

# MIPP Data Cable protocol

## Physical definition:

Each front-end is fitted with two RJ-45 connectors, each of which contains four differential pairs. Two pairs are bussed; two other pairs are daisy-chained. The controller transmits on the bussed pairs and receives on the daisy-chained pairs (Fig. 1). One of the bussed pairs (timing bus) is used to send trigger, timing and initialization (Fast Command, Clk/FC). Control information and requests for status are sent on the other bussed pair (control bus, Control). The timing bus and control bus require a termination at the further end. The front-ends transmit on the daisy-chained pairs. One is used for sending event data (data chain), the second for sending status information (status chain). The direction toward the readout controller is defined as “downstream”. The number one front-end is defined to be the front-end furthest from the controller (see Fig. 1).

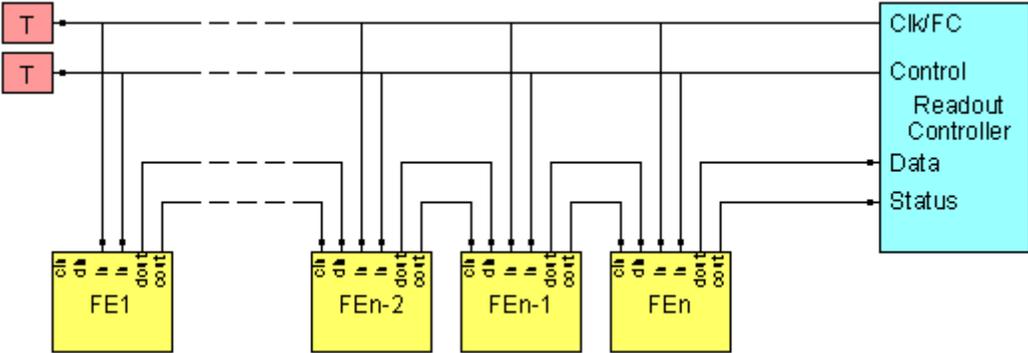


Figure 1 MIPP Interconnection Scheme

## Electrical definition:

The signal levels on the pairs within a cable are defined as LVDM, which is designed to drive a double terminated line. Transformer coupling is required on bussed pairs and at least on two chained outputs. The encoding scheme is FM with bits represented as two frequencies: 26.5MHz (logic 1) and 13.28MHz (logic 0) (see Fig. 2). Front-ends are required to phase lock their transmit channels to the timing bus clock (RF/2). Data transmitted by the front-ends will be at a known frequency, and a fixed but unknown phase with respect to the synchronous link.

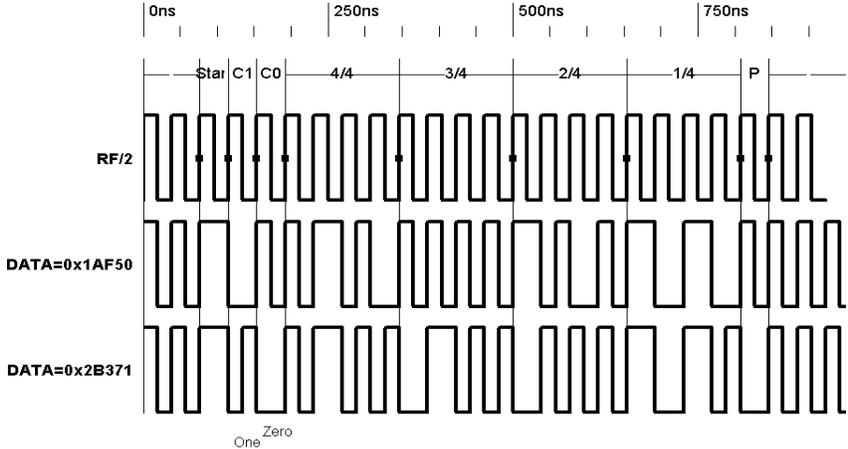


Figure 2 FM Encoding

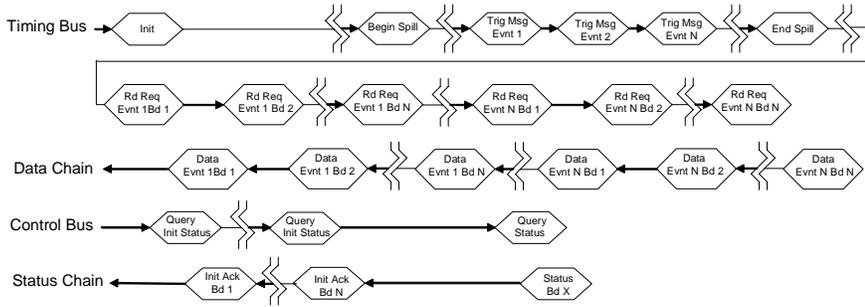
**Logical definition:**

The information is transmitted serially as packets called frames. Each frame consists of a 20 bit sequence beginning with a start bit (S=0) followed by 18 bits of data, and a parity bit (P) (see Fig. 3).

Bit Position																			
Start Bit	Cmd Bits		Data Bits																Parity
S	C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00	P

**Figure 3 Frame Definition**

A minimum interval of at least one clock period between frames is required. A string of 1's is sent between frames. The first two bits of the data field are used to distinguish between control and data words. Two control codes are used to define a begin event and end event character. The order of the readout is arbitrary, but for convenience of the data processing the last front-end in the chain (address 1) will send event data first followed by its neighbor and so on until the last transmission by the first front-end in the chain (address N). The expectation is that the controller will instruct each front-end to send event data. This command will include event synchronization information. Any front-end that is not transmitting its own data must transparently pass the data received from its upstream neighbor to its downstream neighbor. When a front-end changes from one transmit state to the other, the phase of the FM cannot be disrupted, so as not to confuse the receiver on the downstream neighbor (Fig. 4).



**Figure 4 Event Control and Data Flow**

When a response to a message from the front-end is required, the readout controller has to wait for a minimum timeout period and generate an error if there is no response.

**Chain Address assignment sequence:**

The daisy chained links allow for the automated front-end address assignment on a cable. When an assign address message is broadcast on the control bus, all front-ends will block the re-transmission of data from their daisy-chained links until after they have received a response word from their upstream neighbor. The last front-end in the chain will send a response message on its status pair and assign itself a requested address in the chain. The FM protocol on the cable requires that all transmitters will be sending a pulse train when idle. The absence of this pulse train on its upstream connector will allow a front-end to identify itself as the last in a chain (see Fig. 1). The response must include the front-end address. Each of the remaining front-ends will receive a response from its upstream neighbor and increment the address information by one, assign itself this address and send the modified response to its downstream neighbor. After a front-end has sent its configuration response, it must transparently link the upstream and down stream daisy chained ports. The first front-end will send its address not to another front-end, but to the readout controller. After a specified time interval the controller will be able to know how many front-ends are on the chain.

## Trigger/Timing Link Messages:

Note that upper two bits (C1, C0) are used to distinguish between various types of messages sent over the timing bus.

### Reset:

In order to facilitate hardware reset a disruption of the FM signal is used. An internal hardware reset equivalent to the power up reset will be generated by the front-end when the FM signal at the timing bus is static for ~ 10  $\mu$ s.

### Initialization:

This message would presumably be sent once per run. This will refresh software setup parameters, clear any data in the event buffers, reset the event counters and any timers used, and clear any latched error bits (**0xF500**).

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
0	0	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	0

### Clear status:

This message would presumably be sent once per spill. This will clear any latched error bits (**0xF501**).

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
0	0	1	1	1	1	0	1	0	1	0	0	0	0	0	0	0	1

### Generate test pulse:

This message may be sent within the spill to generate test events (**0xF701**).

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	1

### Begin Spill:

At this time the 53MHz timers used for event synchronization would be reset (**0xF301**).

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
0	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	1

### End Spill:

This message is an indication that readout may now start (**0xF302**).

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
0	1	1	1	1	1	0	0	1	1	0	0	0	0	0	0	1	0

### Triggers:

Trigger information will include event synchronization data (E0...E9) and some trigger selection bits (T0...T5). The desire is to keep the length of the trigger message shorter than the minimum time between triggers for the sake of simplicity. The trigger messages can only be sent during the spill (i.e. between Begin Spill and End Spill signals).

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
1	0	T5	T4	T3	T2	T1	T0	E9	E8	E7	E6	E5	E4	E3	E2	E1	E0

### Read specified event:

This message would include event synchronization information (E00...E15). Front-ends are required to send exactly one events' worth of data in response to a read event request. Event data is restricted to sequential access, and cannot be accessed during the spill. Each front-end checks that the stored synchronization information corresponding to the event fragment ready to be sent in its buffer matches the synchronization information in the read next event message. Note that this is the only message on this bus that has both upper bits (C1, C0) set to one.

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
1	1	E15	E14	E13	E12	E11	E10	E09	E08	E07	E06	E05	E04	E03	E02	E01	E00

### Status/Control Link Messages:

These messages are for writing front-end setup parameters and returning slow control and monitoring information. The details of this data are front-end dependent. Note that upper two bits (C1, C0) are used to distinguish between various types of messages sent over the control bus. No status messages can be processed during the spill.

### Assign address:

This message will be sent by a controller to initiate automatic address assignment (AAA, 0xF0XX). Bits A0...A7 represent the initial address assigned to the first front-end in the chain.

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
1	1	1	1	1	1	0	0	0	0	A7	A6	A5	A4	A3	A2	A1	A0

### Write to register:

This message will be sent by a controller to write a 16-bit value to a register. Bits A0...A7 represent the internal front-end address. Bits C0...C7 correspond to the assigned front-end chain address. Note that this is two-word message.

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
0	1	C7	C6	C5	C4	C3	C2	C1	C0	A7	A6	A5	A4	A3	A2	A1	A0

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
0	1	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

### Read from register:

This message will be sent by a controller to read the content of a register. Bits A0...A7 represent the internal front-end address. Bits C0...C7 correspond to the assigned front-end chain address. The front-end will respond with 16-bit word shown below.

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
1	0	C7	C6	C5	C4	C3	C2	C1	C0	A7	A6	A5	A4	A3	A2	A1	A0

This is front-end's response.

C1	C0	D15	D14	D13	D12	D11	D10	D09	D08	D07	D06	D05	D04	D03	D02	D01	D00
1	0	D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0

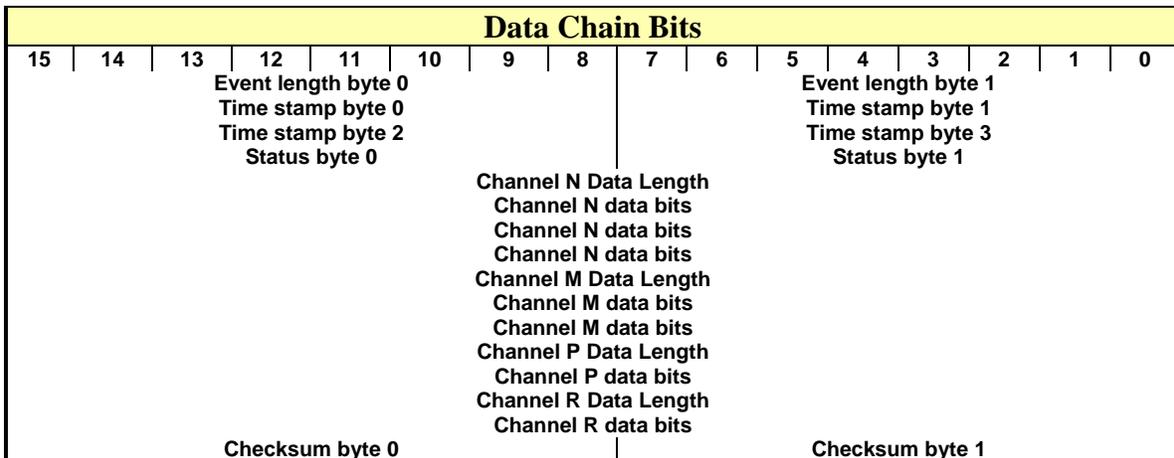
## Error handling:

There are two classes of errors: those associated with particular events (e.g. sync errors) and those associated with the state of the front-ends (e.g. buffer overflow, link parity error). Trigger synchronization errors must be handled in real time by setting a specific error bit in the event header. Also, a specific error bit has to be set in the front-end static status register to indicate occurrence of this error. Sync errors during readout have to be handled in real time by setting error bit in the status word in the event header. A different error bit has to be set in the front-end static status register to indicate occurrence of this error. The other class of errors would be included in the status information send in response to status queries by the controller. See "Read from register" message described above.

## Event data format:

### From the front-end:

Since the data destination of the front-ends is a VME module, Big Endian byte ordering is chosen. The event data block consists of an event word count, followed by synchronization information, a status word and, finally, the subsystem specific event data. A two byte checksum is appended at the end of the event. The event word count is inclusive of itself. Each front-end must provide enough information for the software to extract individual channel data. Data length words have to be used for zero compressed data of variable length. *Bits (C1, C0) are set to 01 in the first word, to 00 for all data words and to 10 in the checksum word.*



### From the controller:

The event data from all the front-ends will be concatenated into a single block. Redundant information from the individual front-ends will be stripped off. The block will consist of a word count, one set of synchronization information a generic header block, a subsystem specific header block followed by the data from all the relevant front-ends attached to the controller.

## Front-end functional requirements:

### Addressing:

Each front-end must respond to an assign address command and be capable of assigning itself an address based on data received from its upstream neighbor and passing the appropriate message to its downstream neighbor.

**Setup data:**

Each front-end must respond appropriately to setup reads and writes when addressed. Each front-end must power up in a specific known state. In this state each front-end must transparently re-transmit any input data on status and data chains. The power up reset issued by the controller has to be followed by the address assignment sequence described above. Optionally, front-ends may have the ability to recover setup parameters on power up reset from non-volatile storage. The front-ends will not accept setup commands between begin-of-spill and end-of-spill timing messages.

**Initialization:**

Each front-end must respond to an initialization command by refreshing its setup parameters, resetting any counters, flushing its data buffers etc. The controller may poll the front-ends to determine when their initialization sequences are complete.

**Triggering:**

On receipt of a trigger message, each front-end must be capable of cross checking the synchronization data contained in the message. Synchronization information must be stored with each event for further checks during readout. There must be a capability for delay adjustment of the on-board event synchronization timers to compensate for signal propagation differences of the data cables. Any synchronization errors must be reported in the event data header. Each front-end must have the ability to store (e.g. pipeline) data long enough for a trigger decision to be made. The front-ends will not accept triggers between end-of-spill and begin-of-spill messages.

**Buffering and readout:**

Each front-end must be capable of storing all the event data for a four second spill which is specified to be 20,000 events. Thirty seconds are allotted for transferring the data from the front-ends to the controller. The front-ends will not accept readout requests between begin of spill and end of spill.