the Master node timing out because no slave data response from the VMS host was received.

CONDITION VALUES RETURNED	MBP_NOSLAVE	No slave paths were allocated
	MBP_MBXFAIL	Unable to create mailbox (see secondary status, will contain the VMS system service code)
	MBP_SLVASSIGNED	Slave path already assigned for unsolicited messages
	MBP_INVALIDARG	Invalid argument
	MBP_SLVBUSY	Attempt to allocate second read on same slave path
	MBP_SLVREADFAIL	Slave read failure

2.7 MBP_OPEN_NET

Enable Modbus Communications.

FORMAT **MBP_OPEN_NET(MBPPEX,DEVICE,MASTER_PATHS, SLAVE_PATHS[,CTRL_PATHS])**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: write
mechanism: by reference

The structure to receive the starting and ending address of the process specific data area used by the Modbus Plus interface subroutines (see MBPPEX in MBPlus.TLB).

DEVICE
type: character
access: read only
mechanism: by descriptor

This specifies the name of the Modbus Plus interface (MPA1:, MPB1:, MPC1:, or MPD1:).

MASTER_PATHS
type: word
access: read only
mechanism: by value

This specifies the number of master paths the caller desires to allocate for its use. Depending on the controller, there are a certain number of paths available to be used. See Modbus Plus documentation for a description of "PATH". If the entire number of paths specified cannot be allocated, then none are allocated.

SLAVE_PATHS
type: word
access: read only
mechanism: by value

This specifies the number of slave paths to allocate for the caller for the purpose of receiving unsolicited messages. See "MBP_REGISTER_UNSOLICITED" subroutine entry.

CTRL_PATHS
type: word array
access: read only
mechanism: by reference

Optional controller slave path numbers for each slave path allocated. If not specified, the driver will be asked to allocate the next free controller slave path number. If the slave path is not available (e.g. in use by another application) the driver will return a path in use error code.

DESCRIPTION

This subroutine open channels and allocate paths to be used for communicating with the Modbus Plus network. This subroutine expands the callers P0 space to allocate working storage for future calls to the Modbus Plus network. The structure MBPPEX that is returned to the user must be passed on all subsequent calls to the other Modbus Plus software interface routines. This is normally the first call by the user application software.

The structure MBPPEX is defined in the MBP_C.TLB text library. The MBPPEX is filled with the actual slave paths and master paths that were allocated. The process expanded region is created as a group global temporary section. This allows other programs to map the region for analysis purposes while a user program is executing. The name of the region is the process name prefixed with the device name (e.g. MPA1_SCANNER Where SCANNER is the process name, and MPA1: is the device name). A process may connect to more than a single controller if desired using a unique MBPPEX structure for each controller.

All master paths are allocated by the driver. The slave paths are allocated starting with one unless the optional parameter: "ctrl_paths" is present indicating the associated actual SQ85 controler slave paths to allocate for each path.

The master and slave paths allocated may or may not start with master path one. It is possible that some but not all of the master or slave paths could not be allocated. Therefore, if this routine fails, the caller should do its normal error recovery and then exit.

The actual master and slave paths allocated are stored in the process expanded region, returned in the MBPPEX structure, and may be viewed with Modbus Plus process monitor utility.

CONDITION VALUES RETURNED	MBP_WRNGNMBRPATHS	Wrong number of paths
	MBP_MAPFAIL	Unable to create process region

MBP_ASSIGNFAIL	Unable to assign device
MBP_PATHFAIL	Unable to allocate a path

2.8 MBP_OPEN_PROGMASTER

Enable Modbus Communications using Program Master Paths

FORMAT **MBP_OPEN_PROGMASTER(MBPPEX,DEVICE,MASTER_PATHS)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: write
mechanism: by reference

The structure to receive the starting and ending address of the process specific data area used by the Modbus Plus interface subroutines (see MBPPEX in MBPlus.TLB).

DEVICE
type: character
access: read only
mechanism: by descriptor

This specifies the name of the Modbus Plus interface (MPA1:, MPB1:, MPC1:, or MPD1:).

MASTER_PATHS
type: word
access: read only
mechanism: by value

This specifies the number of progam master paths the caller desires to allocate for its use. Depending on the controller, there are a certain number of paths available to be used. See Modbus Plus documentation for a description of "PATH". If the entire number of paths specified cannot be allocated, then none are allocated.

DESCRIPTION

This subroutine open channels and allocates Modbus Plus Program Master paths to be used for communicating with the Modbus Plus network. This subroutine expands the callers P0 space to allocate working storage for future calls to the Modbus Plus network. The structure MBPPEX that is returned to the user must be passed on all subsequent calls to the other Modbus Plus software interface routines. This is normally the first call by the user application software. This routine provides Program Master versus Data Master paths. This routine is used normally for diagnostic programs where the data paths are all being used by user applications.

The structure MBPPEX is defined in the MBP_C.TLB text library. The MBPPEX is filled with the actual Program master paths that were allocated. The process expanded region is created as a group global temporary section. This allows other programs to map the region for analysis purposes while a user program is executing. The name of the region is the process name prefixed with the device name (e.g. MPA1_SCANNER Where SCANNER is the process name, and MPA1: is the device name). A process may connect to more than a single controller if desired using a unique MBPPEX structure for each controller. All Program Master paths are allocated by the driver.

The program master paths allocated may or may not start with program master path one. It is possible that some but not all of the master could not be allocated. Therefore, if this routine fails, the caller should do its normal error recovery and then exit. When a path is required for other library calls, the path numbers are relative to those allocated by the driver.

The actual program master paths allocated are stored in the process expanded region returned in the MBPPEX structure. Master Program paths have the form of "80" hex plus the path controller allocated path. The path information may be viewed with Modbus Plus process monitor utility.

CONDITION VALUES RETURNED

MBP_WRNGNMBRPATHS	Wrong number of paths
MBP_MAPFAIL	Unable to create process region
MBP_ASSIGNFAIL	Unable to assign device
MBP_PATHFAIL	Unable to allocate a path

2.9 MBP_PRESET_SINGLE_REGISTER

Set a single Programmable Controller register.

FORMAT **MBP_PRESET_SINGLE_REGISTER[W](MBPPEX,PATH,ROUTE, REGISTER_NUMBER,VALUE,STATUS,EFN[,MTMO])**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MBP_ error status codes.

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

Structure returned by MBP_OPEN_NET.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is to be directed to. Must be between one and the number master paths allocated by the MBP_OPEN_NET call.

ROUTE
type: five byte array
access: readonly
mechanism: by reference

This is an array filled in by the caller that specifies the route path to the particular PLC. The content of this array requires knowledge of the connected Modbus Plus network devices and their addresses. This subroutine does not validate the route array.

REGISTER_NUMBER
type: longword
access: readonly
mechanism: by value

The register number to force in the addressed PLC (40001 to 49999)

VALUE
type: word
access: readonly
mechanism: by value

New value for the register in normal VMS VAX 16 bit word format. The data will be byte swapped before written to the PLC holding register.

STATUS
type: structure MBP_STATUS
access: write
mechanism: by reference

Final status.

EFN
type: long word
access: read
mechanism: by value

Event flag to set when I/O is complete.

MTMO
type: long word
access: read
mechanism: by value

Modbus Read Master Response timeout value in seconds. Must be greater than two.

DESCRIPTION

This subroutine sets a single output register using the specified path. The event flag is used for I/O synchronization and if none is specified, event flag zero is used. The event flag will be set when the register requested to be set has been set. This routine uses ASTs to process all I/O completions, and therefore must have adequate AST quota. This routine will byte swap the user register before sending it to the holding register within the PLC addressed by the route array.
Both the synchronous and asynchronous versions are available.

CONDITION VALUES RETURNED	MBP_NOTINITIALIZED	MBPPEX returned by MBP_OPEN_NET is corrupt or invalid
	MBP_BADPATH	The specified path is less than or greater than the number of master paths allocated by the MBP_OPEN_NET call.
	MBP_PATHINUSE	A request to do a second read or write to the master path, but the path is already active with a previous I/O request.
	MBP_NOCHANNEL	There is no channel or path assigned to the specified path. The MBP_OPEN_NET - most likely failed to allocate all of the desired paths. Check status of the MBP_OPEN_NET call.
	MBP_IONOTCOMP	Previous I/O still outstanding on path.

2.10 MBP_READ_EXTENDED

Read Extended Memory of a Slave Modbus Node.

FORMAT **MBP_READ_EXTENDED[W](MBPPEX,PATH,ROUTE,START, COUNT,FILE,BUFFER,STATUS,EFN,FLAG[,MTMO])**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

The structure that was returned by the MBP_OPEN_NET call.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is to be directed to. The path must be between one and the number allocated by the MBP_OPEN_NET call.

ROUTE
type: five byte array
access: readonly
mechanism: by reference

This is an array filled in by the caller that specifies the route path to the particular PLC. The content of this array requires knowledge of the connected Modbus Plus network devices and their addresses.

START
type: long
access: readonly
mechanism: by value

Starting register in addressed PLC to beginning reading registers from. Valid range for each file is 60000 to 69999.

COUNT
type: word
access: readonly
mechanism: by value

The number of packed registers to read from the addressed PLC. If the number of registers exceeds what is able to read from a single Modbus Plus transfer, multiple Modbus Plus transfers will be done.

FILE
type: word
access: readonly
mechanism: by value

The extended memory of the PLC is broken up into 10000 registers each (and the remaining amount for the last segment), allowing registers in the range of 60000 to 69999 per file. The file number ranges from one (1) to the available memory assigned in the PLC for extended memory in ten thousand register increments.

BUFFER
type: array
access: write
mechanism: by reference

Caller array to receive the registers from the PLC.

STATUS
type: structure MBP_STATUS
access: write
mechanism: by reference

Final status.

EFN
type: long word
access: readonly
mechanism: by value

Event flag to set when I/O is complete.

FLAG
type: long word
access: readonly
mechanism: by value

If non-zero, then the data is not byte swapped when it is placed into the caller'

MTMO
type: long word
access: read
mechanism: by value

Modbus Read Master Response timeout value in seconds. Must be greater than two.

DESCRIPTION

This subroutine reads the registers from the extended memory files in the slave PLC on the Modbus Plus network. The data is byte swapped unless the FLAG byte is set to one. This routine make use of the Modbus function code 20. This routine is available in both the asynchronous and synchronous versions. By default, the bytes are byte swapped when they are recieved from the PLC unless the FLAG argument is specified. The user call may translate into multiple I/Os to the selected Modbus Plus device. The event flag will be set when all of the I/O is complete or an error is encountered.

The starting register number for an extended memory file registers is one less than for the other registers (the first register in an extended memory file is 60000, while the first holding register is 40001). Each extended memory file is partitioned to hold 10000 registers except for the last file which occupies whatever is left in the extended memory. Only a single extended memory file may be written per call.

CONDITION VALUES RETURNED

MBP_MBPPEXINVLD	MBPPEX structure is invalid
MBP_INVALIDARG	One of the call arguments is invalid, not readable, or not writable.
MBP_IONOTCOMP	I/O not complete on channel
MBP_NOCHANNEL	No channel open on path
MBP_BADSTARTADDR	Invalid starting address
MBP_PATHINUSE	The specified path already has an I/O active.
MBP_INVALIDEFN	Invalid event flag specified
MBP_NOTINITIALIZED	MBPPEX structure not initialized
MBP_BADPATH	Path specified not allocated or out of range
MPB_INVLPATH	Specified path is not between one and the number allocated by MBP_OPEN_NET

2.11 MBP_READ_GLOBAL_DATA

Read the 64 bytes of global data.

FORMAT **BP_READ_GLOBAL_DATA (MBPPEX,PATH,NODE,BUFFER,EFN,STATUS)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MBP_ error status codes.

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

Structure returned by MBP_OPEN_NET.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is to be directed to. This path number must be between one and the number allocated by the MBP_OPEN_NET call.

NODE
type: word
access: readonly
mechanism: by value

The Modbus Plus node the global data is requested of.

BUFFER
type: array
access: write
mechanism: by reference

Contains the buffer to recieve this controller's global data area.

EFN
type: long word
access: readonly
mechanism: by value

I/O event flag for synchronization.

STATUS
type: structure MBP_STATUS
access: write
mechanims: by reference

I/O completion status

DESCRIPTION

This subroutine will read the 64 bytes of global data.This routine operates synchronously as the data is local in the computer's controller. No byte swapping is to the data that is written to the controller. The Modbus Plus global data is maintained by all members of the Modbus Plus network.

CONDITION VALUES RETURNED

MBP_NOTINITIALIZED	MBPPEX returned by MBP_OPEN_NET is corrupt or invalid.
MBP_BADPATH	The specified path is less than or greater than the number of master paths allocated by the MBP_OPEN_NET call.
MBP_INVALIDEFN	Invalid event flag specified
MBP_PATHINUSE	A request to do a second read or write to the master path, but the path is already active with a previous I/O request.
MBP_BADPATH	Path specified not allocated or out of range
MBP_NOCHANNEL	No channel open on path
MBP_IONOTCOMP	I/O not complete on channel
MBP_NOCHANNEL	There is no channel or path assigned to the specified path. The MBP_OPEN_NET most likely failed to allocate all of the desired paths. Check status of the MBP_OPEN_NET call.

2.12 MBP_READ_REGISTERS

Read registers from a Modbus Plus slave node.

FORMAT **MBP_READ_REGISTERS[W](MBPPEX,PATH,ROUTE,START, COUNT,BUFFER,STATUS,EFN,FLAG[,MTMO])**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

The structure that was returned by the MBP_OPEN_NET call.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is to be directed to. The path must be between one and the number allocated by the MBP_OPEN_NET call.

ROUTE
type: five byte array
access: readonly
mechanism: by reference

This is an array filled in by the caller that specifies the route path to the particular PLC. The content of this array requires knowledge of the connected Modbus Plus network devices and their addresses.

START
type: long
access: readonly
mechanism: by value

Starting register in addressed PLC to beginning writing user registers to. The ten thousand position of "START" is used to determine if coil status (0xxxx), input status (1xxxx), holding registers (4xxxx), or input registers (3xxxx) are to be read.

COUNT
type: word
access: readonly
mechanism: by value

The number of packed registers to read from the addressed PLC. If the number of registers exceeds what is able to read from a single Modbus Plus transfer, multiple Modbus Plus transfers will be done.

BUFFER
type: array
access: write
mechanism: by reference

Caller array to receive the registers from the PLC.

STATUS
type: structure MBP_STATUS
access: write
mechanism: by reference

Final status.

EFN
type: long word
access: readonly
mechanism: by value

Event flag to set when I/O is complete.

FLAG
type: long word
access: readonly
mechanism: by value

If FLAG is set to 1, then the data is not byte swapped when it is placed into the caller's buffer.

MTMO
type: long word
access: read
mechanism: by value

Modbus Read Master Response timeout value in seconds. Must be greater than two.

DESCRIPTION

This subroutine reads the registers from the specified path. The event flag is used for I/O synchronization and if none is specified, event flag zero is used. The event flag will be set when all of the registers requested to be read have been read. To satisfy the request, this routine may do more than a single Modbus transfer. This routine

uses ASTs to process all I/O completions, and therefore must have adequate AST quota. The size of the buffer should be sized for either 8 bits per byte for discretes (0xxxx, 1xxxx), or two bytes per register (3xxxxx, 4xxxxx). All reads of registers are byte swapped to confrom to standard VMS sixteen bit word formats unless the FLAG argument is set to one.

This subroutine is available in the synchronous and asynchronous versions.

CONDITION VALUES RETURNED

MBP_MBPPEXINVLD	MBPPEX structure is invalid
MBP_INVALIDARG	One of the call arguments is invalid, not readable, or not writable.
MBP_PATHINUSE	The specified path already has an I/O active.
MBP_INVALIDEFN	Invalid event flag specified
MBP_NOTINITIALIZED	MBPPEX structure not initialized
MBP_BADPATH	Path specified not allocated or out of range
MBP_BADSTARTADDR	Invalid starting address
MBP_IONOTCOMP	I/O not complete on channel
MBP_NOCHANNEL	No channel open on path
MPB_INVLPATH	Specified path is not between one and the number allocated by MBP_OPEN_NET

2.13 MBP_READ_UNSOLICITED

Read unsolicited mail from a Master Modbus Plus Node

FORMAT **MBP_READ_UNSOLICITED(BUFFER,CHANNEL,EFN,SIZE [,MBX_SIZE])**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MBP_ error status codes.

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

Structure returned by MBP_OPEN_NET.

BUFFER
type: array
access: write
mechanism: by reference

Caller array to receive the buffer from the PLC. The message will contain at a minimum the the Modbus Plus header. The remainder of the message is dependent on the MSTR block in the particular PLC that sent the message. This buffer should be sized to MP$K_UNSOLICITED_MSGSIZE unless the user used a different size in the MBP_REGISTER_UNSOLICITED_V call.

CHANNEL
type: longword
access: readonly
mechanism: by value

Channel assigned to the unsolicited data mailbox.

EFN
type: longword
access: read
mechanism: by value

Event flag to be used for I/O synchronization.

SIZE
type: longword
access: write
mechanism: by reference

Return size of the message that was read in bytes. Includes complete header and data.

MBX_SIZE
type: longword
access: write
mechanism: by value

Optional parameter that specifies the size of the unsolicited mailbox.

DESCRIPTION

This routine allows the caller to read messages from the unsolicited mailbox set up by either the caller or some other caller.

This routine will read the message as placed by the process that has registered for the Modbus slave messages on a particular path (see MBP_REGISTER_UNSOLICITED).

This routine is passed the channel that is assigned to the unsolicited mailbox. If the process reading the mailbox is the same process that registered for the unsolicited messages, then it is the channel returned by MBP_REGISTER_UNSOLICITED, otherwise, it is the channel number assigned to a permanent mailbox created by the process that called MBP_REGISTER_UNSOLICITED and assigned by calling process using the VMS system service: "SYS$ASSIGN". If there is no message available, this service waits. This routine basically does the QIOW service to read the mailbox. If the user desires an asynchronous version, the user may do a normal QIO (consult the VMS mailbox driver for further details).

The message contains the Modbus header (the “C” header file: mbp_bufdef.h contains the definition of the Modbus Plus header) and the actual data message from the PLC master. No byte swapping is done on the data. If the data is true sixteen bit registers, the bytes may need to be swapped to convert them to the VAX sixteen bit word format. If the data contains ASCII data, or is a group of discretes packed 8 to a byte, then do not swap the bytes.

If the optional parameter MBX_SIZE is not specified then the default size MP$K_UNSOLICITED_MSGSIZE is used. Since this service uses the mailbox driver, a mailbox created by the MBP_REGISTER_UNSOLICITED_V with a mailbox smaller than MP$K_UNSOLICITED_MSGSIZE will result in a mailbox too small error. If the mailbox is larger than MP$K_UNSOLICIED_MSGSIZE a data overun error will occur. The MBP_REGISTER_UNSOLICITED call creates a single mailbox of the default size.

CONDITION VALUES RETURNED	MBP_NOTINITIALIZED	MBPPEX returned by MBP_OPEN_NET is corrupt or invalid.
	MBP_BADPATH	The specified path is less than or greater than the number of master paths allocated by the MBP_OPEN_NET call.
	MBP_PATHINUSE	A request to do a second read or write to the master path, but the path is already active with a previous I/O request.
	MBP_NOCHANNEL	There is no channel or path assigned to the specified path. The MBP_OPEN_NET most likely failed to allocate all of the desired paths. Check status of the MBP_OPEN_NET call.
	SYSTEM	Result of VMS SYS$QIO call
	SS$_MBTOOSML	The user has requested a buffer larger than the size of mailbox. The user buffer should not be larger than: MP$K_UNSOLICITED_MSGSIZE unless the caller has set up this in the MBP_REGISTER_UNSOLICITED_V call.
	SS$_BUFFEROVF	User buffer was not large enough to contain the complete message.

2.14 MBP_REGISTER_UNSOLICITED

Establish mailbox for a Master Modbus Plus nodes to write to.

FORMAT **MBP_REGISTER_UNSOLICITED (MBPPEX,MBX_NAME, MSG_CNT,PERM,MBX_CHANNEL,HOLDING_REGISTERS, NUMBER_HOLDING_REGISTERS)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below)

ARGUMENTS

MBPPEX
type: structure
access: write
mechanism: by reference

The structure to receive starting and ending address of the process specific data area used by the Modbus Plus interface subroutines (see MBPPEX in MBPlus.TLB).

MBX_NAME
type: character
access: read only
mechanism: by descriptor

Specifies the name of the mailbox to be created or a channel assigned to.

MSG_CNT
type: word
access: read only
mechanism: by value

This specifies the number of messages that should be allowed to be queued in the mailbox. This affects the buffer I/O quota for the process.

PERM
type: word
access: read only
mechanism: by value

This specifies if the mailbox should be established as a permanent mailbox. A permanent mailbox will allow messages to be retained over process activation, but not over system bootstraps.

MBX_CHANNEL
type: long word
access: write
mechanism: by reference

The actual channel number assigned to the mailbox is returned to the caller. This channel should be used for the user's or other user's to dequeue messages from using the standard VMS QIO system service calls.

HOLDING_REGISTERS
type: word array
access: read only
mechanism: by reference

Address of holding registers array. Used to service read holding registers from another Master on the Modbus Plus network.

NUMBER_HOLDING_REGISTERS
type: longword
access: read only
mechanism: by value

Number holding registers in holding register array. Each holding register is two bytes long.

DESCRIPTION

This subroutine allows the caller to set up a mailbox to receive slave messages from devices on the Modbus Plus network. The caller passes the name of a mailbox and if the mailbox should be permanent or not. If the mailbox is permanent, the MSG_CNT will not be used if the mailbox already exists on a subsequent activation of the process. The caller must have already allocated slave paths via the MPB_OPEN_NET call.

This subroutine will post another read on each of the slave paths and write the messages to the mailbox when the messages are received. In the event the mailbox becomes full, the software will respond to the remote device with the Modbus Plus slave response indicating a routing error response. The routing failure code will be set to indicate: "10= Slave rejected command".

If the mailbox is to be created as a permanent mailbox, the caller must have "SYSNAM" privilege. The reading process can be either the current process or any other process in the system. The **MBP_READ_UNSOLICITED entry may be called to actually read the message from the mailbox.**

The actual content of the letter contained in the mailbox is documented in the Modicon "DEC Host Based Devices User's Guide" under the "IO$_READ_SC" command. The complete message including the Modbus Plus header is placed into the mailbox. The actual format of the data portion of the message has to be defined with the programmer of the PLC sending Master.

PLCs or other Modbus Plus Master nodes on the Modbus Plus network can only read and write the host holding registers (40001 to 49999). This entry also supports the ability of the calling process to allow master reads from Master PLCs or other Modbus Plus Masters to request "HOST" holding registers. If the calling process

desires to support this functionality, then the address of a word array representing the host's holding registers must be passed. Only the slave paths allocated by this process will be able to respond with host register values. The host holding registers can be a global section that is mapped by the calling process. This routine probes the holding register array for read access.

The byte order and synchronization of data within the Host Holding Register region are the responsibility of the user. The VMS process that calls MBP_REGISTER_UNSOLICITED supports the actual data transfer between the Modbus Plus device driver at AST level. Therefore, disabling the AST recognition of this VMS process while the region is being modified will guarantee atomic and consistent data for Modbus Plus Master nodes reading data from the Host's holding registers. The method used must ensure that the process will be able to respond to the Master Modbus Plus node prior to the Master node timing out because no slave data response from the VMS host was received.

CONDITION VALUES RETURNED	MBP_NOSLAVE	No slave paths were allocated
	MBP_NOTINITIALIZED	MBPPEX structure not initialized
	MBP_SLVREADFAIL	Slave read failure
	MBP_SLVBUSY	Attempt to allocate second read on same slave path
	MBP_MBXCREFAIL	VMS SYS$CREMBX failed. Examine MBP$L_STATUS in the MBPPEX structure for the System Service status.
	MBP_MBXFAIL	Unable to create mailbox (see secondary status, will contain the VMS system service code)

2.15 MBP_REGISTER_UNSOLICITED_V

Establish mailboxes for Master Modbus Plus nodes to write to by slave path. Similar to MBP_REGISTER_UNSOLICITED but unique mailbox per path.

FORMAT **MBP_REGISTER_UNSOLICITED_V (MBPPEX,MBX_ARRAY)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: write
mechanism: by reference

The structure to receive starting and ending address of the process specific data area used by the Modbus Plus interface subroutines (see MBPPEX in MBPlus.TLB)

MBX_ARRAY
type: structure array
access: read/write
mechanism: by reference

This structure should be dimensioned by eight (number of modbus slave paths). An entry should be filled in for slave path allocated in the MBP_OPEN_NET call. The content of each array element is:

Mailbox name
type: character
access: readonly

This is a 1 to 32 character zero terminated name of the mailbox. If the caller does not have system name privilege, the name is ignored by the create mailbox system service.

Permanent_flag
type: byte
access: readonly

If non-zero then the mailbox should be set as a permanent mailbox.

Letter count
type: word
access: readonly

The number of messages that the mailbox should be created to contain.

Letter size
type: word
access: readonly

This contains the size of each message in the mailbox (default is: MP$K_UNSOLICITED_MSGSIZE). The minimum size must be larger than size of the modbus message header.

Mbx channel
type: long word
access: write

This is the returned channel number of the created or assigned mailbox.

DESCRIPTION

Each path is assigned to a unique mailbox. The caller passes the name of each mailbox and if the mailbox should be permanent or not. If the mailbox is permanent, the letter count and letter size will not be used if the mailbox already exists on a subsequent activation of the process. The caller must have already allocated slave paths via the MPB_OPEN_NET call.

This subroutine will post reads on each of the slave paths and write the messages to the mailbox when the messages are received. In the event the mailbox becomes full, the software will respond to the remote device with the Modbus Plus slave response indicating a routing error response. The routing failure code will be set to indicate: "10= Slave rejected command".

If the mailbox is to be created as a permanent mailbox, the caller must have "SYSNAM" privilege. The reading process can be either the current process or any other process in the system.

The actual content of the letter contained in the mailbox is documented in the Modicon “DEC Host Based Devices User's Guide” under the "IO$_READ_SC" command. The complete message including the Modbus Plus header is placed into the mailbox. The actual format of the data portion of the message has to be defined with the programmer of the PLC sending Master.

This subroutine does not support the ability of the remote PLCs to read from the VAX holding registers.

CONDITION VALUES RETURNED	MBP_NOSLAVE	No slave paths were allocated
	MBP_NOTINITIALIZED	MBPPEX structure not initialized

MBP_MBXFAIL	Unable to create mailbox (see secondary status, will contain the VMS system service code)
MBP_NOSLAVE	Caller did not allocate any slave paths
MBP_SLVREADFAIL	Slave read failure
MBP_SLVBUSY	Attempt to allocate second read on same slave path
MBP_MBXCREFAIL	VMS SYS$CREMBX failed. Examine MBP$L_STATUS in the MBPPEX structure for the System Service status.
MBP_MBXSIZE	Size of the mailbox is too small. Must be larger than modbus plus message header.

2.16 MBP_RESUME_UNSOLICITED

Restore the recognition of Modbus Plus slave messages of a Modbus Plus connected VMS process.

FORMAT **MBP_RESUME_UNSOLICITED (MBPPEX,PROCESS_NAME,DEVICE, CTRL_SLAVE_PATH)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: read
mechanism: by reference

The structure to receive starting and ending address of the process specific data area used by the Modbus Plus interface subroutines (see MBPPEX in MBPlus.TLB).

PROCESS_NAME
type: character (zero terminated)
access: readonly
mechanism: by reference

Process name of the target process.

DEVICE
type: character (zero terminated)
access: readonly
mechanism: by reference

Device name that slave path is allocated on (MPA1, MPB1). Note: The Modbus Plus devices are created as two device pairs. Specify the first device (for a single SQ85, devices MPA0: and MPA1: are created, specify MPA1).

CTRL_SLAVE_PATH
type: longword
access: readonly
mechanism: by value

SQ85 controller slave path number assigned to device by the target process to enable recognition of slave messages from.

DESCRIPTION

This subroutine allows the caller to restore the recognition of Modbus Plus slave messages of another process.

To use this entry, the calling process must be in the same group as the target process such that it may map the process expanded region of the target VMS process. The target VMS process is the process that has registered for unsolicited slave reads by calling one of:

MBP_REGISTER_UNSOLICITED
MPB_REGISTER_UNSOLICIED_V MBP_HOST_WRITEABLE_REGION_V

This entry maps with write access the group global region named by concatenating the device (less colon) and the process name with an underscore separating them. For example for a single SQ85 and a process named "SCAN" the region would be: "MPA1_SCAN". This routine simply clears the disable bit in the region for the slave path requested. The slave paths are the controller slave path numberes as allocated by the driver and referenced by the SQ85. This is the second parameter in a PLC MSTR route information in the programmable controller.

CONDITION VALUES RETURNED

MBP_NOSLAVE	No slave paths were allocated
MBP_MBXFAIL	Unable to create mailbox (see secondary status, will contain the VMS system service code)
MBP_NOSUCHPATH	The desired slave path was not allocated to the target process.
MBP_NOSLAVEMATCH	Controller slave path was assigned by process

2.17 MBP_SUSPEND_UNSOLICITED

Stop the recognition of Modbus Plus slave messages of a Modbus Plus connected VMS process.

FORMAT **MBP_SUSPEND_UNSOLICITED (MBPPEX,PROCESS_NAME,DEVICE,CTRL_SLAVE_PATH)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: read
mechanism: by reference

The structure to receive starting and ending address of the process specific data area used by the Modbus Plus interface subroutines (see MBPPEX in MBPlus.TLB).

PROCESS_NAME
type: character (zero terminated)
access: readonly
mechanism: by reference

Process name of the target process.

DEVICE
type: character (zero terminated)
access: readonly
mechanism: by reference

Device name that slave path is allocated on (MPA1, MPA3). Note: The Modbus Plus devices are created as two device pairs. Specify the first device (for a single SQ85, devices MPA0: and MPA1: are created, specify MPA1).

CTRL_SLAVE_PATH
type: longword
access: readonly
mechanism: by value

SQ85 controller slave path number assigned to device by the target process to disable recognition of slave messages from.

DESCRIPTION

This subroutine allows the caller to stop the recognition of Modbus Plus slave messages of another process. If mailboxes are being used, the content of the mailboxes are not effected.

To use this entry, the calling process must be in the same group as the target process such that the calling process may map the process expanded region of the target process. The target VMS process is the process that has registered for unsolicited slave reads by calling one of:

MBP_REGISTER_UNSOLICITED
MPB_REGISTER_UNSOLICIED_V
MBP_HOST_WRITEABLE_REGION_V

This entry maps with write access the group global region named by concatenating the device (less colon) and the process name with an underscore separating them. For example for a single SQ85 and a process named "SCAN" the region would be: "MPA1_SCAN". This routine simply sets the disable bit in the region for the slave path requested.

The slave paths are the controller slave path numberes as allocated by the driver and referenced by the SQ85. This is the second parameter in a PLC MSTR route information.

CONDITION VALUES RETURNED

MBP_NOSLAVE	No slave paths were allocated
MBP_MBXFAIL	Unable to create mailbox (see secondary status, will contain the VMS system service code)
MBP_NOSUCHPATH	The desired slave path was not allocated to the target process.
MBP_NOSLAVEMATCH	Controller slave path was assigned by process

2.18 MBP_WRITE_EXTENDED

Write extended memory files in a Modbus Plus slave node.

FORMAT **MBP_WRITE_EXTENDED[W](MBPPEX,PATH,ROUTE, START,COUNT,FILE,BUFFER,STATUS,EFN,FLAG[,MTMO])**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

The structure that was returned by the MBP_OPEN_NET call.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is to be directed to. The path must be between one and the number allocated by the MBP_OPEN_NET call.

ROUTE
type: five byte array
access: readonly
mechanism: by reference

This is an array filled in by the caller that specifies the route path to the particular PLC. The content of this array requires knowledge of the connected Modbus Plus network devices and their addresses.

START
type: long
access: readonly
mechanism: by value

Starting register in addressed PLC to beginning reading registers from. Valid range for each file is 60000 to 69999.

COUNT
type: word
access: readonly
mechanism: by value

The number of packed registers to read from the addressed PLC. If the number of registers exceeds what is able to read from a single Modbus Plus transfer, multiple Modbus Plus transfers will be done.

FILE
type: word
access: readonly
mechanism: by value

The extended memory of the PLC is broken up into 10000 registers each (and the remaining amount for the last segment), allowing registers in the range of 60000 to 69999 per file. The file number ranges from one (1) to the available memory assigned in the PLC for extended memory in ten thousand register increments.

BUFFER
type: array
access: readonly
mechanism: by reference

Caller array of registers to be written in the PLC.

STATUS
type: structure MBP_STATUS
access: write
mechanism: by reference

Final status.

EFN
type: long word
access: readonly
mechanism: by value

Event flag to set when I/O is complete.

FLAG
type: long word
access: readonly
mechanism: by value

If this FLAG is set to 1, then the data is not byte swapped before it is sent to the PLC.

MTMO
type: long word
access: read
mechanism: by value

Modbus Read Master Response timeout value in seconds. Must be greater than two.

DESCRIPTION

This subroutine writes registers in extended memory files in slave PLCs on the Modbus Plus network. The data is byte swapped unless the FLAG argument is set to one. This routine makes use of the Modbus function code 21. This routine is available in both the asynchronous and synchronous versions. By default, the bytes are byte swapped when before they are written to the PLC unless the FLAG argument is specified. The user call may translate into multiple I/Os to the selected Modbus Plus device. The event flag will be set when the all of I/O is complete or an error is encountered.

The starting register number for an extended memory file registers is one less than that for the other registers (the first register in an extended memory file is 60000, while the first holding register is 40001). Each extended memory file is partitioned to hold 10000 registers except for the last file which occupies whatever is left in the extended memory. Only a single extended memory file may be written per call.

CONDITION VALUES RETURNED

MBP_MBPPEXINVLD	MBPPEX structure is invalid
MBP_INVALIDARG	One of the call arguments is invalid, not readable, or not writable.
MBP_PATHINUSE	The specified path already has an I/O active.
MBP_BADPATH	Path specified not allocated or out of range
MBP_NOCHANNEL	No channel open on path
MBP_BADSTARTADDR	Invalid starting address
MBP_IONOTCOMP	I/O not complete on channel
MBP_NOTINITIALIZED	MBPPEX structure not initialized
MBP_INVALIDEFN	Invalid event flag specified
MPB_INVLPATH	Specified path is not between one and the number allocated by MBP_OPEN_NET

2.19 MBP_WRITE_GLOBAL_DATA

Write Modbus Plus 64 bytes of global data.

FORMAT **MBP_WRITE_GLOBAL_DATA (MBPPEX,PATH,BUFFER,EFN,STATUS)**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MBP_ error status codes.

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

Structure returned by MBP_OPEN_NET.

PATH
type: word
access: readonly
mechanism: by value

Path to use for the I/O.

BUFFER
type: array
access: read
mechanism: by reference

Contains the 64 byte buffer to be written to this controller's global data area.

EFN
type: long word
access: readonly
mechanism: by value

I/O event flag for synchronization.

STATUS
type: structure MBP_STATUS
access: write
mechanims: by reference

I/O completion status.

DESCRIPTION This subroutine will write the 64 bytes of global data. This routine operates synchronously as the data is local in the computer's controller. Only the local SQ85's global data can be written to. No byte swapping of data is performed.

CONDITION VALUES RETURNED	MBP_NOTINITIALIZED	MBPPEX returned by MBP_OPEN_NET is corrupt or invalid.
	MBP_BADPATH	The specified path is less than or greater than the number of master paths allocated by the MBP_OPEN_NET call.
	MBP_PATHINUSE	A request to do a second read or write to the master path, but the path is already active with a previous I/O request.
	MBP_NOCHANNEL	No channel open on path
	MBP_IONOTCOMP	I/O not complete on channel
	MBP_INVALIDEFN	Invalid event flag specified
	MBP_NOCHANNEL	There is no channel or path assigned to the specified path. The MBP_OPEN_NET most likely failed to allocate all of the desired paths. Check status of the MBP_OPEN_NET call.

2.20 MBP_WRITE_REGISTERS

Write registers to a slave Modbus Plus node.

FORMAT **MBP_WRITE_REGISTERS[W](MBPPEX,PATH,ROUTE, START,COUNT,BUFFER,STATUS,EFN,FLAG[,MTMO])**

RETURNS

VMS usage: COND_VALUE
type: longword
mechanism: by value

Longword status as defined by either a system service call or the MPB_xxxx status codes (see error codes listed below).

ARGUMENTS

MBPPEX
type: structure
access: readonly
mechanism: by reference

The structure that was returned by the MBP_OPEN_NET call.

PATH
type: word
access: readonly
mechanism: by value

Path that this I/O is directed to. The path must be between one and the number allocated by the MBP_OPEN_NET call.

ROUTE
type: five byte array
access: readonly
mechanism: by reference

The is an array filled in by the caller that specifies the route path to the particular PLC. The content of this array requires knowledge of the connected Modbus Plus network devices and their addresses.

START
type: long
access: readonly
mechanism: by value

Starting register in addressed PLC to beginning writing user registers to. The START register is used to determine if a FORCE of the coils, or a PRESET of the registers will be done. Only coils (00001 to 09999) and holding registers (40001 to 49999) may be written to.

COUNT
type: word
access: readonly
mechanism: by value

The number of packed registers to write to the addressed PLC. If the number of registers exceeds what is able to written in a single Modbus Plus transfer, multiple Modbus Plus transfers will be done.

BUFFER
type: array
access: read
mechanism: by reference

Caller array to acquire the registers for the PLC.

STATUS
type: structure MBP_STATUS
access: write
mechanism: by reference

Final status

EFN
type: long word
access: readonly
mechanism: by value

Event flag to set when I/O is complete.

FLAG
type: long word
access: readonly
mechanism: by value

If FLAG is set to 1, then the data is not byte swapped before it is written to the PLC.

MTMO
type: long word
access: read
mechanism: by value

Modbus Read Master Response timeout value in seconds. Must be greater than two.

DESCRIPTION

This subroutine writes the user buffer to the registers within the PLC addressed by the specified route array.

The event flag is used for I/O synchronization and if none is specified, event flag zero is used. The event flag will be set when all of the registers specified have been written. To satisfy the request, this routine may do more than a single Modbus

transfer may. This routine uses ASTs to process all I/O completions, and therefore must have adequate AST quota.

The START register number is used to determine which type of output is being performed. The source buffer should be sized for two bytes per holding register, or eight registers per byte for coils.

For register transfers, the VAX sixteen bit word is byte swapped to match the format of the PLC sixteen bit registers unless the "FLAG" argument is set to one.

Both the synchronous and asynchronous version is available.

CONDITION VALUES RETURNED	MBP_MBPPEXINVLD	MBPPEX structure is invalid
	MBP_INVALIDEFN	Invalid event flag specified
	MBP_NOCHANNEL	No channel open on path
	MBP_IONOTCOMP	I/O not complete on channel
	MBP_BADPATH	Path specified not allocated or out of range
	MBP_NOTINITIALIZED	MBPPEX structure not initialized
	MBP_INVALIDARG	One of the call arguments is invalid, not readable, or not writable.
	MBP_PATHINUSE	The specified path already has an I/O active.
	MBP_BADSTARTADDR	Invalid starting address
	MPB_INVLPATH	Specified path is not between one and the number allocated by MBP_OPEN_NET

Chapter 3
Utilities

3.1 Introduction

The following utilities will help the VAX programmer and system manager tune, debug, and provide help in fault analysis. Some of the utilties are provided by with the device driver software, and some are provided with this software.

3.2 Network Diagnostic Utility

The "NDU" utility provides a general purpose program that allows the VAX programmer and system manager to diagnose Modbus Plus health, and operation. It is further documented in the Modicon "DEC Host Based Devices User's Guide".

3.3 Monitor Modbus Plus Process

Any program that is using the Modbus Plus application library also exposes a part of its process as a group global section. The group global section contains statistics and status relative to the execution of the communication routines and the master and slave paths. This process displays information from this region for any process that is currently running to the users terminal. The process to be examined must be in the same group as the user as the section is a Group global section and not a System global section.

DSPMBP is invoked with the following DCL command. The global section name must be specified with the /NAME switch. This must be defined as a foreign DCL command ($DSPMBP == "$MBPLUS_:DSPMBP.EXE").

$DSPMBP /NAME=global_section_name [/FULL] [/CLEAR]

NAME- The user must specify the name of the global section that is to be displayed. The global section name consists of the device name being used to communicate with the Modbus Plus network (ie: MPA1, MPB1...) plus the process name. The global region is created when the indicated process called MBP_OPEN_NET. If the desired process was "SCANNER" and it specified the SQ85 or SA85 device: "MPA1" then the global section name would be: "MPA1_SCANNER".

FULL- Forces the entire contents of the global section to be displayed. The default is to display only selected fields.

CLEAR- Forces specific counters and timers to be reset.

The following are example displays captured from an operational system:

$DSPMBP/NAME=MPA1_MONITOR_SCANNER

```
                   # Paths       MBX       Last        Last      R/W
Start - End Address MST SLV  EFN Chan       Read       Write     Status    Process Name...
001CD200-001CD9FF    01  01  000       4 11:29:19  4 11:29:19    00000000  MONITOR_SCANNER


  * * * * MASTER PATH * * * *        1                          * * * * SLAVE PATH * * * *        1
  Modbus+ I/O status:                                           Modbus+ I/O status:
          Condition code    :          1                                  Condition code    :          1
          Transfer          :         20                                  Transfer          :          0
          Crash code        :          0                                  Crash code        :          0
          Path number       :          2                                  Path number       :         66
   Last read / write :       0.020000 /     0.040000            Last read           :       0.000000
   Read / Write counter :    10390 /  10374                     Last write          :       0.000000
   Start of read  :    4-MAY-1994 11:29:19.28                   Write / Read counter:       0 /      0
   Start of write :    4-MAY-1994 11:29:19.30                   Start of read       :
   Avg time master read   :       0.027727                      Start of write      :
   Avg time master writes :       0.037956                      Avg time slave reads  :
                                                                Avg time slave writes :
```

From the display, the following can be determined:

- The process has allocated master path number two.
- The last condition code was successful.
- The size of the last transfer was twenty bytes.
- The time of the Modbus Plus read or writes
- The average time for a master read or write to complete (total time for the master write plus the response from the addressed slave)
- The time for the last master read or write
- How many master writes and reads have been done by the process.

Appendix A
Header Files

A.1MBPPEX Structure

The following structures are provided in the include library:

1) slave.h- Definition of the message placed in the unsolicited message mailbox.

2) mbp_errors.h- Definition of all MBP error codes (this does not include those from the Modbus Plus communication device driver).

3) mbp_status.h- Standard status block returned from most calls that require I/O to the Modbus Plus network.

Appendix B
MBP Error Codes

The MBP software has the errors listed below along with the recovery procedures. The user should also consult the Modbus Plus Device Drivers Manual for any Modbus errors. Additional errors can be returned by the VAX/VMS system service calls. The MBPPEX structure has a secondary error status word that should also be consulted.

Any errors that cannot be returned to the caller, such as the errors that occur during the response to a PLC master are logged to the system operator console via the SYS$SNDOPR service.

B.1 MBP Error Codes

MBP_ASSIGNFAIL-Unable to assign device

The MBP_OPEN_NET was unable to assign the Modbus Plus communication device. Most probable causes are that the driver is not loaded, or the user is attaching to the wrong device. Examine the mbp$l_status in the MBPPEX structure.

MBP_BADPATH-Path specified not allocated or out of range

A call to one of the subroutines showed in path number that was not within range. The path number must be between one and the number that was allocated by the MBP_OPEN_NET call.

MBP_BADSTARTADDR-Invalid starting address

User has passed an invalid starting register or coil number, or has passed the value incorrectly.

MBP_EXCEPTION-Modbus exception response, see byte count

The Modbus Slave has responded with an exception. The byte count returned to the caller contains the execption. Consult the Modicon "DEC Host Based Devices User's Guide" for exception codes. At the time of this writing, the following were documented:

Exception Code	Error Condition
01	Illegal function for the addressed slave
02	Illegal data address within the information field ofr the addressed slave
03	Illegal data value in the information field for the addressed slave. Also, if the master requests a QIO call with too large a buffer.
06	Busy- the function just requested cannot be performed at this time because a long duration program command is beind processed.

MBP_INVALIDARG-Invalid argument

One of the user passed arguments is invalid.

MBP_INVALIDEFN-Invalid event flag specified

An event flag has been specified that is not valid. Consult the VMS system service manual for valid event flag useage.

MBP_INVALIDEFN-Invalid event flag specified

The event passed is an invalid VMS event flag. Consult the VMS system service manual for the correct usage of event flags.

MBP_IONOTCOMP-I/O not complete on channel

User has attempted to requested another function on a path that has not completed the prior request.

MBP_MAPFAIL-Unable to create process region

The MBP_OPEN_NET failed to create the group global section. The most probable error is insufficient privilege. Examine the mbp$l_status in the MBPPEX structure.

MBP_MBXCREFAIL-Unable to create slave data mailbox

The register unsolicited was unable to create the mailbox. Examine the mbp$l_status in the MBPPEX structure for the reason. Most probable are insufficient privilege, or insufficient buffer quota.

MBP_MBXSIZE-Specified mailbox is too small

The specified mailbox size for slave reads (PLC master writes) is too small. It must be at least as large as the Modbus Plus header size.

MBP_MBXWRITFAIL<MBX failed, ios: !XL, systs: !XL, size: !XL

Failed to write Modbus slave message to the unsolicited mailbox. Examine the status and determine recovery.

MBP_NOCHANNEL-No channel open on path

The user has passed a path that has no channel assigned to the MPA1 device. The most probable cause is that not all of the user's requested paths were allocated by the MBP_OPEN_NET call.

MBP_NOSLAVE-Caller did not allocate any slave paths

The user has requested to provide slave service, but no slave paths were allocated by the MBP_OPEN_NET call. Either none were available, or the MBP_OPEN_NET call is in error.

MBP_NOSLAVEMATCH-Controller slave path was not assigned

The current process has not set up the VAX for unsolicited reads for the indicated path. The path number may be in error, or the call has not correctly registered for slave messags.

MBP_NOTINITIALIZED-MBPPEX structure not initialized

The user has not called MBP_OPEN_NET, the MBPPEX structure is corrupt, or the MBPPEX was passed incorrectly.

MBP_PATHFAIL-Unable to allocate a path

The MBP_OPEN_NET was unable to open the number of paths as requested by the caller. Examine the mbp$l_status in the MBPPEX structure.

MBP_PATHINUSE-Path is in use

User has attempted to requested another function on a path that has not completed the prior request.

MBP_SLAVEFAIL-Slave read I/O failure: !/!XL, !XL, master: !XB>

This error is logged on the system console. It is generated by the AST services for the unsolicited messages. Examine the I/O status displayed and consult the Modicon DEC host Based Devices User's Guide.

MBP_SLVASSIGNED- Slave path already assigned for unsolicited messages

The caller has tried to register for unsolcited reads (master writes from a PLC) a second time.

MBP_SLVBUSY-Attempt to allocate second read on same slave path

A second MBP_REGISTER_UNSOLICITED was attempted to the same controller.

MBP_SLVREADFAIL-Slave read failure

The register unsolicited attempted to post a read on the slave paths but failed. Examine mbp$l_status for system service or Modbus Plus device error codes.

MBP_SLVREADQIO-Error posting IO$READ_SR status:!/ !XL

This error is logged on the system console. It is generated when the HOST is attempting to post a read on one of the slave data paths. Examine the I/O status displayed and consult the Modicon DEC Host Based Devices User's Guide.

MBP_SLVRSPFAIL-Slave response I/O failure:!/ !XL, !XL

This error is logged on the system console. It is generated when the VAX host attempts to acknowledge the write of the holding registers to the VAX. Examine the I/O status displayed and consult the Modicon DEC Host Based Devices User's Guide.

MBP_SLVRSPQIO-Error posting IO$WRITE_SR status:!/ !XL, !SL bytes to: !XB

This error is logged on the system console. It is generated when the VAX host attempts to a Slave Write to the PLC master. Examine the I/O status displayed and consult the Modicon DEC host Based Devices User's Guide.

MBP_WRNGNMBRPATHS-Wrong number of paths

The caller specified an incorrect number of paths.

B.2 Routing Errors

The following errors are a result of a Modbus Plus routing error. The routing failure code was translated from the code listed in Modicon "DEC Host Based Device User's Guide" to one of the following symbolic codes. Additional information in the buffer (at time of this writing) is:

Byte Number	Contents
4	Contains MAC function code (Hex 13)
8	Routing Failure index
15	Device type of the failed node (see Documentation)
16	Failure code translated to one of the following MBP error codes.

MBP_BADDESTADDR-Bad destination address
MBP_BRIDGEBUSY-Bridge Plus Paths busy
MBP_EXCEPTIONRESPONSE-Exception response received
MBP_FORGOTTEN-Forgotten transaction
MBP_INVALIDPATH-Invalid path
MBP_INVALIDROUTE-Invalid route
MBP_NODEOFFLINE-Node is off-line
MBP_NORRESPONSE-No Response received from Modbus node
MBP_PROGACCESSDENIED-Program access denied
MBP_SLAVEDOWN-Slave device is down
MBP_SLAVEREJECT-Slave rejected message
MBP_UNKRTRERR-Unknown routing error
MBP_UNSUPPORTED-Unsupported MAC function

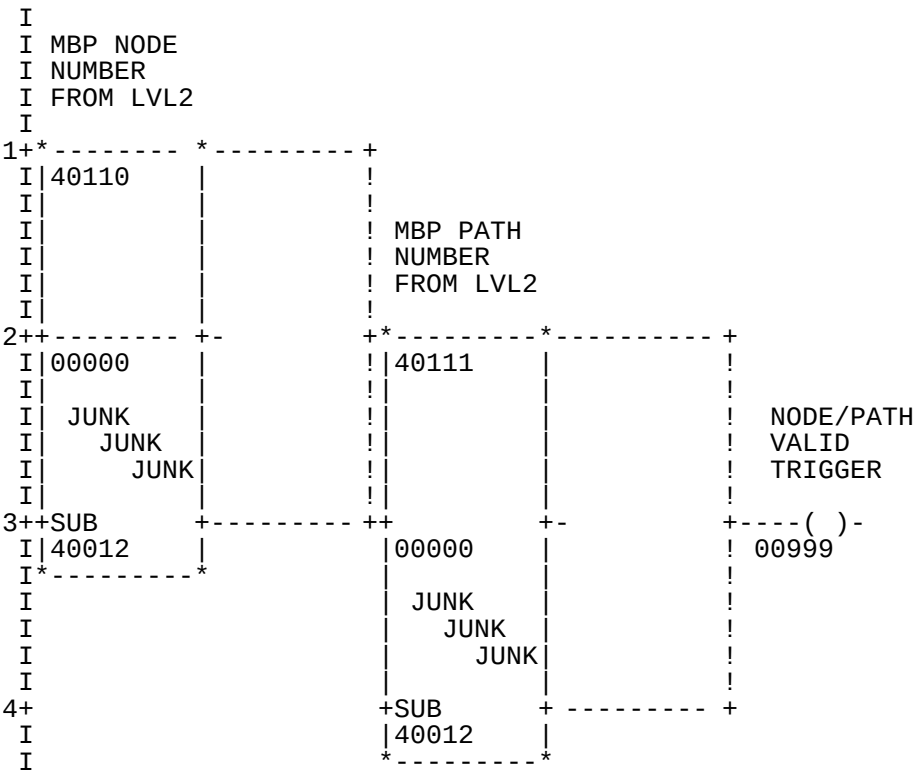
Appendix C
Sample PLC MSTR

C.1MSTR Example

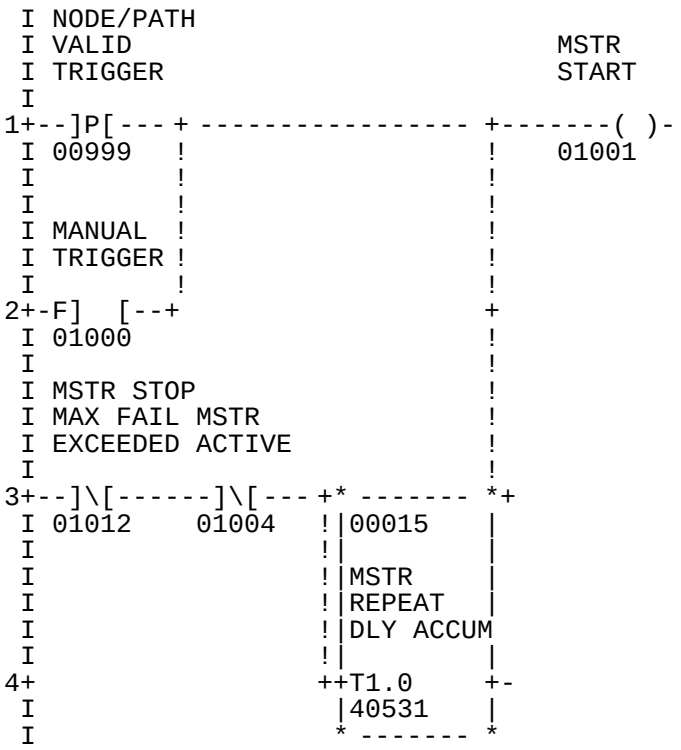
The following ladder diagram shows the use of the MSTR block within a PLC.

NETWORK 0005 Segment: 01

This logic determines whether the NODE and PATH ultimately provided by the Level 2 system is valid. This logic will enable or disable any MSTR instruction that generated the unsolicited messages to the Level 2 system, based on the availability of the Level 2 system to provide the NODE and PATH. Currently, the NODE and PATH are merely hard-coded into these registers for test purposes. The Level 1 system would probably clear these registers once per scan and recopy the values from the Level 2 write area. If the registers remain cleared, the Level 2 system is no longer available.

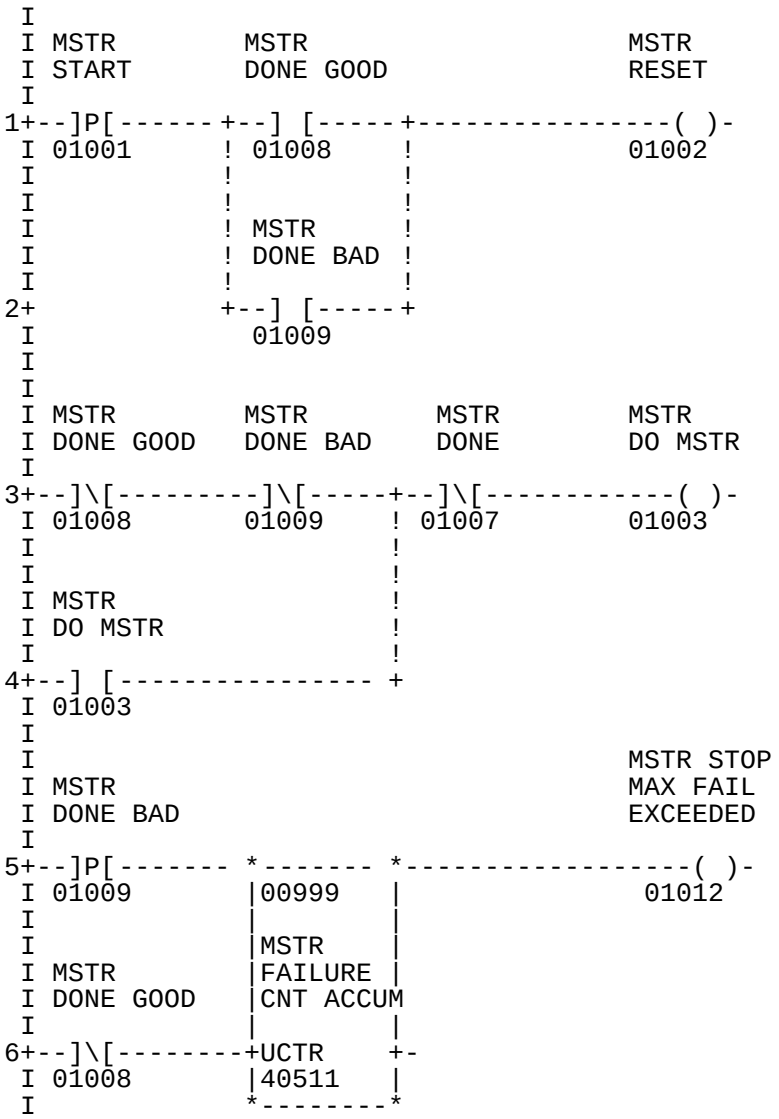


NETWORK 0006 Segment: 01 These are the triggers for the MSTR instruction execution. The instruction can be triggered by a valid NODE and PATH initialization (see comment for network 5), manually with a contact from a forced coil, or repetitively with a timer providing delay between triggers. Additionally, the repeat function may be deactivated if a preset number of successive failures occur.



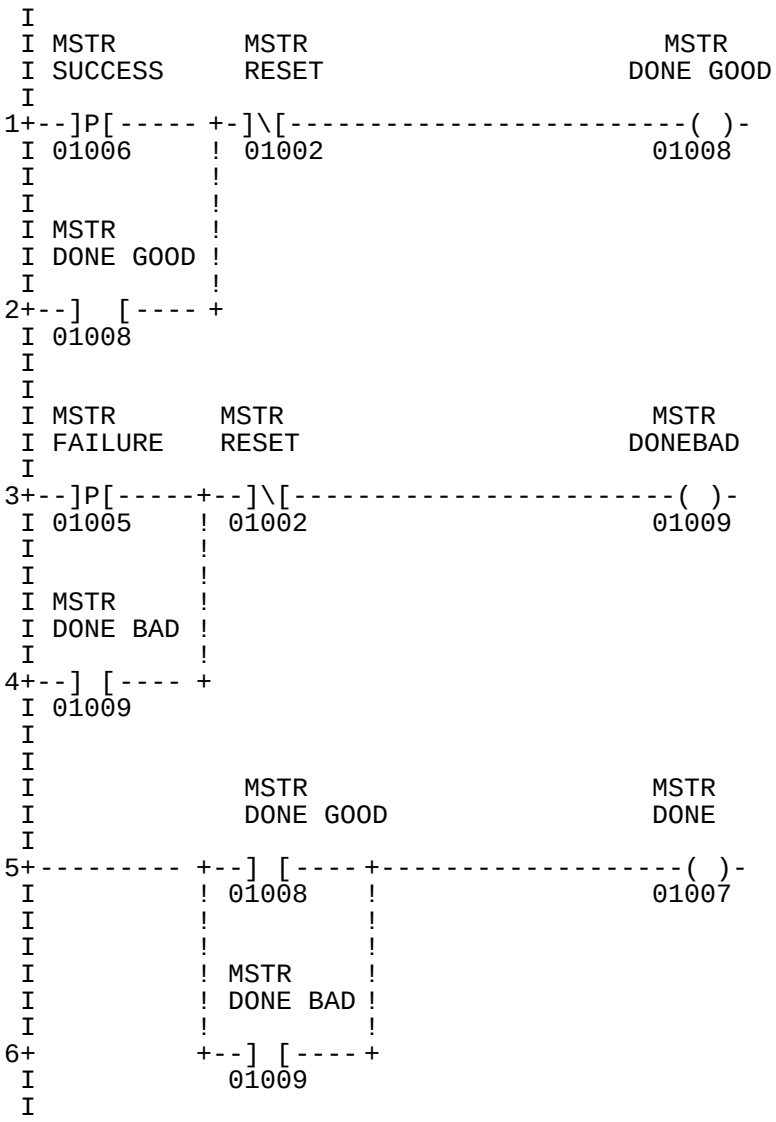
NETWORK 0007 Segment: 01

Whenever the MSTR instruction is to be executed (trigger occurs), the success or failure result of the last execution must be cleared (reset). After the result of the last MSTR instruction execution is cleared, the MSTR instruction execution is enabled by (DO MSTR) until it is completed (DONE). When an MSTR instruction execution terminates unsuccessfully, a counter is incremented, when termination is successful, the counter is reset. If the counter accumulator value reaches the preset value (999), the MSTR instruction is disabled via STOP MAX FAIL EXCEEDED.

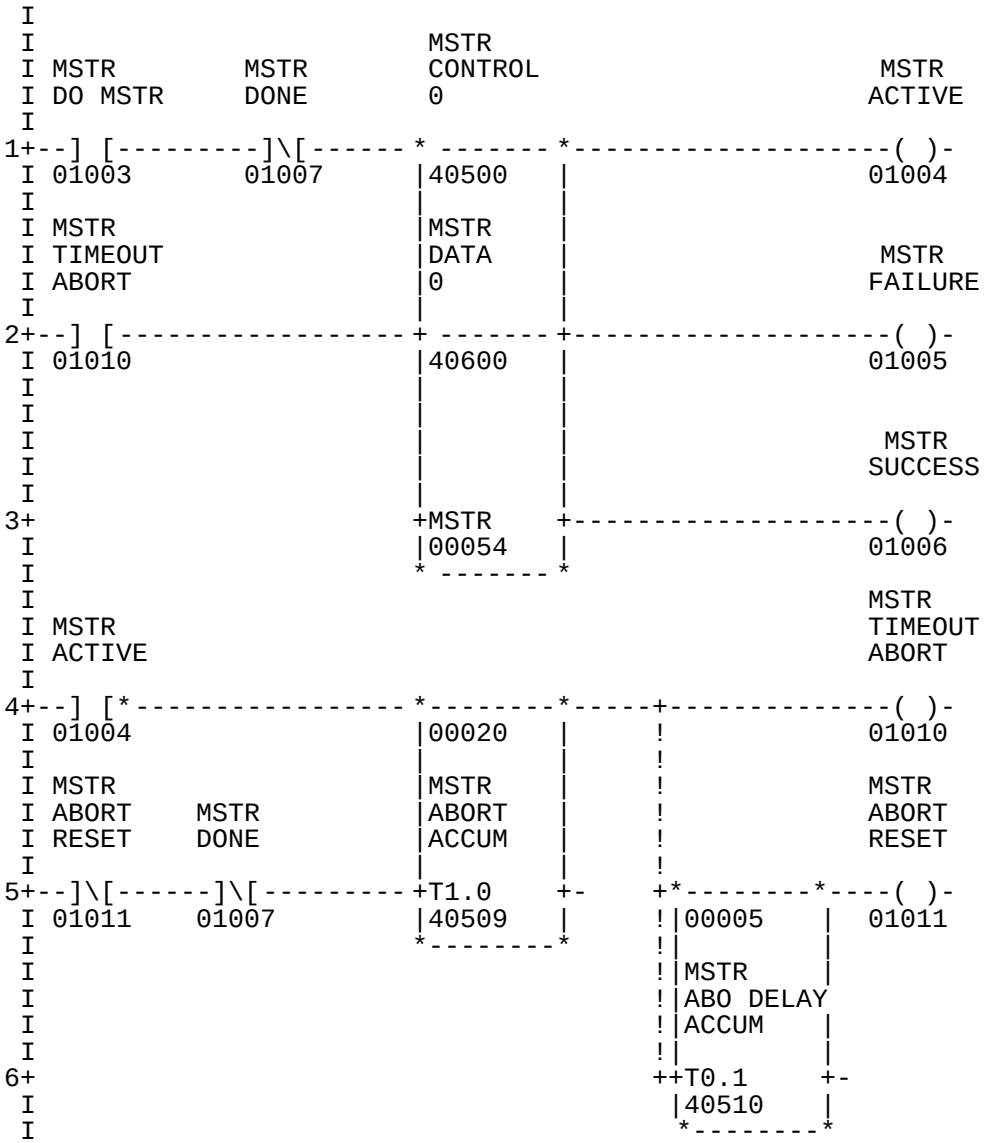


NETWORK 0008 Segment: 01

When an MSTR instruction execution terminates successfully, DONE GOOD is set (and sealed-in). When an MSTR instruction execution terminated unsuccessfully, DONE BAD is set (and sealed-in). When the MSTR instruction terminates, DONE is set.



NETWORK 0009 Segment: 01 The MSTR instruction is allowed to execute when DO MSTR is set, until DONE is set. DONE is set when the MSTR instruction terminates (see the comment for network 8). When the MSTR instruction is executing, ACTIVE is set. The MSTR control block registers start at 40500 CONTROL 0, and the data registers begin at 40600 DATA 0. The data area is 54 registers in length. The MSTR instruction execution may be aborted by TIMEOUT ABORT, which occurs when an MSTR instruction execution does not complete within 20 seconds. (TIMEOUT ABORT is reset 0.5 seconds after it occurs.)



Appendix D
Example VMSINSTALL

D.1VMSINSTALL Example

The following assumes all defaults when prompted. The SQ85 should be installed in the microvax with the CSR address and VECTOR as entered below. Users unfamilar with VMSINSTAL should consult the VMS System Manager documentation set. A directory listing of the product directory is shown after the VMSINSTAL had completed.

IPACT:: MicroVAX II running VMS V5.4-2

Username: SYSTEM
Password:

Welcome to VAX/VMS V5.4-2

$ set def sys$update $ @vmsinstal VAX/VMS Software Product Installation Procedure V5.4-2 It is 15-OCT-1992 at 08:06. Enter a question mark (?) at any time for help. %VMSINSTAL-W-ACTIVE, The following processes are still active:
LAKIA
* Do you want to continue anyway [NO]? y
* Are you satisfied with the backup of your system disk [YES]? y
* Where will the distribution volumes be mounted: MUA0:

Enter the products to be processed from the first distribution volume set.
* Products: mbp012
* Enter installation options you wish to use (none):
The following products will be processed:
MBP V1.2
Beginning installation of MBP V1.2 at 08:07

%VMSINSTAL-I-RESTORE, Restoring product save set A...

Modbus Application Interface Library and
Software Installation By: IPACT Inc.

Modbus Communications Device Driver By: Modicon

The distribution files have been restored from the saveset. The installation procedure is continuing.

Your system or common device/directory: IPACT$DUA0:[SYS0.SYSCOMMON.] This product creates the directory:

IPACT$DUA0:[SYS0.SYSCOMMON.][MBP012]

There appears to be an old version of the software installed in your system.
There is an old version currently installed in your the system. This will prevent us from doing the Installation Verification Procedure. You may choose abort this installation and remove the old version and unload the driver if it is loaded.

* Continue with installation [Y]? y
* How many SQ85 devices [1]:
* Enter CSR for controller (in OCTAL): 0 [766770]: 766000
* Enter VECTOR for controller (in OCTAL): 0 [310]: 320
SQ85 0 CSR: 766000 VECTOR: 320
* Are these correct [Y]? y

MBP_I_COFFEEBREAK answer section complete

MBP_I_LINKING linking shared image MBP_SHARE
MBP_I_LINKING linking SQ85 device driver MPDRIVER
MBP_I_COMMAND Building startup command file MBP_STARTUP.COM MBP_I_DIRECTORY Creating IPACT$DUA0:[SYS0.SYSCOMMON.][MBP012] %VMSINSTAL-I-SYSDIR, This product creates system disk directory IPACT$DUA0:[SYS0.SYSCOMMON.][MBP012].
%VMSINSTAL-I-SYSDIR, This product creates system disk directory IPACT$DUA0:[SYS0.SYSCOMMON.][MBP012.EXAMPLES].
MBP_I_MOVING Marking files to move
MBP_I_COMPLETE KITINSTAL.COM complete

This kit is supplied with an IVP that is part of the VMSINSTAL kit. If you choose to execute the IVP, then the driver and shared image will be installed. If you have an old version on your system, you should not run the IVP.

Additonally if you suspect that the SQ85 may not be installed correctly, or you have never tried it before you should not execute the IVP as a system crash may result.

You may at a convient time (when you can tolerate a system crash), execute the MBP_STARTUP.COM command file that was created, and then run the utilties CK_PROG, and then NDU (see documentation).

The product directory also contains the files MBPLUS.COM and MBPLUS.RNO which will allow you to create additional copies of the documentation.

* Execute IVP [Y]? n

Your installation is now complete. After the files are moved, we will test your installation if you requested it.

%VMSINSTAL-I-MOVEFILES, Files will now be moved to their target directories...
Installation of MBP V1.2 completed at 08:18 Enter the products to be processed from the next distribution volume set.
* Products:
VMSINSTAL procedure done at 08:18

Directory SYS$COMMON:[MBP012]

CK_PROG.EXE;1
DSPMBP.EXE;1
EXAMPLES.DIR;1
MBP.OLB;1
MBPLUS.COM;1
MBPLUS.RNO;1
MBP_C.TLB;1
MBP_DBG.OLB;1
MBP_FOR.TLB;1
MBP_SHARE.EXE;1
MBP_STARTUP.COM;1
MPDRIVER.EXE;1
NDU.EXE;1

Total of 13 files.

Directory SYS$COMMON:[MBP012.EXAMPLES]

BITSTR.FOR;1
BUILD_ALL.COM;1
DSPMBP.COM;1
DSPMBP.FOR;1
EXAMPLE.C;1
FORCE_SINGLE_COIL_EX.COM;1
FORCE_SINGLE_COIL_EX.EXE;1
FORCE_SINGLE_COIL_EX.FOR;1
GET_DRIVER_STATISTICS_EX.COM;1
GET_DRIVER_STATISTICS_EX.EXE;1
GET_DRIVER_STATISTICS_EX.FOR;1
GET_NETWORK_STATISTICS_EX.COM;1
GET_NETWORK_STATISTICS_EX.EXE;1
GET_NETWORK_STATISTICS_EX.FOR;1
MENU.COM;1
PRESET_SINGLE_REGISTER_EX.COM;1
PRESET_SINGLE_REGISTER_EX.EXE;1
PRESET_SINGLE_REGISTER_EX.FOR;1
READ_WRITE_GLOBAL_DATA_EX.COM;1
READ_WRITE_GLOBAL_DATA_EX.EXE;1
READ_WRITE_GLOBAL_DATA_EX.FOR;1
READ_WRITE_REGISTERS_EX.COM;1
READ_WRITE_REGISTERS_EX.EXE;1
READ_WRITE_REGISTERS_EX.FOR;1
REGISTER_UNSOLICITED_EX.COM;1
REGISTER_UNSOLICITED_EX.EXE;1
REGISTER_UNSOLICITED_EX.FOR;1
RW_EXTENDED_EX.COM;1
RW_EXTENDED_EX.EXE;1
RW_EXTENDED_EX.FOR;1
SCAN.COM;1
SCAN.FOR;1
SMG_DISPLAY.FOR;1
SYS_ERROR.FOR;1

Total of 34 files.

INDEX