MMS 77422 Protocols

December 19, 2018

- DRAFT -

History

At the time that the MMS 77422, and “MAGnet” or “MMSnet”, was developed, “Ethernet” was in it’s infancy. There was a lot of debate about the problematic nature of Ethernet's “wanton collision” scheme of arbitration. In addition, Ethernet was very expensive since it required special transceivers and specialized coaxial cable. Since that time, Ethernet has added collision detection and avoidance, boosted speeds, and introduced (the now ubiquitous) twisted pair wiring, making Ethernet far more economical. However, at that moment in the early 1980’s, the future of Ethernet appeared, to many, to be rather dim.

One proposed solution to the woes of Ethernet was “token passing”, where a node could only send (originate) messages if it “owns” the token. In such a scheme, a node stores up messages to be sent until it is given the token, then sends one or more messages, and then passes the token along to another node. This method of “collision avoidance” was chosen by Magnolia Microsystems for their network product. Also, an inexpensive physical network medium was desired, so RS-422 (differential pair, similar to modern twist-pair Ethernet) transceivers were chosen, and simple twisted-pair cabling used. The resulting network was a physical bus topology, where all nodes are directly connected only once, but the logical topology was token passing “ring”. This differs from “token ring” networks, where each node had to be daisy chained in a fully-connected loop.

In addition, due to the lack of broad customer interest in networking at the time, the board was made “dual use”, with a non-network application as a coprocessor for the Z89. This was a way to expand the 2MHz Z89 to run programs at 4MHZ, and with a much larger CP/M TPA than was possible while running on the native Z89 CPU. In this mode, the customer’s CP/M applications actually ran on the 77422, and the Z89 was an I/O slave processor, used to communicate with the outside world. The CP/M on the 77422 ran a very skinny BDOS/BIOS that did nothing more than “function ship” operations to the Z89.

Host (Z89) Interface

The host interface consists of 2 input ports and 4 output ports, based at either 0x7c or 0x78 depending on JP8 and JP9 and the chosen Z89 I/O slot.

The base port is the data port. Input will read the AM2950 parallel port “A” side (/OE|AS, S-register), output will write the “A” side (CPR, R-register). The data ports are driven by the 77422 DMA controller, however it is still required for the Z89 to check status before reading/writing a byte.

Input base+1 reads various status bits:

bit 0:	INT-422 status
bit 1:	INT-Z89 status (used to determine if shared INT is from 77422)
bit 2:	Output Buffer Empty (ready for OUT)
bit 3:	Input Buffer Full (ready for IN)

Output base+1 clears INT-Z89 status.

Output base+2 triggers NMI to the 77422 Z80.

Output base+3 triggers INT-422 to the 77422 Z80, IM2 vector 0xff.

INT-Z89 is triggered at the end of a 77422-to-Z89 transfer.

The typical sequence to RESET the 77422 and confirm presence is to issue an NMI, then trigger an INT-422, and poll the INT-422 status bit to see it cleared. This sequence is used, for example, to prepare the 77422 for coprocessor mode, as well as to exit from coprocessor mode (and resume network operation).

Host (Z89) Protocol – Network Mode

All host-77422 communications uses a 7-byte header followed by 0 or more bytes of payload. The header consists of one control (command) byte and three 16-bit integers (sometimes interpreted as single bytes). 16-bit values are always little-endian. The payload format depends on the command.

In general, all command codes less than 0x80 will be sent directly on the network (with a few exceptions). Bytes 1,2 of the header contain the payload length, and byte 4 contains the destination node. An acknowledgment response will be returned with the same command but bit 0 set to “1”.

Command Type 0: CP/NET messages		
Byte	**Value**	**Notes**
0	00H/01H	CP/NET request/response
1,2	length	Number of bytes in the payload
3	unused	
4	dest	Destination node ID
5,6	unused	
7..	payload	CP/NET message (FMT 0/1)

Command Type 1: Load/Execute code		
Byte	**Value**	**Notes**
0	11H-15H	
1,2	length	Length of code, may be 0 for 15H
3,4	unused	
5,6	addr	Address to load/execute
7...	code	Code to load into memory at ‘addr’

A successful boot request will result in one or more of these arriving at the host. Code 11H is to load and execute (jump to ‘addr’), 13H is for load only, and 15H is for execute only.

Command Type 2: Boot from server		
Byte	**Value**	**Notes**
0	20H	
1,2	0001H	
3	unused	
4	dest	Destination node ID
5,6	unused	
7	device	Device type, for selection of boot code.

A boot request is sent to the server ‘dest’. The server normally examines the requestor’s node ID and and device type and chooses a boot image, which is sent via type 1 Load/Execute Code messages. A special response type 28H indicates that booting from this server is not allowed/possible.

Device type 0 indicates a 77422 diskless workstation, type 1 indicates a Z89.

Command Type 3: Get Network Status (not transmitted on network)		
Byte	**Value**	**Notes**
0	30H	
1,2	0000H	
3	unused	
4	type	Declare node type (1-15), no change if “0”
5,6	unused	

This command is handled locally, no network traffic required. Returns the network table, as it existed during the most recent TOKEN ownership. If byte 4 is not zero, the low 4 bits will be used as the node type that is advertised on the network.

Response Type 3: Network Status Return (not transmitted on network)		
Byte	**Value**	**Notes**
0	30H	
1,2	0041H	Payload length, 65 bytes
3	node	Node ID of this 77422
4	status	Bit 7 “1” if we’re online (there are other nodes)
5,6	unused	
7..67	net tbl	Network table, shared between all active nodes.

The network table contains information about all discovered nodes and the timing of polling or error drop-out of nodes. See “Type 13 messages” under Network Protocol for more information.

Response Type 3: Result of Send (not transmitted on network)		
Byte	**Value**	**Notes**
0	38H	
1,2	0000H	
3	error	Result of send
4	unused	
5,6	unused	

Every message from the host that is sent over the network will have a result frame sent back to the host immediately after sending, regardless of any response message that might come later. Result codes are: 0=ACK, 1=timeout, 2=NAK, 3=CRC, 4=invalid, 5=protocol. Note, this frame is not a network message, but is generated by the local 77422 to indicate the results of the send.

Command Type 6: Load/Execute Code (not transmitted on network)		
Byte	**Value**	**Notes**
0	60H/61H	Load+goto/Load-only
1,2	length	Number of bytes in the payload
3,4	unused	
5,6	addr	77422 Z80 Address to place code, also execution address unless 61H
7..	payload	Code image

If command is 61H, code is loaded into memory but not executed.

Response Type 7: Special ROM functions (not transmitted on network)		
Byte	**Value**	**Notes**
0	70H	Low digit selects function, 0-15
1,2	0000H	(no payload)
3,4	BC	Value to pass in BC
5,6	DE	Value to pass in DE

The only implemented function is 15: enter the debug monitor mode. All others result in a restart of the ROM. No response frame is defined, but future functions may implement a response.

Host (Z89) Protocol – Coprocessor Mode

Corprocessor mode is activated by loading new code into the 77422 and executing it. The standard code, contained in CPM422.COM, should operate on any CP/M running on the Z89 – or at least any CP/M 2.2 (CP/M 3 should work, as it is backward compatible with CP/M 2.2, but was not tested as far as I recall). CPM422.COM does not depend on any MMS CP/M features, or CP/M 3 features.

When running coprocessor mode, the normal 77422 networking functions are disabled. The protocol between the Z89 and the 77422 becomes a pure BDOS/BIOS function shipping protocol. The same 7-byte header is used, but the command byte is the BDOS function code (with some extensions) and the three 16-bit integers represent BC, DE, and HL, respectively. For BIOS calls, the command byte sent to the Z89 contains the BIOS entry code (0-15) ORed with 0xf0. The following CP/M 2.2 BIOS entry codes are used:

Code	**Entry Mnemonic**
0	WBOOT
1	CONST
2	CONIN
3	CONOUT
4	LSTOUT
5	PUNOUT
6	RDRIN
7	HOME
8	SELDSK
9	SETTRK
10	SETSEC
11	(unused)

Code	Entry Mnemonic
12	READ
13	WRITE
14	LSTST
15	SECTRN

Note that the BIOS entry code is effectively the index relative to WBOOT in a CP/M 2.2 BIOS. This number is multiplied by 3 and added to the address in the JMP vector at location 0000H on the Z89.

The header is always sent to the Z89, and additional data as required (WRITE sends the 128-byte contents of the current DMA address). Some functions do not return any header.

Note that things like the current DMA address on the 77422 have no meaning on the Z89. These sorts of adaptations are done by the respective sides.

Also note that since the 77422 is not running the Network ROM code when in coprocessor mode, the protocol is purely at the agreement of the two sides of the function-shipping code.

Network Protocol

The protocol on the wire is SDLC over RS-422, clock and data pairs, half-duplex. The network topology is a physical bus but logically is a token-passing ring. Nodes have two network connectors as a convenience only. It is not required that both connectors be used.

Once the token owner is established, the token owner is allowed to send messages and will then pass the token along to the next active node in the table. The token message includes the network table, and thus the network table represents the accumulation of all information collected by each node. When a node owns the token, it is also responsible for polling offline node IDs to see if they have come online, and will record that information in the network table. An offline node must monitor the network for traffic, and only if the network is idle for a long period of time will it declare itself the token owner and broadcast a new token reset message. Since a normal network will include a token owner that polls, sends messages, and passes the token along, the network should never be idle for long periods of time. A node’s idle timeout is based on it’s node ID, and thus only one node should timeout (first) and declare itself the new token owner.
The SDLC message frame on the wire is formatted as follows:

...flag	dest	ctrl	src	seq	data	...	crc	flag...

The “flag” bytes are SDLC sync flags (01111110b) and are inserted/discarded automatically by the Z80-SIO. The CRC is also computed, inserted, and checked/discarded by the Z80-SIO. ‘dest’ is the destination node ID, ‘src’ is the sender’s node ID, and ‘ctrl’ is the command byte. When a network message originated from the Z89, the ‘ctrl’ byte will match the command byte from the host header.

‘seq’ is a sequence number, used to detect retry (redundant) frames. ‘data’ the the message payload, and does not include the host header.

The ‘seq’ byte contains a retry counter in the low 4 bits, and the high bit is used to indicate when a frame is a retry. A frame with ‘seq’ bit 7 “0” will always be accepted (NOTE: out of order delivery is not possible). If a retry frame that matches the current sequence number for the sending node is received, it will be ignored. Any frame that is accepted will have it’s sequence number updated into the table entry for the sending node. A frame will be retried 10 times before declaring a send failure.

Sequence number processing and retry is a higher level function, and the 77422 only performs this when acting a printer server. The Z89 must perform that function for CP/NET, if it desires it.

The control messages ACK, NAK, and BSY are used to provide confirmation of delivery. These messages contain no ‘seq’ or ‘data’. These messages are sent regardless of any response message. Every message sent on the wire, type 0 through type 14, requires a confirmation message.

Broadcasting is supported, using the “node ID” 255. Broadcast messages never have a confirmation.

Type 0 messages are used to transmit CP/NET requests/responses

Ctrl	**Function**	**Data**
00H	CP/NET Request	cpnet...
01H	CP/NET Response	cpnet...
02H	CP/NET Mail	Implementation Unknown (more like chat than email?)

Requests are generally only sent by CP/NET Clients (“requestors”),
Responses are generally only sent by CP/NET Servers.

Type 1 messages are used to boot a node, either the host or the 77422.

Ctrl	**Function**	**Data**
10H	Load and goto on 77422	Addr, code...
11H	Load and goto on host	Addr, code...
12H	Load on 77422	Addr, code...
13H	Load on host	Addr, code...
14H	Goto on 77422	Addr
15H	Goto on host	Addr

‘Addr’ is 2-bytes, little-endian.

Type 2 messages are used to request booting of a node, either the host or the 77422.

Ctrl	**Function**	**Data**
20H	Request Boot	Device code

Ctrl	Function	Data
28H	Boot refused	

Device codes 0 (77422) and 1 (Z89) are defined.

Type 13 messages are used to pass the token.

Ctrl	Function	Data
D0H	Pass token	next, net.table
DFH	Error reset	

Error reset is broadcast to announce that a new token owner exists, after network timeout.

‘next’ is the node ID of the next offline node to poll. It is essentially +1 the node ID that was last polled.

‘net.table’ is the status of all possible 64 nodes, one byte per node ID. Each byte contains the node type code in the high 4 bits, and the low 4 bits contain a counter. For offline nodes (node type 0), the counter represents a poll counter and will freely cycle 0-15. For online nodes (node type 1-15), the counter represents a dead count, the number of consecutive failed attempts to communicate. This count is reset whenever a node responds to communication. NAK is considered a successful communication, as it indicates that the node is still alive. When the dead count reaches the maximum, the node is switched to offline.

Type 14 messages are used to poll offline nodes.

Ctrl	Function	Data
E0H	Poll	

A poll should always be ACKed, regardless of whether the node is already online.

Type 15 messages are used to communicate message delivery status.

Ctrl	Function	Data
F0H	ACK - Acknowledge	
F1H	NAK - Negative Acknowledge	
F2H	BSY - Busy (printer server)	

All receivers of messages type 0-14 should immediately acknowledge them, and if requiring a response will send that later (when owning the token). NAK or BSY indicate that no formal response will be coming. Type 15 messages are the only messages a node is allowed to send unless it owns the token.

Debugging Monitor

The firmware includes debug monitor code, that may be used by attaching an RS-232 terminal to the serial port. See schematics for cable pin-out. The terminal should be set for 8 data bits, no parity, 1 stop bit. Normally, board jumpers will be set for 9600 baud, so use a matching rate. The Z80-SIO is programmed for “auto enables”, so both CTS and DCD must be asserted in order to use the port. This port will assert RTS and DTR, which may be looped-back to CTS and DCD if needed.

To get the attention of the firmware, press RETURN. You will get an “MMS-net:” prompt. The only two recognized commands are ‘B’ (network boot) or ‘D’ (enter debug monitor). Any other character returns to the initial, idle, state (the firmware is never spinning on character input, unless in debug monitor mode). [network boot is not discussed here]

On entering debug mode, the message “MMS 77422 monitor M2001-35” is printed followed by the prompt character, ‘:’. At this point the monitor is accepting commands. The command ‘?’ prints help. The monitor is very similar to a subset of CP/M’s DDT commands. Also, while in the monitor the firmware is ignoring the host interface (unclear whether token passing is still accepted).

The board may also be put in the debug monitor mode from the host, using the designated command.

Use the ‘R’ command to return to normal operation, or RESET the board. Note, the host interface /RESET is separate from the internal RESET. Only the reset button on the network dongle will perform a true RESET, and the nearest equivalent is for the host to trigger an NMI.

Theory Of Operation

The intent of the 77422 was to provide network connectivity to CP/M machines while offloading as much of the work as possible. To maintain responsiveness, both to the Z89 as well as the network, an AM9517 DMA controller was added to the board. A Z80-SIO is used as the network protocol engine, programmed to SDLC mode. Some additional circuitry is used to detect IDLE and collision events. An AM2950 parallel port IC is used as the interface to the host. 64K of DRAM is provided on-board, with an option to expand to 256K. Note that the DMA controller is unable to access extended memory. An 8Kx8 EPROM is used for the firmware.

The DMA controller does most of the I/O as well as larger memory copy operations. DMA channel 0 is connected to the SIO, and operates in a half-duplex mode (same as the network). DMA channel 2 is used to receive data from the host, and channel 3 is used to send data to the host. DMA channel 1 is not connected.

Z80-SIO channel A is the network port, and operates at 500Kbps. The handshake signals are extended off-board, but are not used in any known implementations. The port is programmed to synchronous, SDLC, mode and will automatically ignore messages that are not addressed to the programmed node ID.

Z80-SIO channel B is an auxiliary RS-232 Asynchronous serial port. It is used for debug purposes, or in applications where the 77422 is housed stand-alone as a printer server or diskless workstation. This port can operate at 9600 or 1200 board, or with firmware changes (or software extensions) may operate at any standard rate between 600 and 19200.

A 32-byte PROM is used to decode the I/O space for the Z80-CPU (DMA driven I/O works off signals from the DMA controller directly). The CPU I/O space is divided into 32-port segments, with some of those unused. Ports 0-3 are used for the Z80-SIO, port 40H is the control/status port for the 77422, ports 80H-8FH are the DMA controller, and port E0H is auxiliary access to the host interface, and associated status conditions.

A 512-byte PROM does the main DRAM/EPROM selection. It selects 1-of-4 64K banks of DRAM, or the EPROM, or conditions required for DRAM Refresh cycles.

A PAL16R4 is used to do some specialized system timing and control. It also controls memory mapping during DMA operations. This PAL operates off an 8MHz clock in order to achieve the necessary timing granularity.

The contents of the PROMs and PAL are not known at this time.

Refer to specific chip documentation for details on using Z80-SIO, AM9517, and AM2950.

The input port at 40H has the following bits:

7	6	5	4	3	2	1	0
/jp2	Idle	/Node ID					

Note that 'jp2' and node ID bits are inverted – jumper present appears as a "0".

'Idle' may be latched or free-floating, depending on the setting of the 'ilat' control bit.

The output port at 40H controls the following bits:

7	6	5	4	3	2	1	0
led2	led1	ilat	X	X	map		

'ilat' controls the IDLE detection latch. A '1' will prevent 'Idle' from returning to 0 once it has been triggered.

'map' selects the current memory bank mapping:

value	**Memory map**
0	ROM 0000-1FFFH, bank 0 RAM at 2000-FFFFH
1	Bank 0 RAM at 0000-FFFFH
2	Bank 1 RAM 0000-DFFFH, bank 0 RAM E000-FFFFH (56K common)
3	Bank 1 RAM 0000-BFFFH, bank 0 RAM C000-FFFFH (48K common)
4	Bank 2 RAM 0000-DFFFH, bank 0 RAM E000-FFFFH (56K common)
5	Bank 2 RAM 0000-BFFFH, bank 0 RAM C000-FFFFH (48K common)

value	Memory map
6	Bank 3 RAM 0000-DFFFH, bank 0 RAM E000-FFFFH (56K common)
7	Bank 3 RAM 0000-BFFFH, bank 0 RAM C000-FFFFH (48K common)

The 77422 uses Z80 interrupt mode IM2 (vectored interrupts). The only true IM2 interrupting device is the Z80-SIO. The DMA controller does not interrupt the CPU, and the INT-422 from the host is designed to let the interrupt vector bits float to FFH. The Z80 will actually fetch the second byte of this vector from the next page, rather that wrap around inside the same page. This is the only case where the Z80 will fetch interrupt vector data from outside the page defined in the I register. The net effect is that the Z80-SIO interrupt vectors must start at XXE1H (where ‘XX’ is the value in the I register).

The firmware will create the interrupt vector table in RAM, at 20E1H. This allows for dynamic adaptation of the interrupts – for example a “state machine” for input from an attached terminal is implemented by changing the channel B Rx interrupt based on the previous key pressed.