ANSI/VITA 12-1996

Approved as an American National Standard by ANSI

American National Standard

for The Mezzanine Concept M-Module Specification

Secretariat
VMEbus International Trade Association

Approved May 20, 1997
American National Standards Institute, Inc.

VMEbus INTERNATIONAL TRADE ASSOCIATION
7825 E. Gelding Drive, Suite 104, Scottsdale, AZ 85260-3415
PH: 480-951-8866, FAX: 480-951-0720
E-mail: info@vita.com, URL: http://www.vita.com

ANSI/VITA 12-1996

American National Standard
for M-Module

Secretariat
VMEbus International Trade Association

Approved May 20, 1997
American National Standards Institute, Inc.

Abstract
This specification defines minimum mechanical and electrical characteristics of M-Modules, a method of implementing modular circuit boards with specific functions that can be used to add functionality to other larger printed circuit boards.

American National Standard

Approval of an American National Standard requires verification by ANSI that the requirements for due process, consensus, and other criteria for approval have been met by the standards developer.

Consensus is established when, in the judgment of the ANSI Board of Standards Review, substantial agreement has been reached by directly and materially affected interests. Substantial agreement means much more than a simple majority, but not necessarily unanimity. Consensus requires that all views and objections be considered, and that a concerted effort be made toward their resolution.

The use of American National Standards is completely voluntary; their existence does not in any respect preclude anyone, whether they have approved the standards or not, from manufacturing, marketing, purchasing, or using products, processes, or procedures not conforming to the standards.

The American National Standards Institute does not develop standards and will in no circumstances give an interpretation of any American National Standard. Moreover, no person shall have the right or authority to issue an interpretation of an American National Standard in the name of the American National Standard Institute. Requests for interpretations should be addressed to the secretariat or sponsor whose name appears on the title page of this standard.

CAUTION NOTICE: This American National Standard may be revised or withdrawn at any time. The procedures of the American National Standards Institute require that action be taken periodically to reaffirm, revise, or withdraw this standard. Purchasers of American National Standards may receive current information on all standards by calling or writing the American National Standards Institute.

Published by

VMEbus International Trade Association
7825 E. Gelding Drive, Suite 104, Scottsdale, AZ 85260

Printed in the United States of America - R1.0
ISBN 1-885731-07-8

Contents

Foreword

(This foreword is not part of VITA 12-199x.)

There is a relatively large choice of I/O boards for standard bus systems such as the VMEbus.

Boards with the Single Eurocard and Double Eurocard formats are particularly widespread. The Double Eurocard format has the advantage of better utilization of volume and lower overhead for the bus interface. Single Eurocards have the advantage of greater modularity but do not permit peripheral equipment to be connected on the bus side with a second connector. Both concepts are relatively rigid when it comes to implementing custom solutions.

The search for a solution that would combine the advantages of both formats at low cost and allow a large degree of flexibility for different applications led to the development of the M-Modules.

Comparison of I/O concepts:

Criterion	Single Eurocard	Double Eurocard	M-Module
Volume utilization	poor	good	very good
Connection capabilites	very poor	very good	very good
Overhead for bus interface	very high	low	low
Expense for small applications	low	very high	very low
Expense for medium-size applications	high	high	low
Expense for large applications	very high	low	low
Service costs	low	high	low
Flexibility	high	low	very high
Effort for custom development	low	high	very low

Advantages of M-Modules and MA-Modules

- up to four (different) modules per Double Eurocard base board
- two methods of connecting peripherals:
  - direct, using the front panel
  - indirect, via the base board using a 24-pin connector
- uncomplicated interface to the base board
- 16-bit or 32-bit data bus
- 8-bit and 24-bit address bus
- interrupt and DMA capabilities
- module identification facility
- trigger lines
- optimum utilization of area and volume
- good thermal characteristics
- high mechanical stability
- open bus specification
- wide range of modules available through cooperation between vendors

There are two annexes in this standard. Annexes A and B are informative and are not considered part of this standard.

This M-Module specification was prepared by the Manufacturers and Users of M-Modules association.

Suggestions for improvement of this standard will be welcome. They should be sent to MUMM association, Simon-Schöffel-Straße 21, 90427 Nürnberg, Germany.

This standard was processed and approved for submittal to ANSI by Accredited Standards Committee on [title and alphanumerical designation of committee]. Committee approval of the standard does not necessarily imply that all committee members voted for its approval. At the time it approved this standard, the [designation] Committee had the following members:

Name, Chairman

Name, Vice-Chairman

Name, Administrative Secretary

Organization Represented	Name of Representative
xxxxxx	name
xxxxxx	name
xxxxxx	name
xxxxxx	name

VITA Standard
for Modular Industrial Computers-

The Mezzanine Concept-
M-Module Specification

1 Scope and purpose

1.1 Scope

This specification defines minimum mechanical and electrical characteristics of M-Modules, with the aim of ensuring that any M-Module can be used on any base board capable of being fitted with M-Modules. Also conceivable are base boards for special applications that can be fitted both with M-Modules according to this standard and with specialized modules that can only be used with the particular base board. This means that the user has access to a wide variety of standardized M-Modules but is not prevented from developing or using specialized (non-standard) modules.

For instance, the basic electrical specification specifies modules with a 2-row connector for communication between the base board and the module. All modules which comply with the minimum requirements of this specification are designated M-Modules.

For special requirements, a third row can be added to the base board connection. This is described in the mechanical specification. The use of the additional signals or extended use of existing signals (in compliance with the basic electrical specification) can be standardized to a certain extent. Therefore, certain signals and signal groups are defined in the extended electrical specification as regards their function, timing and electrical characteristics. Modules complying with these additional specifications are designated MA-Modules.

1.2 Purpose

The M-Module concept is deliberately designed to allow specialized modules to be developed simply, quickly and cheaply. These specialized development efforts can be undertaken by different vendors or by the customers themselves.

2 Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this VITA standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this VITA standard are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below.

IEC 68-2-6, *Basic Environmental Testing Procedures. Part 2: Tests. Test Fc and guidance: Vibration*; International Electrotechnical Commission; 1982

IEC 68-2-27, *Environmental testing. Part 2: Tests. Test Ea and guidance: Shock*; IEC; 1987

IEC 68-2-29, *Basic environmental testing procedures. Part 2: Tests. Test Eb and guidance: Bump*; IEC; 1987

DIN 40040, *Applicability classes and reliability figures for components of telecommunications and electronics*; Deutsches Institut für Normung, Burggrafenstr. 6, 10787 Berlin; April 1987

DIN 85-M3x8, *Flachkopfschraube mit Schlitz*; Deutsches Institut für Normung, Burggrafenstr. 6, 10787 Berlin

3 Conventions

3.1 Use of the slash

A slash preceding the name of signals that are level-significant denotes that the signal is valid when it has "L" potential.

A slash preceding the name of signals that are edge-significant denotes that the actions triggered by that signal occur on a high to low transition.

3.2 Hexadecimal numbers

Hexadecimal numbers are preceded by "0x".

4 M-Module characteristics

Characteristics of modules as regards their interface with the base board are given a fixed designation. Specific characteristics must then correspond to capabilities of the base board for this function to be used. For instance, a module which can generate an interrupt is designated an "interrupter". If it is desired to make use of this capability on the base board, then the latter must be capable of handling interrupts - it must be an "interrupt handler".

An interrupter M-Module can be used on a base board that cannot handle interrupts, but in that case without interrupts.

For instance, an MA-Module can be used only with functional limitations on a base board suitable only for M-Modules. If the MA-Module has 24-bit addresses (A24) and requires these, it cannot be used on base boards that have only 8 bit addresses.

Table 1 - Characteristics of M-Modules and their designation

Module	Characteristics
M-Module	Module which complies with the M-Module Specification and has a 2-row, 40-pin connector for communication with the base board
MA-Module	Module which complies with the M-Module specification and the MA-Module specification and has a 3-row, 60-pin connector for communication with the base board
A08	M-Module with an I/O address space of max. 256 bytes
A24	MA-Module with a memory address space of max. 16 Mbytes
D08	M-Module with an 8-bit data bus (D00..D07)
D16	M-Module with a 16-bit data bus (D00..D15)
D32	MA-Module with a 32-bit data bus (D00..D31)
INTA	M-Module with interrupt-request capabilities (type A, software-end-of-interrupt)
INTB	M-Module with interrupt-request capabilities (type B, hardware-end-of-interrupt without vector transfer)
INTC	M-Module with interrupt-request capabilities (type C, hardware-end-of-interrupt with vector transfer)
DMA08	M-Module with an 8-bit DMA interface (using D00..D07)
DMA16	M-Module with a 16-bit DMA interface (using D00..D15)
DMA32	MA-Module with a 32-bit DMA interface (using D00..D31)
TRIGI	MA-Module with trigger inputs
TRIGO	MA-Module with trigger outputs
IDENT	M-Module with identification

M-Module characteristics are hardware-stored in a serial EEPROM. Information is coded in 16 16-bit words as described in the following overview. The basic electrical specification contains a detailed description of the EEPROM.

Word 0 (sync. code):

- D15..D00 0x5346

Word 1 (module number):

- D15..D00 module number (binary coded)

Word 2 (revision number):

- D15..D00 revision number (binary coded)

Word 3 (module characteristics):

- D15
  - '0' no burst access
  - '1' burst access
- D14 reserved
- D13 reserved
- D12
  - '0' module does not need ± 12V
  - '1' module does need ± 12V
- D11
  - '0' module does not need + 5V
  - '1' module does need + 5V
- D10
  - '0' module without trigger outputs
  - '1' module with trigger outputs (TRIGO)
- D09
  - '0' module without trigger inputs
  - '1' module with trigger inputs (TRIGI)
- D08..D07
  - '00' no DMA requester
  - '01' 8-bit DMA (DMA08)
  - '10' 16-bit DMA (DMA16)
  - '11' 32-bit DMA (DMA32)
- D06..D05
  - '00' no interrupter
  - '01' interrupter method a) (INTA)
  - '10' interrupter method b) (INTB)
  - '11' interrupter method c) (INTC)
- D04..D03
  - '00' 8-bit data bus (D08)
  - '01' 16-bit data bus (D16)
  - '10' 32-bit data bus (D32)
  - '11' reserved
- D02..D01
  - '00' 8-bit address bus (A08)
  - '01' reserved
  - '10' reserved
  - '11' 24-bit address bus (A24)
- D00
  - '0' M-Module does not support memory access
  - '1' M-Module supports memory access

Words 4..7:

- D15..D00 0x0000 reserved

Words 8..15:

- D15..D00 user-defined

5 Basic electrical specification

5.1 Communication between M-Module and base board

Every M-Module

- can be accessed via a 16-bit data bus;
- has an I/O address space of 256 bytes, or 128 words;
- may be capable of generating interrupts and requesting a DMA transfer.

5.1.1 Signals

D00..D15 Bidirectional data bus. D00 is the least significant data bit, D15 is the most significant.

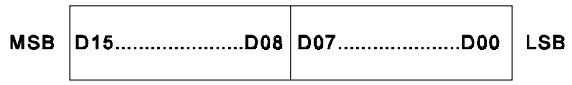


Figure 1 - Bit order

If the M-Module has only an 8-bit data bus, only D00..D07 shall be used, i.e. every second byte is used in a 16-bit structure.

A01..A07 Unidirectional address bus (input signals). A01 is the least significant address bit, A07 is the most significant. The 7 address bits make it possible to distinguish between 128 words on each module. A00 is not present. Differentiation between the two bytes of the word is made by lines DS0 and DS1 (q.v.).

/WRITE Input signal. If this signal is "L", the data bus (when active) is directed toward the module. If this signal is "H", the data bus is directed away from the module (if allowed).

/CS Input signal. If this signal is active, the module is being accessed. While /CS is "L", the address bus, /WRITE, DS0, and DS1 are valid; on write access (/WRITE is "L") the data bus is also valid. For read access, the module must keep the data stable as long as /CS is active.

/DTACK Output signal. With this signal, the module terminates access when /CS is "L". A defined period of time later, the /CS signal is deasserted. For read access, the data must remain available for a defined period of time after /DTACK (until /CS is deasserted).

/DS0..1 Input signals. They are valid while /CS, /IACK or /DACK is active. When DS0 is "L", the data bus is active from D00 to D07; when DS1 is "L", the data bus is active from D08 to D15. When DS0 and DS1 are "L", a 16-bit access to the complete word is performed. When /CS is active, at least one of DS0 and DS1 is active.

/DS0	/DS1	D00..D07	D08..D15
"L"	"L"	transfer	transfer
"H"	"L"	no transfer	transfer
"L"	"H"	transfer	no transfer
"H"	"H"	invalid	

/RESET Input signal. Resets the module to a defined state. It is normally activated on power-up.

/IRQ Output signal. The module can generate an interrupt by setting "L" at this pin. This signal must remain until the interrupt is reset by /IACK or software.

/IACK Input signal. /IACK acknowledges a pending interrupt. It is a matter for the module concerned to decide whether this signal is used to reset the interrupt or not. It is also for the module to decide whether it interprets this /IACK only as an "L" pulse of defined duration or as an access signal at which an interrupt vector has to be transmitted (via D00..D07). At all events, the signal is a reaction to the pending interrupt initiated by this module.

/DREQ	Output signal. A module can request a DMA transfer by setting "L" at this pin.
/DACK	Input signal. When it is "L", the base board performs a DMA transfer with this module. A read or write access is performed, but the /CS line is "H" and the address bus is invalid. With this cycle, the /DREQ line must be "H" if it is desired to prevent initiation of another DMA cycle.
SYSCLK	Input signal. 16-MHz clock, asynchronous to all other signals.

GND Logical reference signal, power supply connection for the supply voltages.

+5V Power supply connection.

+12V Power supply connection.

-12V Power supply connection.

5.1.2 Principles of operation

5.1.2.1 Read/write access

Neglecting interrupts, which are described elsewhere, data transfer can be divided into two different types of access - read access and write access. This distinction is expressed by the /WRITE signal. Timing is always the same, regardless of the number of bytes transferred in a cycle. How many bytes, and which bytes are transferred at one access, depends only on the signals /DS0 and /DS1. Timing during access is determined solely by the signals /CS and /DTACK. This substantially decreases the circuit complexity on the M-Modules.

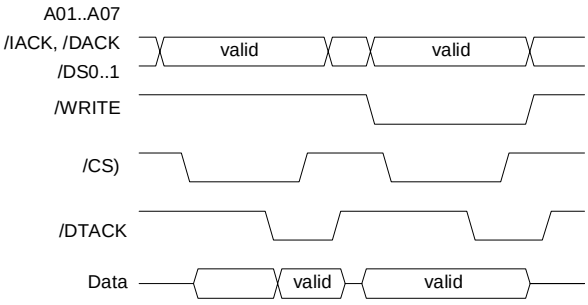


Figure 2 - Signal sequence of a read/write access

If there are only ("fast") hardware registers on an M-Module, /DTACK and /CS can be linked with each other, which results in well-defined access times.

5.1.2.2 Interrupt

The module can generate an interrupt by activating the /IRQ line. It is not necessary for masking of the

interrupt to be possible. This can always be done on the base board. Depending on the module concerned, there are three methods of responding to an interrupt on the module:

a) Type A, software-end-of-interrupt:

In the interrupt routine, the bus master resets the /IRQ line by accessing the module (for instance by reading a status register). At timing parameter #20a after the rising edge respectively #20b after the falling edge of /CS at the latest, the module shall de-activate the interrupt request line to signal that no further interrupt is pending.

b) Type B, hardware-end-of-interrupt:

At the beginning of the interrupt routine, an "L" pulse is set at the /IACK line of the module and this can be used to reset the /IRQ line. It is forbidden to assert /DTACK for this cycle. The width of the pulse is min. timing parameter #5 and max. timing parameter #3. At timing parameter #20a after the rising edge respectively #20b after the falling edge of /IACK at the latest, the module shall de-activate the interrupt request line to signal that no further interrupt is pending.

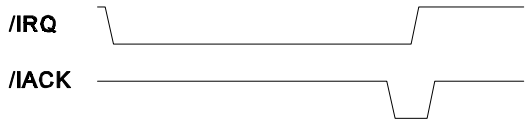


Figure 3 - Signal sequence of an interrupt with hardware-end-of-interrupt

c) Type C, interrupt vector transfer with hardware-end-of-interrupt:

The base board generates an interrupt acknowledge cycle, differing from a normal read cycle in that /IACK is asserted instead of /CS. The module must make its vector available at D00..D07. The vector is transferred as byte information using D00..D07. The addresses remain unused.

At timing parameter #20a after the rising edge respectively #20b after the falling edge of /IACK or /CS at the latest, the module shall de-activate the interrupt request line to signal that no further interrupt is pending.

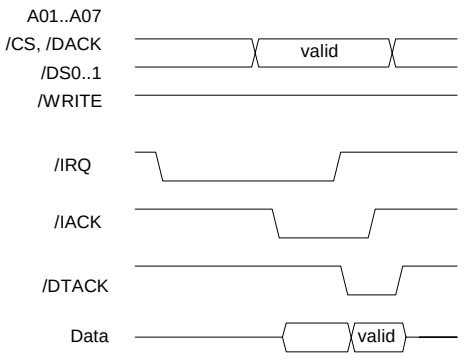


Figure 4 - Signal sequence of interrupt vector transfer cycle

Recommendation

Since it can be relatively difficult to construct base boards which support interrupt methods b) and c), all modules should have the alternative possibility of a "software-end-of-interrupt" (method a)).

5.1.2.3 DMA transfer

Any module can use the /DREQ line to request the base board to perform a DMA cycle with the module. This normally requires a DMA controller on the base board. The DMA cycle is similar to a normal read or write access except that the /DACK line is asserted instead of /CS. The address bus is not used for this cycle. DMA transfer can be on a byte or word basis.

With the end of a cycle (before /DTACK is "H") the module must reset /DREQ to "H" if it is desired to prevent initiation of another DMA cycle.

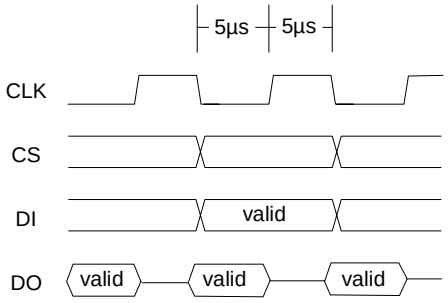
5.1.2.4 Module identification

The identification of an M-Module is optional. It is accomplished using the read and write operations already described. No special lines are necessary. No special timing applies for the accesses. Identification is performed by accessing a serial EEPROM with an organization of 16 by 16 words. This EEPROM must be compatible with the standard IC (9306). Accesses are performed after reset by read or write operations to the highest module I/O address. If this address is required for other purposes during normal module operation, it is permissible to disable the EEPROM by accessing the addresses 0..0xfd. In this case accesses to the EEPROM are only possible after a reset.

Since serial EEPROMs have very long access times (several µs), the control and data signals required for access to the EEPROM are stored in registers on the M-Module. The shift timing is controlled by software. For this reason, the EEPROM's input signals are obtained from data lines D00..D02 using registers. The data input is obtained from D00, the serial clock from D01 and the chip-select for the EEPROM from D02.

The registers are loaded by write operations in I/O space as appropriate with all address lines being "high". The serial information is read by read access to the same address. D00 represents the data output of the EEPROM.

The software performing the access must ensure that the following timing is adhered to:



Notes

1 CLK = Clock (registered D01)

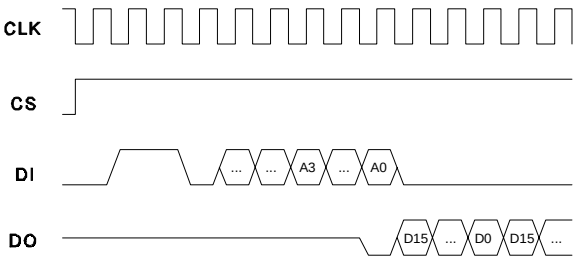
2 CS = Chip-Select (registered D02)

3 DI = Data input (registered D00)

4 DO = Data output (reading D00)

Figure 5 - Timing

In order to read serial information, it is first necessary to transfer a command to the EEPROM. For this purpose the following sequence must be used:



Notes

1 CLK = Clock (registered D01)

2 CS = Chip-Select (registered D02)

3 DI = Data input (registered D00)

4 DO = Data output (reading D00)

Figure 6 - Command transfer to the EEPROM

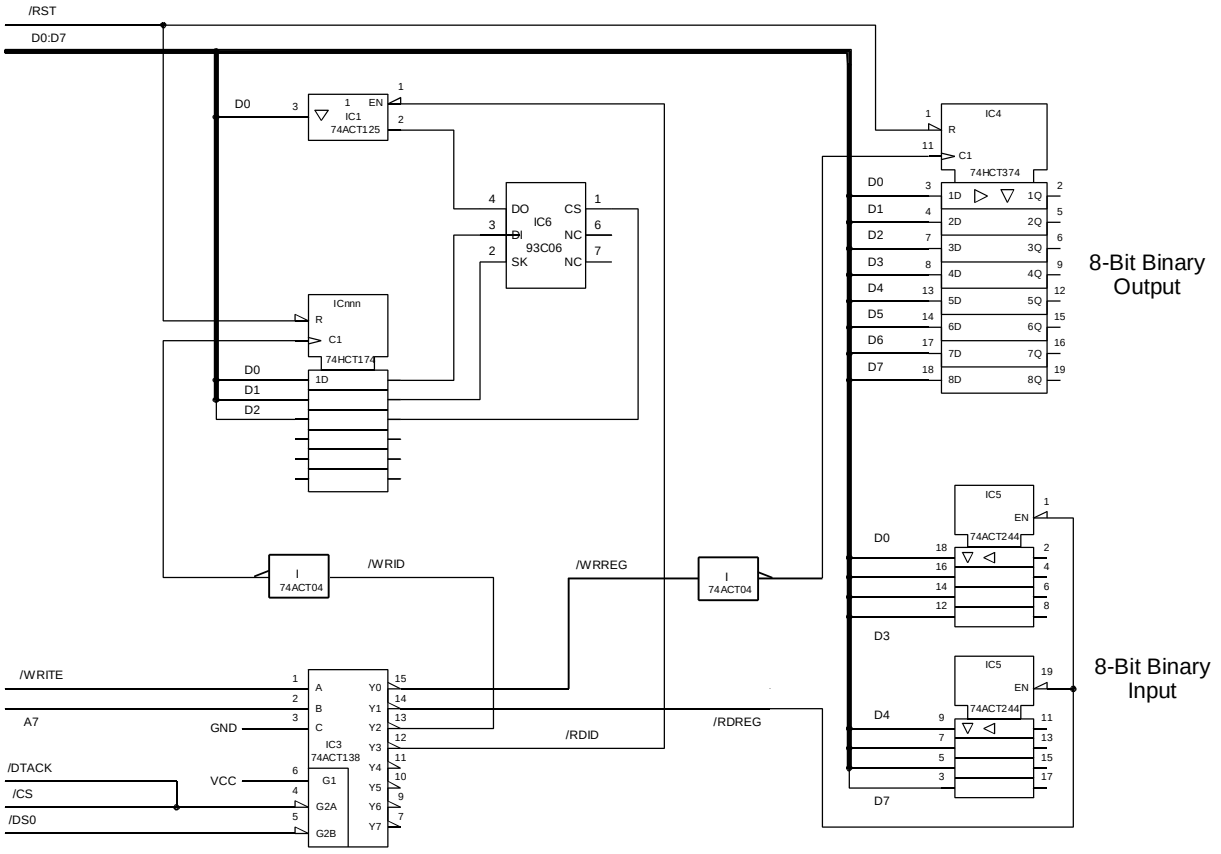


Figure 7 - Sample circuit for a module with module identification

5.1.2.5 Power-up

After power-up, i.e. after all voltages are within the prescribed tolerances and the SYSCLK signal complies with the specification, the base board must ensure that the /RESET signal remains active at least for timing parameter #12a. After /RESET is asserted, all other lines must be inactive. To reset the module after power-up, /RESET must be activated at least for timing parameter #12b.

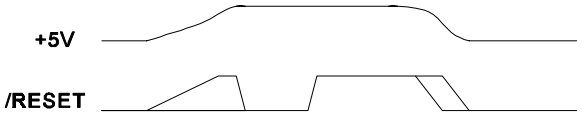


Figure 8 - Power-up and power-down

5.1.3 Driver Characteristics

Table 2 - Signal name and type

Signal name	Type	
D00..D15	TS I/O	A
A01..A07	input	B
/DS0..1	input	B
/CS	input	B
/WRITE	input	B
/DTACK	output	C
/IRQ	output	C
/IACK	input	B
/DREQ	output	C
/DACK	input	B
/RESET	input	B
SYSCLK	input	B

Table 3 - Electrical specification of the drivers over the whole temperature range

Type	Parameter, characteristic		Value
A,B	V_{ih}	Min. high-level input voltage	2.2V
A,B	V_{il}	Max. low-level input voltage	0.8V
A,B	I_{ih}	Max. high-level input current	-0.05mA
A,B	I_{il}	Max. low-level input current	0.2mA
A,C	V_{oh}	Min. high-level output voltage (I_{out} < 1.6 mA)	2.4V
A,C	V_{ol}	Max. low-level output voltage (I_{out} < 1.6mA)	0.4V
A,C	I_{oh}	Min. high-level output current (V_{out} > 2.4V)	1.6mA
A,C	I_{ol}	Min. low-level output current (V_{out} < 0.4V)	-1.6mA

The capacitive load of a pin must not exceed 25pF.

Normal controller chips fulfill these requirements, so, on modules including such controllers, it is not necessary to install data bus drivers. If it should be necessary to install data bus drivers for particularly simple modules without their own controller chips, types such as the 74ABT245 are suitable.

NOTE - It may be difficult to design a base board with both low-cost and high-performance design. Driver loads and bus driver should be carefully selected.

5.1.4 Power supply

The power for the module is supplied via GND, +5V, +12V and -12V. The following specifications apply:

Voltage	Tolerance	Max. current
+5V	+5%, -5%	1000mA
-12V	+5%, -5%	200mA
+12V	+5%, -5%	200mA

NOTE - The maximum currents indicated apply to M-Modules. Base boards must be designed in such a way that they can provide these power supplies.

Tolerance includes all ripple. However, one should bear in mind that the power supply must pass through various connectors and printed wiring. Every base board must be capable of supplying the module with all supply voltages using these lines.

5.1.5 Timing specifications

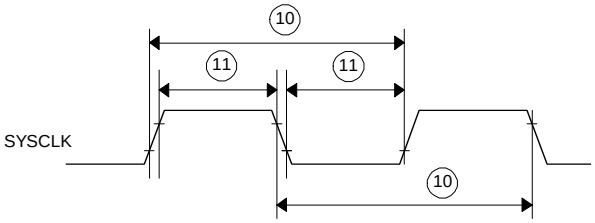


Figure 9 - SYSCLK timing

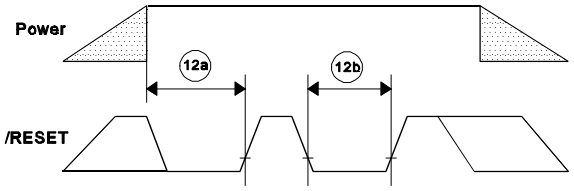
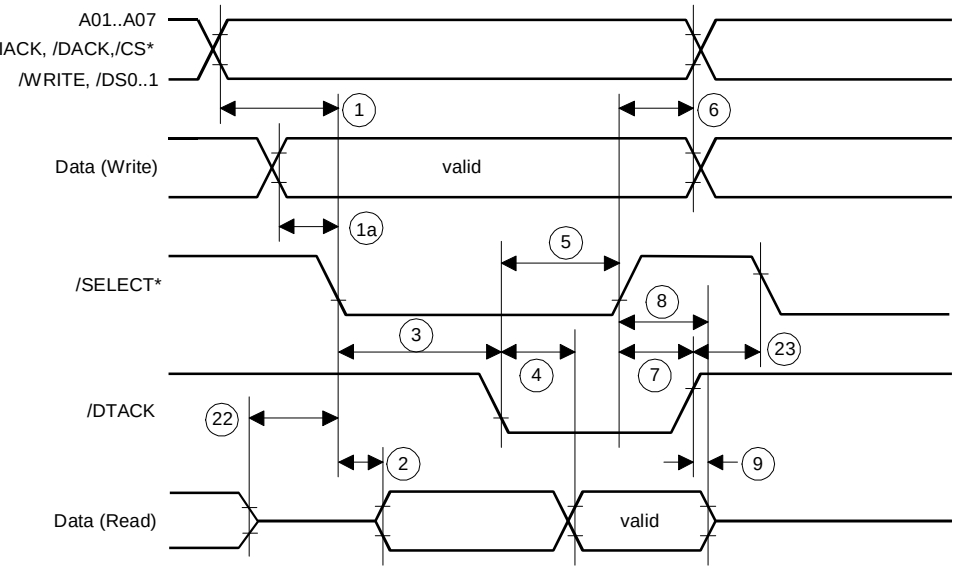


Figure 10 - /RESET timing

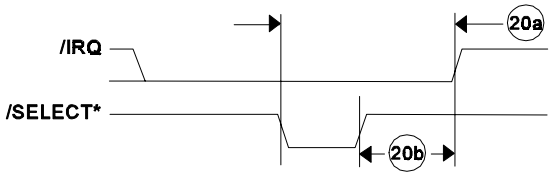


NOTES

*: Only one of the three signals /CS, /DACK and /IACK is asserted (named "/SELECT") "L" during a cycle; the others are "H" as indicated in the following table. Timing parameter #1 is irrelevant for the /SELECT signal.

Cycle type	/CS	/DACK	/IACK
Normal access	L	H	H
DMA access	H	L	H
Interrupt acknowledge	H	H	L

Figure 11 - Read/write timing



NOTES

*: Only one of the two signals /CS or /IACK is asserted (named "/SELECT") "L" during a cycle; the other is "H" as indicated in the following table. Timing parameter #1 is irrelevant for the /SELECT signal.

Cycle type	/CS	/IACK
Software-end-of-interrupt	L	H
Hardware-end-of-interrupt	H	L
Vector transfer	H	L

Figure 12 - Interrupt request timing

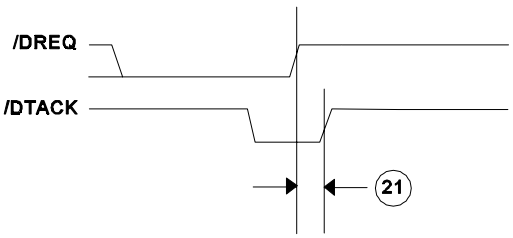


Figure 13 - DMA request timing

Table 4 - Timing parameters

No.	Characteristic	Min.	Max.	Unit
1	/WRITE, /DS0..1, /CS, /DACK, /IACK to /SELECT (Setup), A01..A07	10	-	ns
1a	/D00..15 to /SELECT (Setup)	0	-	ns
2	/SELECT to data-out active	0	-	ns
3	/SELECT to /DTACK	0	10000	ns
4	/DTACK to data valid	-	25	ns
5	/DTACK to /SELECT (Hold)	30	-	ns
6	/SELECT to /WRITE, /DS0..1, /CS, /DACK, /IACK, A01..A07, D00..15 (Hold)	10	-	ns
7	/SELECT = "H" to /DTACK (Hold)	0	100	ns
8	/SELECT = "H" to data invalid (Hold)	0	-	ns
9	/DTACK = "H" to data-out high impedance	-	25	ns
10	SYSCLK cycle time	61	64	ns
11	SYSCLK pulse width	22	40	ns
12a	/RESET low time, at power up	200	-	ms
12b	/RESET low time, at normal operation	1	-	µs
20a	/SELECT "L" to /IRQ inactive	0	-	µs
20b	/SELECT "H" to /IRQ inactive	-	1	µs
21	/DREQ to end of cycle (/DTACK = "H")	0	-	ns
22	Data bus floating to /SELECT active	0	-	ns
23	/DTACK inactive to /SELECT active	25	-	ns

5.1.6 Pin assignment

The control interface between the base board and the module is via two 20-pin plug connectors (pin headers) on the base board, which correspond to receptacles on the module.

Table 5 - Pin assignment

Pin	Row A	Row B
1	/CS	GND
2	A01	+5V
3	A02	+12V
4	A03	-12V
5	A04	GND
6	A05	/DREQ
7	A06	/DACK
8	A07	GND
9	D08	D00
10	D09	D01
11	D10	D02
12	D11	D03
13	D12	D04
14	D13	D05
15	D14	D06
16	D15	D07
17	/DS1	/DS0
18	/DTACK	/WRITE
19	/IACK	/IRQ
20	/RESET	SYSCLK

5.2 Connecting peripherals

Peripheral equipment may be connected using a front

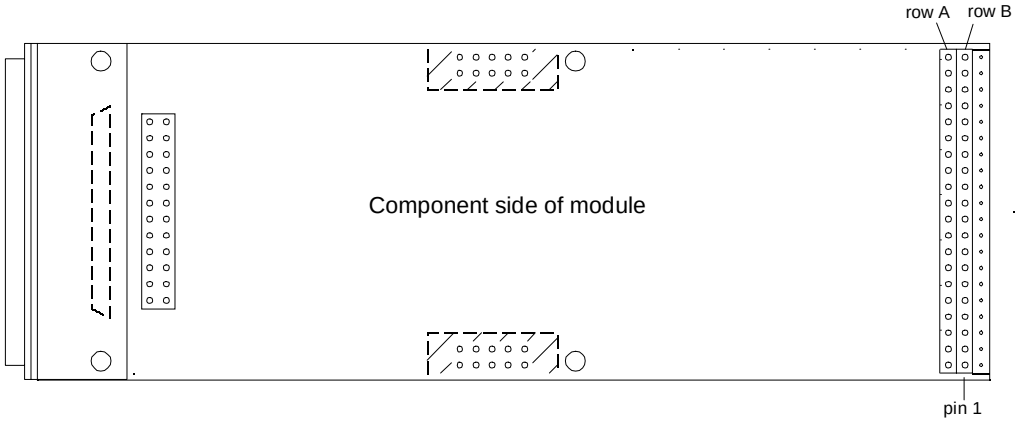


Figure 14 - Orientation of the receptacle connector on the module

panel connector, for instance a 25-pin D-Sub connector or by means of a 24-pin receptacle connector via the base board.

If the 24-pin receptacle connector is not used, the free space can be used only for components with a maximum height of 1.2mm.

5.2.1 Pin assignment of module

The pin layouts of the D-Sub connector (or other front-panel connector) and of the 24-pin receptacle connector are not defined.

5.2.2 Correspondence between 25-pin D-Sub connector and 24-pin peripheral connector

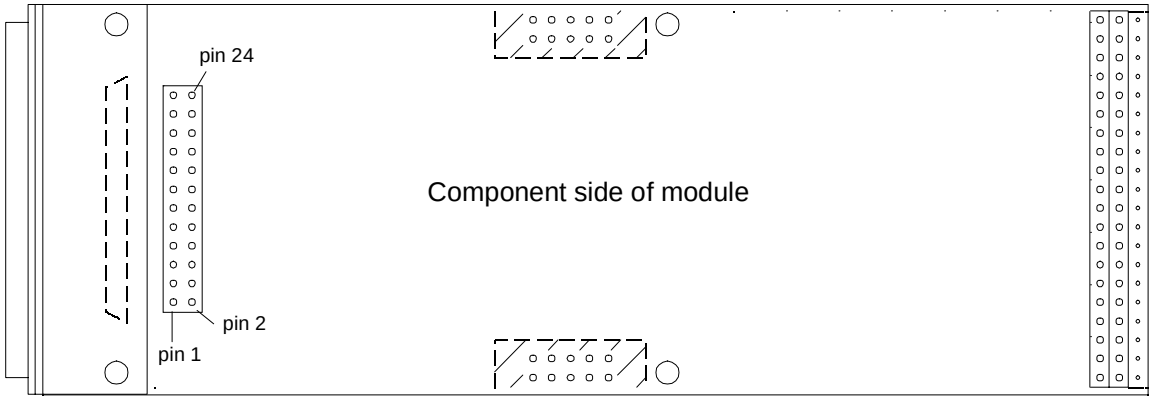


Figure 15 - Orientation of the 24-pin receptacle connector on the module

Recommendation

The 25-pin D-Sub connector is very common as a front panel peripheral connector for M-Modules. In principle, it is possible to connect different peripheral equipment signals via this connector and via the base board peripheral connector. However it is often desired to duplicate the signals. For this special - but very frequent - case, the correspondence between the two connectors is best implemented in a standard manner according to the following recommended scheme.

Table 6 - Recommendation for correspondence between 25-pin D-Sub connector and 24-pin peripheral connector

25-pin D-Sub		24-pin receptacle connector
receptacle	plug	
1	13	1
2	12	3
3	11	5
4	10	7
5	9	9
6	8	11
7	7	13
8	6	15
9	5	17
10	4	19
11	3	21
12	2	23
13	1	-
14	25	2
15	24	4
16	23	6
17	22	8
18	21	10
19	20	12
20	19	14
21	18	16
22	17	18
23	16	20
24	15	22
25	14	24

5.2.3 Electrical load

The electrical load on a connector pin depends on the plug component used, the receptacle component used and the printed wiring. No statements can be made about the front panel connector. As a rule, higher currents can be transmitted via the front panel connector than via the base board.

The breakdown strength (and thus the isolation voltages for electrically isolated modules) is usually higher if the 24-pin receptacle connection is not used. The current permitted to flow via the base board peripheral connector is limited to 1A per pin. However, this does not guarantee that the base board can tolerate such high currents! The breakdown strength between 2 pins is min. 100V AC.

5.3 Intermodule connections

A connection between one module and another may be necessary when several modules need to exchange data (for instance signal processors) without using the base board. Such connections are also desirable when modules are to be extended on the analog side - for instance A/D converters with multiplexers. The connection is made using two 10-pin connectors, arranged in such a way that it is possible to make a point-to-point connection from one module to the next using ribbon cable or plug adapter or to construct a bus connection joining several modules.

In certain cases it is even possible to connect modules on different base boards using ribbon cable with plug connectors. Note that these connectors are necessary only for intermodule connections, and modules which do not require such connections can use the space for other components.

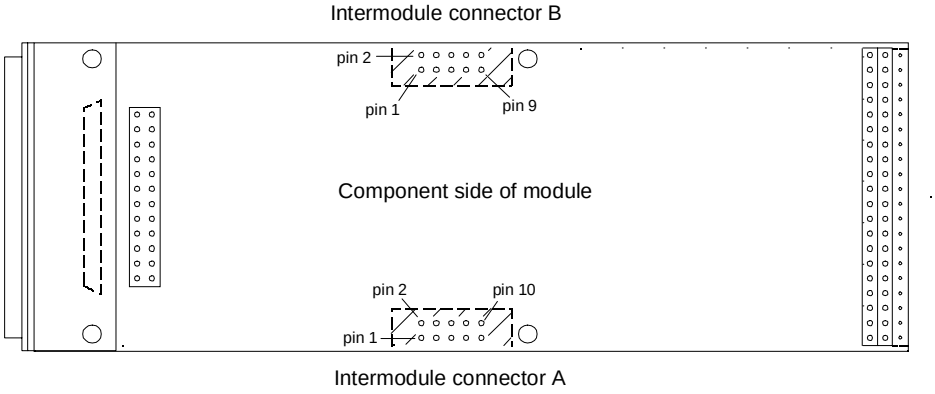


Figure 16 - Orientation of the 10-pin receptacle connectors on the module

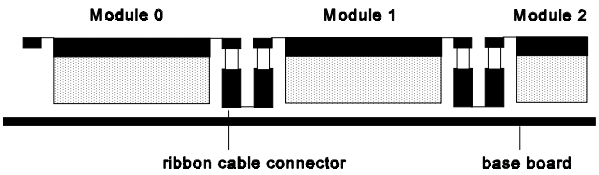


Figure 17 - Intermodule connector, cross section (point- to-point wiring)

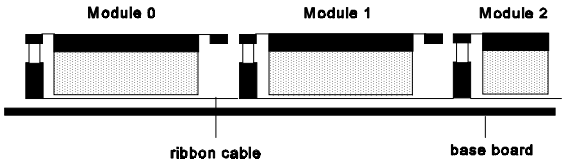


Figure 18 - Intermodule connector, cross section (bus connection)

5.3.1 Pin assignment

The pin assignment of the 10-pin receptacle connector is not fixed. The use of intermodule connections is specific to the module; for instance, analog or digital intermodule connections may be implemented.

5.3.2 Electrical specification

Since the scope of the intermodule specification is not fixed, the electrical characteristics cannot be defined. It should be pointed out that the reference ground for the signals could be line GND (i.e. the intermodule connection has an electrical connection to the base board), or it could be another potential isolated from the base board (for instance for analog multiplexers).

6 Mechanical specification

6.1 Dimensions of the modules

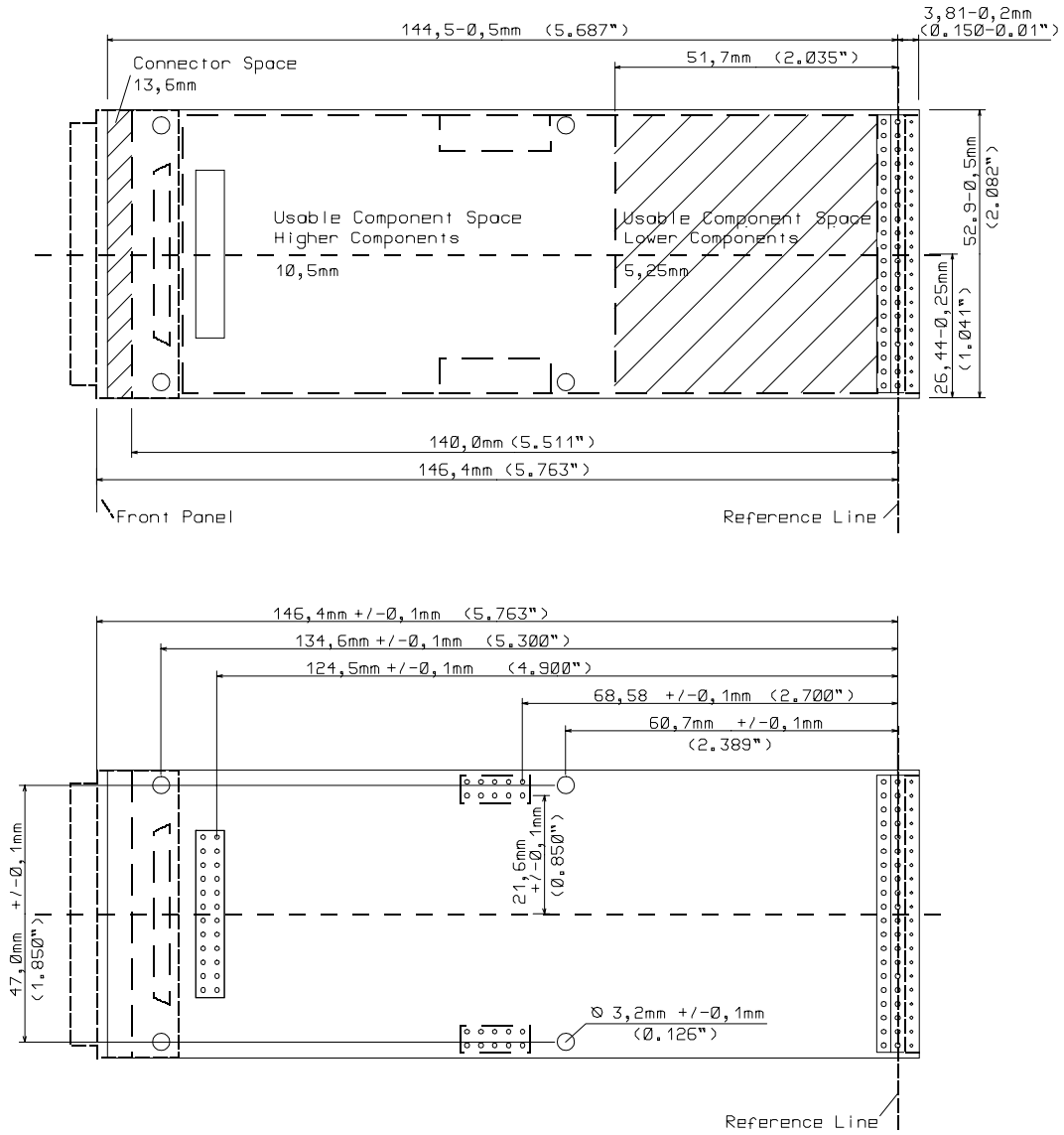


Figure 19 - Dimensions of a single MA-Module and position of connectors

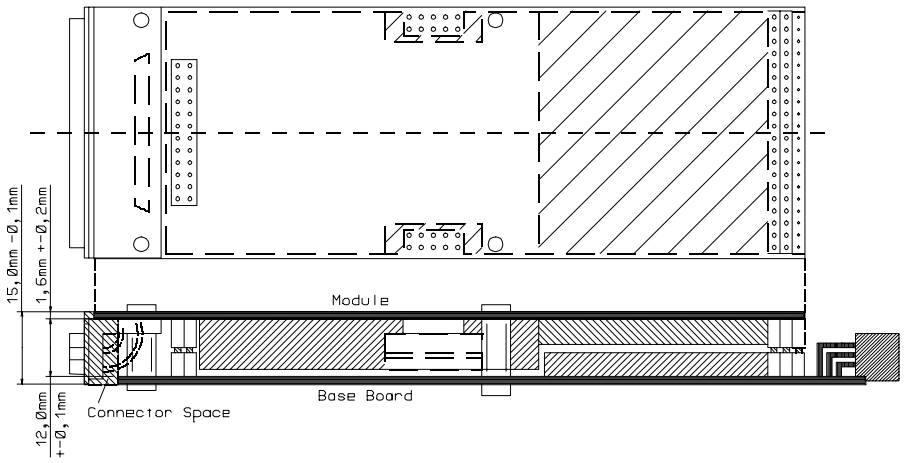


Figure 20 - Example for mounting a single MA-Module on a base board

The reference line is drawn through the mid-points of the pins of the rightmost row of the plug header connector and not the outer edge of the printed circuit board.

An area of at least 20.6mm by 8.5mm shall be left free of other components in the area of the Intermodule Port connector to leave room for connectors from different suppliers, if such a connector is used.

Optimum utilization of volume is of great importance in the M-Module concept. Thus the modules are made as large as possible.

Owing to the "sandwich construction" formed by the modules and the base board, the maximum height is, of course, less than for the base board alone.

In order to accommodate larger components, two component zones are defined, one for lower components (in an area where components may also be fitted on the base board) and one for higher components (in an area where only very flat components are allowed on the base board).

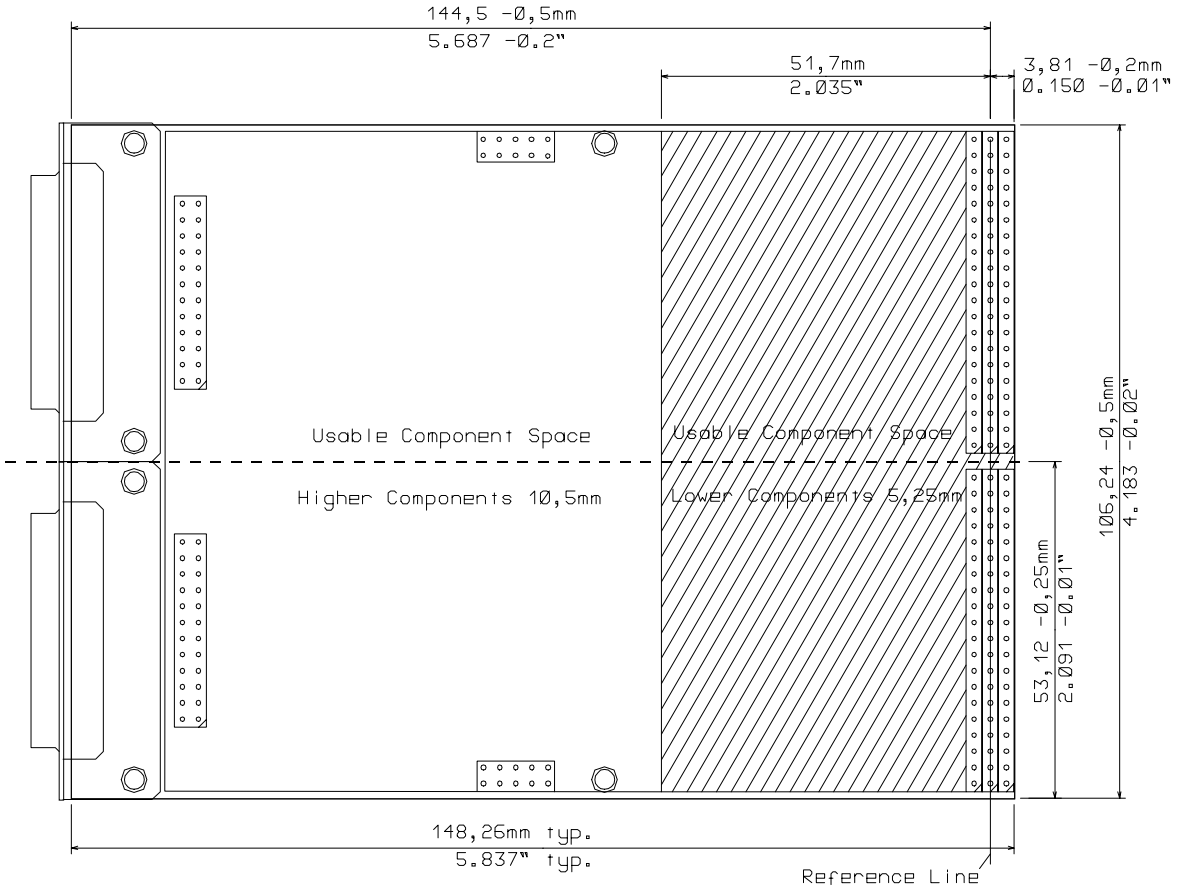


Figure 21 - Dimensions of a double MA-Module

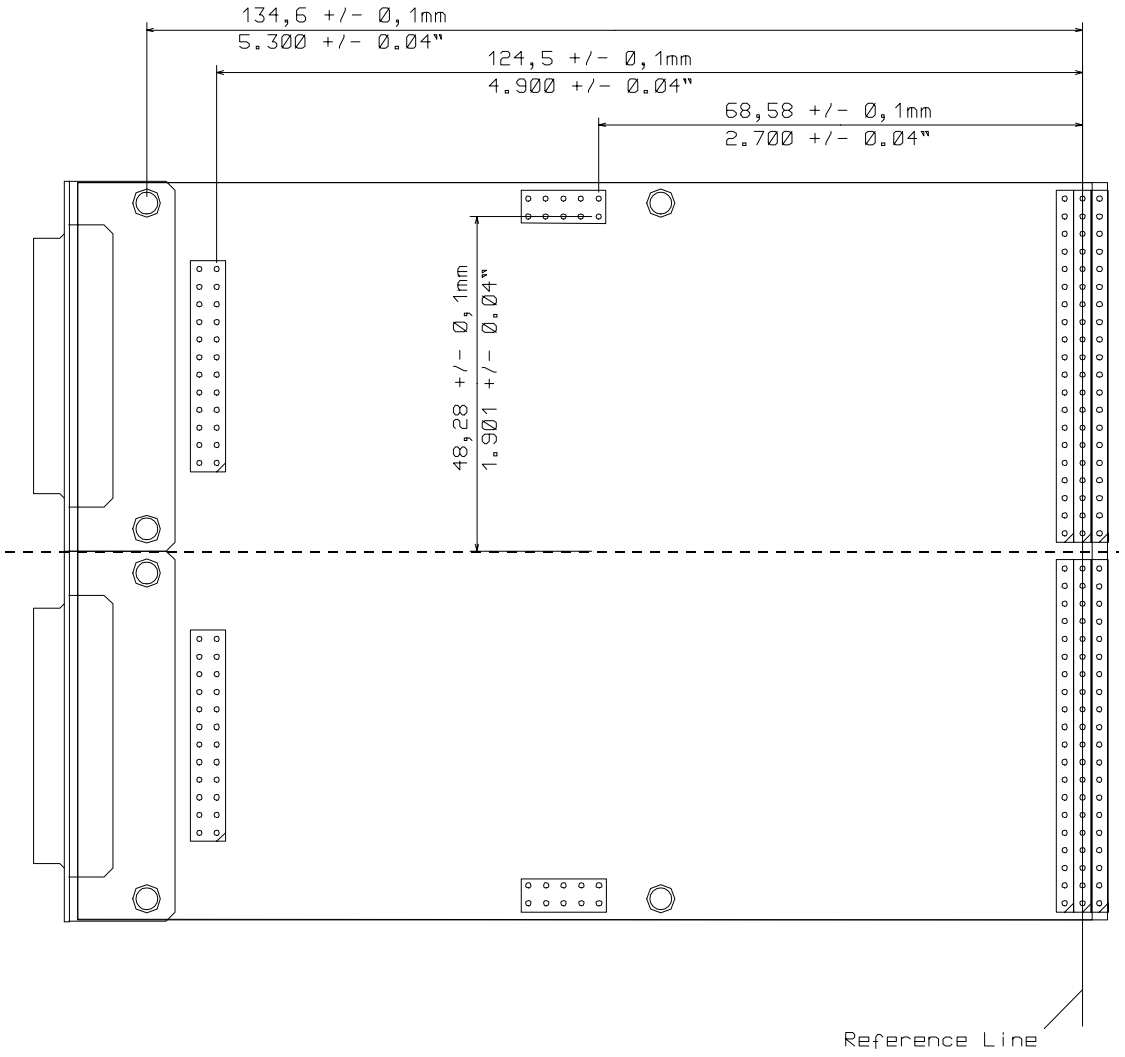


Figure 22 - Position of connectors

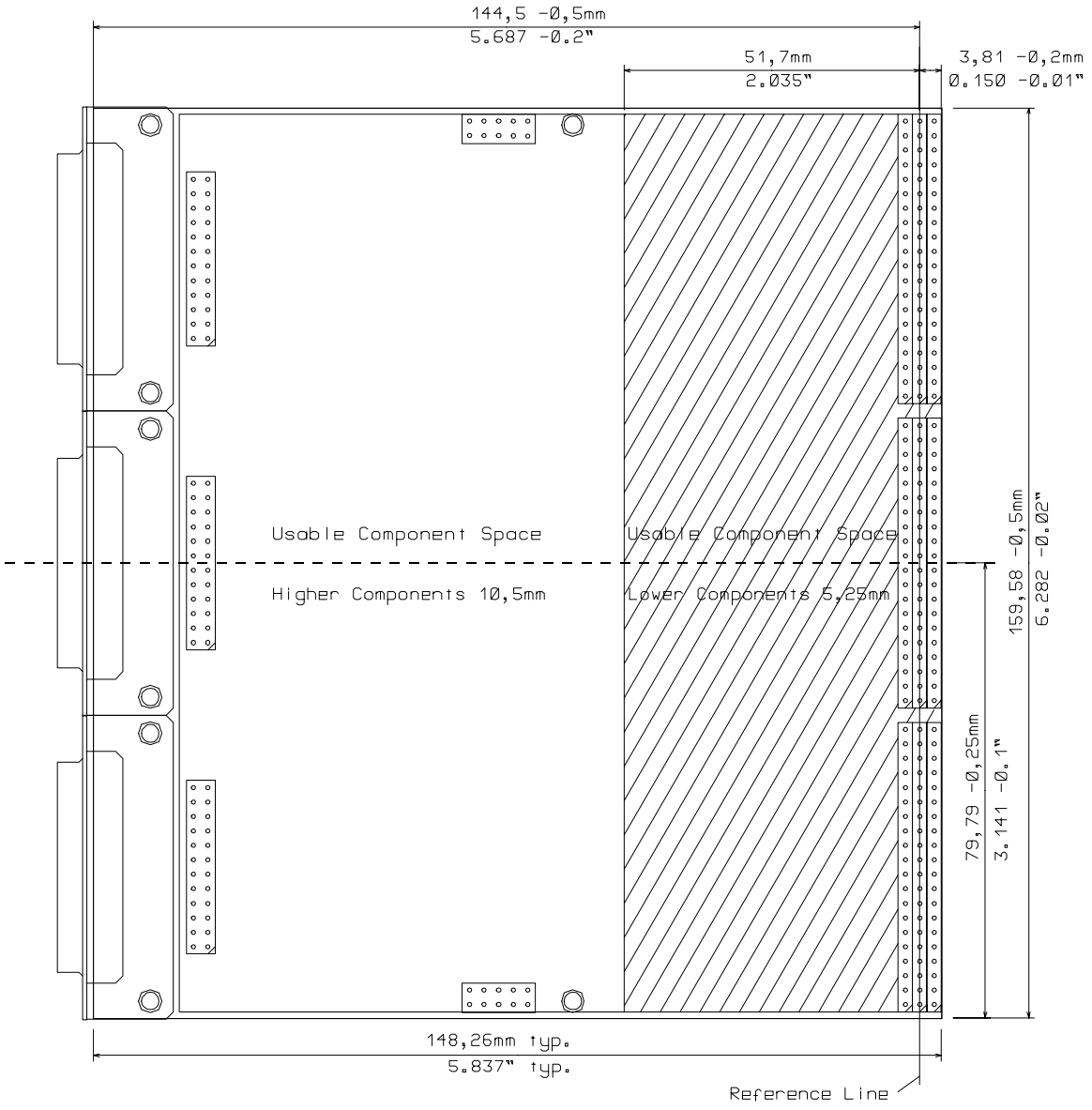


Figure 23 - Dimensions of a triple MA-Module

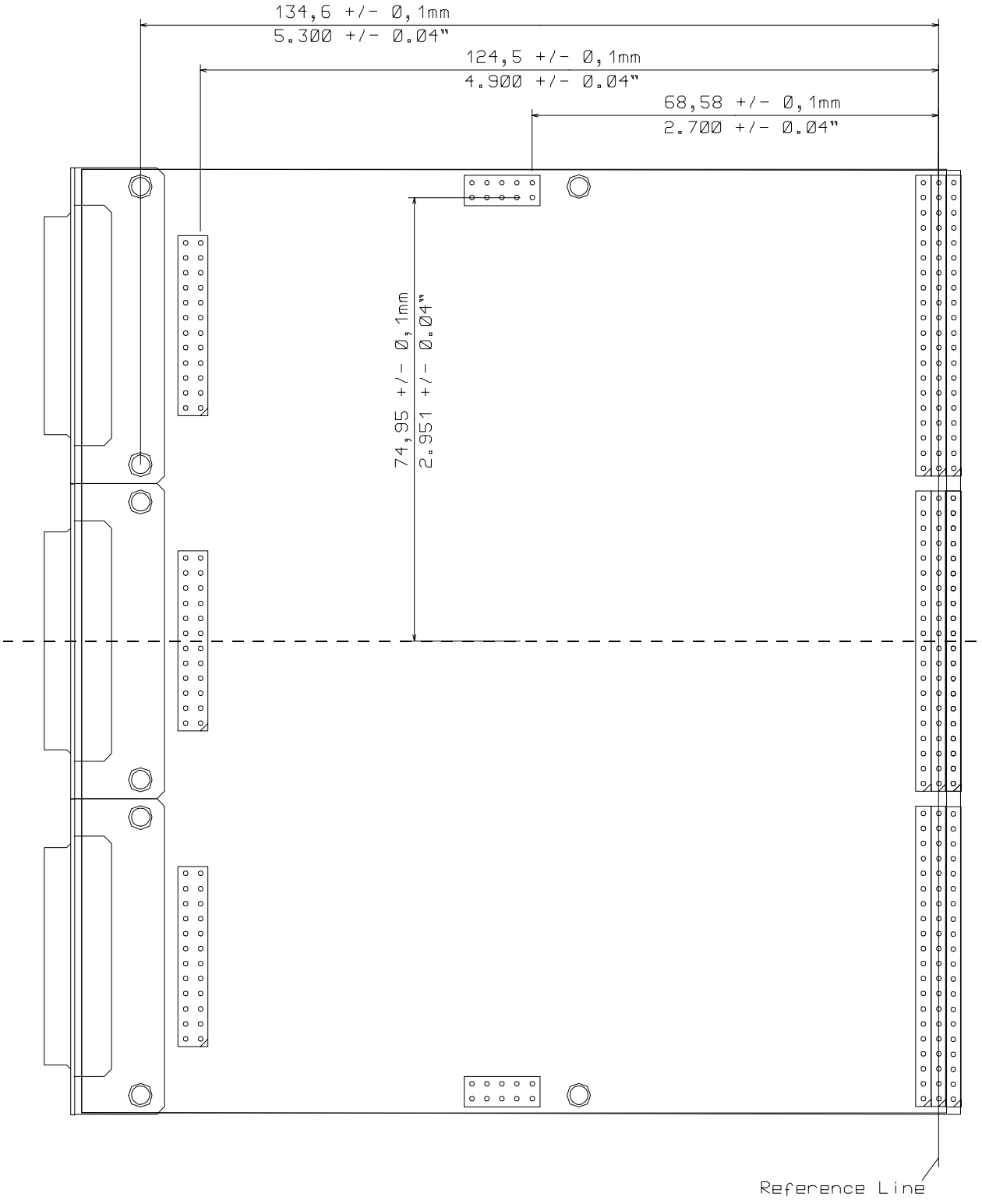


Figure 24 - Position of connectors

On the reverse side, the components may project by a maximum of **1.2mm** over the surface of the M-Module printed circuit board.

6.2 Front panel of base boards

If the M-Module is mounted on a base board, it presumes that there is a gap for the front panel connector conforming to the following figure.

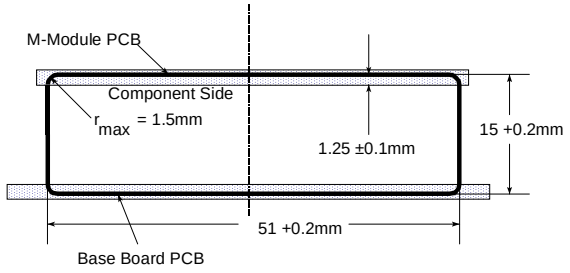


Figure 25 - M-Module front

6.3 Connectors used

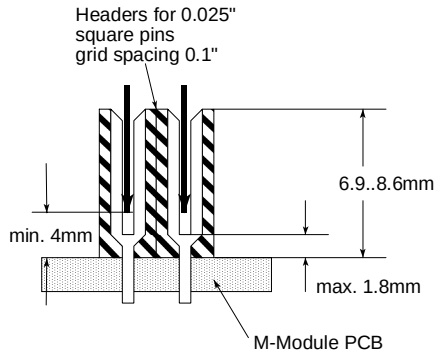


Figure 26 - M-Module connectors

The connectors listed must correspond to figure 26[1)].

6.4 Mounting the M-Module on the base board

Four screws (DIN 85-M3x8) should be used with the module's hexagonal bolts for attaching the module to the printed circuit board.

Recommendation

Use the bolts to contact protective ground to the module to meet EMC requirements.

6.5 Resistance to mechanical shock and vibration

The modules must be able to withstand vibration of **1g** in the range **10..2000Hz** (2 cycles, duration: 30mn) according to IEC 68-2-6.

They must be able to withstand bumps of at least **25g** for **6ms** (half-sine) according to IEC 68-2-29.

They must be able to withstand a mechanical shock of at least **40g** for **6ms** (half-sine) according to IEC 68-2-27.

7 Thermal specification

The maximum power dissipation of a module must not exceed **10W**. Power may be supplied via the base board or via the peripheral connector.

All M-Modules must be fully operational at a temperature range of **at least 0..60°C**. The permissible storage temperature range must be **-20..+70°C** or greater. Naturally, extended temperature ranges are conceivable for special operating conditions.

The air flow for cooling the module must be at least **400l per hour**. The air temperature of the cooling air must not exceed the maximum ambient temperature of the module.

Recommendation

CMOS components with low power dissipation should be given preference for use on M-Modules.

The standard humidity class according to DIN 40040 for all modules is: **F, non-condensing**.

---

1) Example of base board connection:
40-pin connectors that can be used for interface to the base board are, e. g.,
- BL6-40-Z (Fischer-Metroplast);
- AMP HV-100 series.

24-pin receptacle connectors that can be used for alternative peripheral connection via the base board are, e. g.,
- BL6-24-Z (Fischer-Metroplast);
- AMP HV-100 series.

Example of peripheral connection:
25-pin D-Sub connector for peripheral connection:
- ZDF-25ABZKUNG (receptacle) Conec;
- ZDS-25ABZKUNG (plug) Conec.

These connectors have round screw bolts attached by rivet for strain relief.
10-pin receptacle connectors that can be used for intermodule connection are, e.g., MK2G-10 (Fischer-Metroplast) or compatible connectors.
10-pin ribbon cable plug connectors that can be used for the connecting cable of the intermodule connection are, e.g., the Scotchflex printed circuit board connector 3910-2000T3M or compatible connectors.

8 Extended electrical specification

8.1 Introduction

Four main requirements led to this extension to the M-Module specification:

- extension of the address bus to a maximum of 24 bits, allowing up to 16 Mbytes to be addressed;

- extension of the data bus width to 32 bits;

- provision for trigger signals for measurement applications, e.g. for use in VXIbus systems;

- burst access for higher data rates.

Modules with these capabilities need additional signals for communication with the base board and this requires additional pins. M-Modules with a 3-row, 60-pin base-board interface connection are termed MA-Modules. The design of the base board enables M-Modules and MA-Modules to be mixed. This presents no problems if certain restrictions are borne in mind. Base boards with only a 2-row, 40-pin connection normally cannot utilize all features of an MA-Module (trigger features for instance). Nevertheless, it is often possible to operate an MA-Module on a base board for M-Modules. At all events, any M-Module or MA-Module can be operated on base boards designed for MA-Modules.

8.1.1 Extended address space

The extended address space enables M-Modules to be used for applications extending beyond typical I/O functions. The address space can be extended so as to have M-Modules with a 24-bit address bus. When choosing a base board it is merely necessary to ensure that the address space of the base board is at least as large as the address space of the M-Module. If not, there may be restrictions on the operation of the module, but such a combination may be justified for some modules. If the base board is not able to supply all 23 lines for a 24-bit address bus, the unsupplied lines shall be driven to "low".

The address information is transferred across the data bus in multiplexed mode, reducing the number of pins required. An additional line - /AS - indicates the current use of the bus. The normal M-Module store cycle is embedded in the address transfer cycle. The high-order (most significant) address information is transferred at the beginning of a cycle while /AS is at "H" potential. The module can store this information at the falling edge.

During the following data transfer cycle (which is completely compatible with the basic electrical specification), /AS functions as an address modifier. If /AS is "H" the access is what is now termed an I/O access, using the 256-byte address space. However, if /AS is "L" - i.e. a falling edge has occurred and the high-order part of the address has been transferred across the data bus - it is what is now termed a memory access.

This permits construction of modules with a 16-Mbyte memory area and a 256-byte I/O area.

One such application could be a graphics module, where any base board - including base boards with an 8-bit address bus - can access the I/O area and a base board with a 24-bit address bus can also access video memory directly.

8.1.2 32-bit data bus

The data bus extension is implemented using 16 pins from the third row. This true parallel 32-bit data bus allows M-Modules to be optimized for maximum data transfer rates.

An additional control line is used to provide the necessary dynamic bus sizing. This permits 32-bit M-Modules to be used on a base board with a 16-bit data bus. The dynamic bus sizing is also compatible with dynamic buses such as the VMEbus or the bus used by Intel's 80486 microprocessor.

The extension is primarily aimed at increasing data throughput. The increase in the width of the data bus can be seen as completely independent of the address bus extension (for instance for controlling fast FIFOs with a width of 32 bits, which do not require an extended address space).

8.1.3 Burst access

Burst access serves for fast data exchange between the base board and an M-Module. Burst access can be applied with all addressing modes (A08 and A24).

8.1.4 Trigger lines

The universal trigger lines can be used for any kind of synchronization between M-Modules or for synchronization of M-Modules with the base board.

Two pins are provided for this purpose. They must always be inputs following M-Module reset in order to guarantee compatibility between different M-Modules and base boards.

Typical applications are multichannel analog-to-digital converters which must sample at exactly the same time.

8.2 Communication between M-Module and base board

8.2.1 Signals

D00/A08..D15/A23 — Multiplexed data/address bus. When line /AS is at "H" potential, the most significant bits of the address can be transmitted using these lines. They must be stored by the module at the falling edge of /AS. Thereafter these lines form the bidirectional data bus as defined in the basic electrical specification.

D16..D31 — Most significant portion of the bidirectional data bus. D16 is the least significant data bit and D31 is the most significant bit. It should be stressed here that this does not specify any byte order, merely a bit order and an identification of the bytes.

If the M-Module has only an 8-bit data bus, only D00..D07 shall be used, a 16-bit module must use D00..D15.

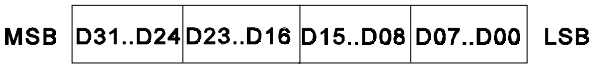


Figure 27 - Bit order

/AS — Input signal. Address strobe. Indicates the meaning of the signals on the multiplexed address/data bus. If /AS is "H", address information can be transmitted via lines D00/A08..D15/A23. The M-Module must store the address at the falling edge. While /CS is asserted ("L"), /AS functions as an address modifier: if /AS is "L", the access is to the extended address range; if /AS is "H", an access in I/O mode is performed.

/DS0..2 — Input signals. They are valid while /CS, /IACK or /DACK is active. Together with A1 the three data-select lines specify whether the access is a byte, 2-byte or 4-byte access. Also, for byte access or 2-byte access the data routing, i.e. the active part of the data bus and the data flow, is specified. The following figure and table show the meaning of the signals.

TRIGA,TRIGB — Trigger inputs/outputs. After power-up these lines must always be inputs. During operation they can be programmed under software control as outputs. It is possible to trigger specific actions on the M-Modules at the positive edge. If only one trigger line is used it must be TRIGA.

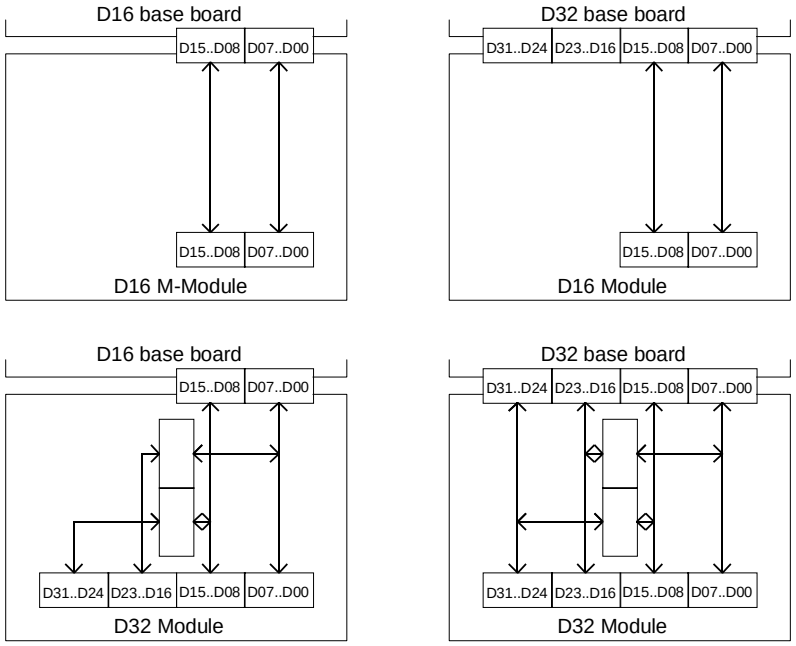


Figure 28 - Data bus routing

Table 7 - Data bus routing

/DS2	/DS1	/DS0	A1	M-Module interface	D32 M-Module internal	D16 M-Module internal	
H	H	H	L	-	-	-	
H	H	L	L	D00..D07	D16..D23	D00..D07	
H	L	H	L	D08..D15	D24..D31	D08..D15	D16/
H	L	L	L	D00..D15	D16..D31	D00..D15	D32
H	H	H	H	-	-	-	base
H	H	L	H	D00..D07	D00..D07	D00..D07	board
H	L	H	H	D08..D15	D08..D15	D08..D15	
H	L	L	H	D00..D15	D00..D15	D00..D15	
L	H	H	L	D00..D31	D00..D31	-	
L	H	L	L	D16..D23	D16..D23	-	
L	L	H	L	D24..D31	D24..D31	-	D32
L	L	L	L	D16..D31	D16..D31	-	base
L	H	H	H	D00..D31	D00..D31	-	board
L	H	L	H	D00..D07	D00..D07	D00..D07	
L	L	H	H	D08..D15	D08..D15	D08..D15	
L	L	L	H	D00..D15	D00..D15	D00..D15	

8.2.2 Principles of operation

8.2.2.1 Read/write access - no burst

The normal M-Module access cycle is embedded in the address transfer cycle. At the beginning of one of these accesses, while /AS is at "H" potential, the most significant address information is transferred. On the falling edge the module can store the information. The address information is transferred over the data bus in multiplexed mode.

The actual access cycle is in accordance with the basic

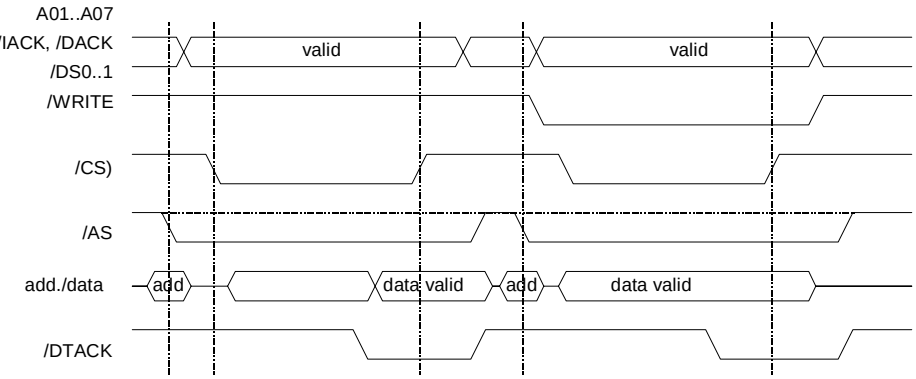


Figure 29 - Signal sequence of a read/write access

electrical specification.

8.2.2.2 Burst access

A burst transfer cycle starts with a normal access cycle (read or write) involving both address and data transfers. This first cycle is followed by an arbitrary number of cycles involving data transfers only.

An M-Module with burst mode capability must implement an internal address generator which is loaded with the start address during the first (address/data cycle) and must generate follow-on addresses for the data cycles in the remainder of a burst transfer. This address generator must at least provide two features: (1) linear up and down counting and (2) wrap around modulo a fixed length of the address range covered. It may of course provide additional features such as programmable address range, more sophisticated address sequencing such as "reverse carry" etc.

The first cycle of a burst transfer must always be an A24 access differing in no aspect from a normal (non-burst) access. The module has to pick up the start address combined from A1..A7 and multiplexed A8..A23 in a look-ahead fashion, regardless if the following cycle continues with data bursts or not. The /AS signal is kept low after the address/data cycle and during all data cycles in the burst. This can easily be decoded by the module for internal control of burst transfers.

Data transfers within a burst are handshaked by /CS (or linked to that, change of data) and /DTACK. Suitable timing of the leading (falling) edge of /DTACK ensures that the module's access time is met. Timing of the trailing (rising) edge forces the base board not to start with the next cycle before the module has completed the on-going transfer. For that reason, the only extension of timing specifications is that in consecutive cycles during a burst, /CS must go low 0ns after /DTACK has been de-asserted at the earliest. All other timing relationships are retained and can be taken over from the basic electrical specification.

The figure below illustrates a burst transfer sequence for write transfers. Read transfers are similar, the timing can be derived from the basic electrical specification. The data direction cannot change during a burst. The status of the /WRITE signal can be latched in the first cycle.

8.2.3 Driver characteristics

Table 8 - Signal name and type

Signal name	Type	
A01..A07	address	A
A08/D00..A23/D16	TS input/output	A
A08..A23 D00..D31	D16..D31 addr data	A
/DS2	input	B*
/AS	input	B*
/CS	input	B*
/TRIG0..TRIG1	input	B*
Note /DTACK		

*: These input lines must be terminated with a 10-KΩ pull-up resistance (after VCC). This ensures that the modules will operate with a three-row receptacle connector on two-row base boards.

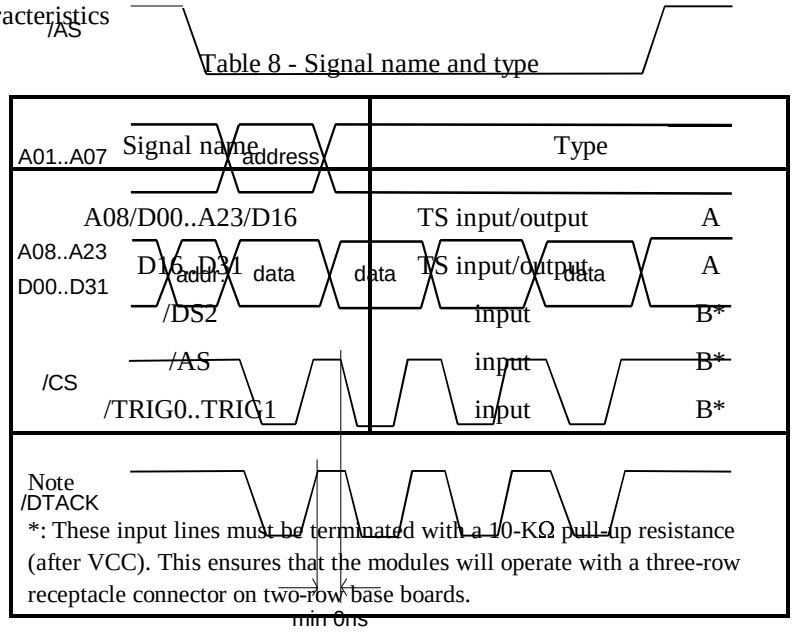


Figure 30 - Burst transfer sequence for write transfers

The electrical specification of the drivers over the whole temperature range is given in table 3 in the basic electrical specification.

The capacitive load of a pin must not exceed 25pF.

8.2.4 Timing specifications

Regardless of the minimum specification, the appropriate data sheets for the modules must be compared in order to ensure that the maximum trigger rates and maximum frequencies of modules and base boards are compatible.

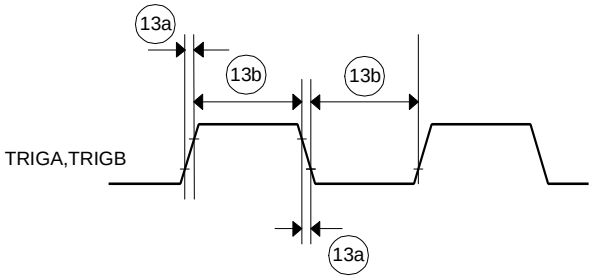
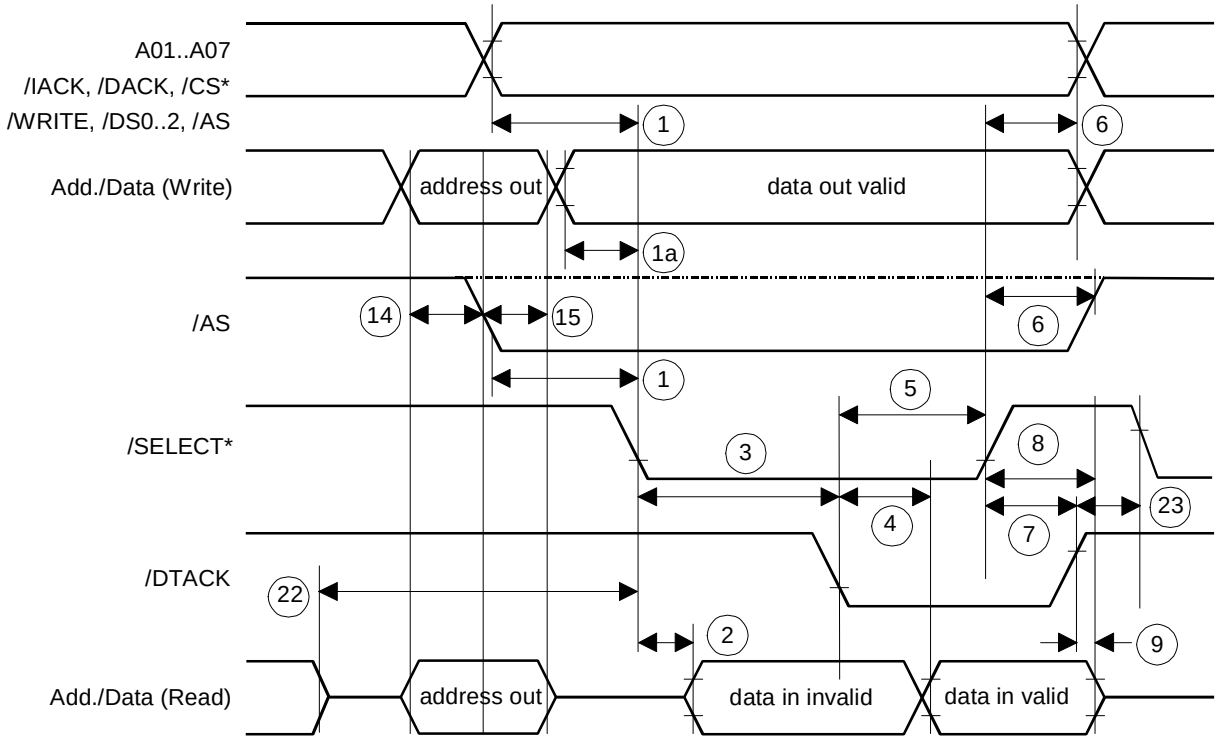


Figure 31 - TRIGA and TRIGB timing



NOTE

*: "/SELECT" represents one of the three signals /CS, /DACK or /IACK. Only one of these (designated "/SELECT") is asserted ("L") during a cycle; the others are "H" as indicated in the following table.

Cycle type	/CS	/DACK	/IACK
Normal access	L	H	H
DMA access	H	L	H
Interrupt acknowledge	H	H	L

Figure 32 - Read/write timing

Table 9 - Timing parameters

Num	Characteristic	Min	Max	Unit
13a	TRIGA, TRIGB rise and fall time	-	20	ns
13b	TRIGA, TRIGB hold time	7	-	ns
14	Address setup	10	-	ns
15	Address hold	10	-	ns

8.2.5 Pin assignment

The control interface between the base board and the module is via three-row plug connectors (pin headers) on the base board, which correspond to receptacles on the module.

Rows A and B are in accordance with the pin assignments described in the basic electrical specification. Row C is additional.

Table 10 - Pin assignment

Pin	Row A	Row B	Row C
1	/CS	GND	/AS
2	A01	+5V	D16
3	A02	+12V	D17
4	A03	-12V	D18
5	A04	GND	D19
6	A05	/DREQ	D20
7	A06	/DACK	D21
8	A07	GND	D22
9	D08/A16	D00/A08	TRIGA
10	D09/A17	D01/A09	TRIGB
11	D10/A18	D02/A10	D23
12	D11/A19	D03/A11	D24
13	D12/A20	D04/A12	D25
14	D13/A21	D05/A13	D26
15	D14/A22	D06/A14	D27
16	D15/A23	D07/A15	D28
17	/DS1	/DS0	D29
18	/DTACK	/WRITE	D30
19	/IACK	/IRQ	D31
20	/RESET	SYSCLK	/DS2

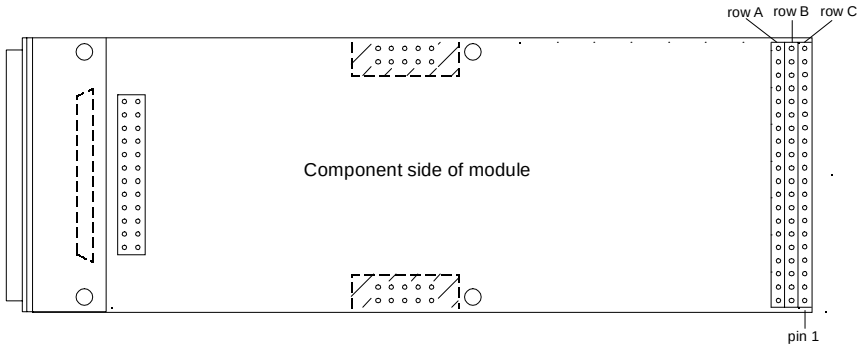


Figure 33 - Orientation of the receptacle connector on the module

Annex A

(informative)

Driver software interface

Revision 1

A.1 Introduction

This annex specifies a software interface for M-Modules in accordance with the hardware specifications.

The interface has been designed in such a way that all hardware functions of an M-Module can be controlled by means of eight defined driver functions (plus an interrupt handler). In this specification the functions of this interface are referred to as low-level driver (LL driver).

Usually, a higher software layer is added to the LL driver. In this specification the functions of this layer are referred to as high-level driver (HL driver).

By means of pre-defined service functions that must be provided by an HL driver, it is possible to trigger the HL driver functions from the LL driver. This includes system-specific functions and buffer management functions.

The interface described here has been implemented in the "C" programming language.

The LL driver can be used as follows:

- as an LL driver: part of an MDIS driver;
- as an LL driver: part of one's own driver developments;
- as a function library: direct hardware access from an application.

A.2 Definitions

A.2.1 Device

A device is a physical unit (M-Module). Each device is described by a structure, the descriptor structure. This structure contains all the physical and logical parameters of the device.

Within a system, a device is uniquely identified by its base address, i.e. there must not be more than one device with this base address.

A.2.2 Channel

An M-Module is a device with one or more physical channels. Each channel is characterized by a specific channel width (in bits), transmission direction, and type. The number of channels of a device is limited to 256.

A.3 Remarks concerning the high-level driver

A.3.1 Functions

The definition of the LL interface is based on the following proposed HL driver functions:

- system initialization and interrupt installation;
- device initialization and de-initialization;
- read and write accesses to the device;
- allocation and management of an input buffer or output buffer or both;
- read and write accesses to the input/output buffers;
- handling of events via signals.

The above-mentioned functions are only a proposal. Apart from the defined service functions, implementation of the HL driver is left completely to the user.

A.3.2 Descriptor structure

Each device driver must be able to operate one or several devices of the same kind. For this purpose, each device is referenced by means of a structure referred to here as a descriptor structure. The descriptor structure contains all physical and logical attributes of a device (e.g. number and description of channels). Additional data structures required by the HL driver to manage the device may be part of the descriptor structure.

The descriptor structure is managed by the HL driver and passed to the LL functions as a pointer.

Some elements of the descriptor structure are available to the LL driver for read or write access. Accessing those elements must be done by means of macros (see A.7). Therefore, the descriptor structure must contain at least the elements which are available to the LL driver.

The exact descriptor structure depends on the implementation of the HL driver and is not part of this specification.

A.3.3 The device options field

Some LL drivers require an additional data area for storing global device data, referred to here as the device options field.

To access this area, the **OPT_PTR** is provided for in the descriptor structure. It can point to any kind of data block, the size of which is defined in **OPT_LEN**.

The HL driver must allocate a memory area of the size determined and enter the starting address in the descriptor parameter (**OPT_PTR**). The device options field must be allocated separately for each device.

The LL driver does not check whether or not a device options field exists.

A.3.4 I/O buffer management

The HL driver has the task of allocating, providing and managing an input buffer or output buffer or both. It provides the LL driver with one service function each for requesting and terminating buffer entries. When requesting, the LL driver is supplied with a buffer entry. The termination function signals the HL driver that the buffer entry is terminated. Then the HL driver can update its buffer pointers and wake up waiting processes.

A.3.5 Handling of events

An LL driver can support up to 4 events, referred to here as signal conditions. These are used for signaling a process (user program) via a software interrupt that certain events are occurring on the device.

When a signal condition is activated or de-activated by the HL driver, this is signaled to the LL driver by the setstat routine being called. When activating (code: SIG_set_cond?), the driver must initialize the hardware in such a way that the signal condition can trigger an interrupt. When de-activating (code: SIG_clr_cond?), the driver must initialize the hardware in such a way that the signal condition cannot trigger an interrupt.

The HL driver provides the LL driver with a service function for handling the signal conditions. If such an event occurs in the form of an interrupt, the LL driver must signal this event to the HL driver by calling the **M_sig_cond()** service function. This makes it possible for the HL driver to trigger a software interrupt, e.g. in the form of a signal.

A.3.6 Interface to the LL driver

The interface to the LL driver is designed in such a way that this driver only has to carry out basic input or output actions to a device, e.g. "read channel 2 from device to base address 0xffe00000". The LL driver is freed from any other tasks, such as system initialization, interrupt installation, buffer management, memory allocation (buffer), etc.

The HL driver ensures that all function parameters passed to the LL driver are valid. The LL routines in turn must terminate with defined return values in order to signal the HL driver whether an action has been successful or not.

A.4 Driver functions of the low-level driver

A.4.1 Survey

An LL driver has the following entry points (functions):

- **<dev>_init()**
  initialize device;
- **<dev>_exit()**
  de-initialize device;
- **<dev>_read()**
  read from one channel of the device;
- **<dev>_write()** write to one channel of the device;
- **<dev>_block_read()** read all (input) channels of the device;
- **<dev>_block_write()** write all (output) channels of the device;
- **<dev>_setstat()** set device parameters;
- **<dev>_getstat()** get device parameters;
- **<dev>_irq_c()** interrupt handler.

<dev> is the device name (e.g. device "xyz": xyz_init(), xyz_exit()).

A.4.2 Device initialization

Table A.1 - Device initialization <dev>_init()

Call	Bit32 **<dev>_init(mod_ptr,ch,base,mode)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 ch	channel number (0..255)
	Bit32 base	base address
	Bit32 mode	(not used)
Returns	0	if OK
	err(code)	if an error occurred

The init routine is called by the HL driver for initialization of the device. System initialization must be carried out beforehand. In addition, the device options field must be available if required.

The routine has the task of putting the device with the base address **base** into a defined initial condition, i.e. carrying out a hardware reset on all channels. The **ch** parameter defines the channel selected after the reset. In most cases, this parameter can be ignored.

Possible errors:

ERR_Init	general initialization error
ERR_IllParam	illegal parameter

A.4.3 Device de-initialization

Table A.2 - Device de-initialization - <dev>_exit()

Call	Bit32 **<dev>_exit(mod_ptr,ch,base,mode)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 ch	channel number (0..255)
	Bit32 base	base address
	Bit32 mode	(not used)
Return	0	if OK
	err(code)	if an error occurred

The exit routine is called by the HL driver in order to de-initialize the device. System de-initialization may be carried out only after this.

This routine has the task of de-initializing the device with the base address **base** as far as this is necessary. The **ch** parameter defines the last channel selected. In most cases it can be ignored.

Possible errors:

ERR_Exit	general termination error

A.4.4 Read from device

Table A.3 - Read from device - <dev>_read()

Call	Bit32 **<dev>_read(mod_ptr,ch,base,value)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 ch	channel number (0..255)
	Bit32 base	base address
	Bit32 *value	pointer to read value
Returns	0	if OK
	err(code)	if an error occurred

The HL driver can jump to the read routine if channel **ch** is defined as an input or input/output channel.

This routine has the task of reading the channel **ch** of the device (with the base address **base**) and returning the value obtained via the ***value** pointer.

Possible errors:

ERR_Read general read error

A.4.5 Write to device

Table A.4 - Write to device - <dev>_write()

Call	Bit32 **<dev>_write(mod_ptr,ch,base,value)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 ch	channel number (0..255)
	Bit32 base	base address
	Bit32 value	value to be written
Returns	0	if OK
	err(code)	if an error occurred

The HL driver can jump to the write routine if channel **ch** is defined as an output or input/output channel.

This routine has the task of writing the value **value** to channel **ch** of the device (with the base address **base**).

Possible errors:

ERR_Write general write error

A.4.6 Set device status

Table A.5 - Set device status - <dev>_setstat()

Call	Bit32 **<dev>_setstat(mod_ptr,base,code,value)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 base	base address
	Bit32 code	status code
	Bit32 value	value to be set
Returns	0	if OK
	err(code)	if an error occurred

The setstat routine can be used universally by the HL driver for setting device-specific parameters or for triggering device-specific actions.

This routine has the task of setting a hardware-specific parameter **code** to the value **value**.

In addition the setstat routine must handle the following pre-defined status codes:

Table A.6 - Pre-defined status codes

Status code "code"	Description of LL driver action	Range of "value"
IRQ_enable	enable/disable interrupts on the module	0=disable, 1=enable
SIG_set_cond1..4	activate module for catching condition 1..4	(unused)
SIG_clr_cond1..4	de-activate module for catching condition 1..4	(unused)
CH_option	handle change of channel option	0..0xffff

If an interrupt is installed on a device, it can be enabled or disabled immediately by calling the setstat routine with the appropriate **IRQ_enable** status code.

If a driver supports one or several signal conditions, the corresponding status codes **SIG_set_cond1..4** and **SIG_clr_cond1..4** must also be handled.

If a driver uses the channel option, the corresponding status code **CH_option** must also be handled.

If the channel number is needed, it can be obtained by means of **CH_CURRENT**. Unknown or unsupported status codes result in the **ERR_UnkCode** error code being returned.

Possible errors:

ERR_UnkCode unknown or unsupported status codes

ERR_IllParam illegal parameter

A.4.7 Get device status

Table A.7 - Get device status - <dev>_getstat()

Call	Bit32 **<dev>_getstat(mod_ptr,base,code,value)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 base	base address
	Bit32 code	status code
	Bit32 *value	pointer to obtained value
Returns	0	if OK
	err(code)	if an error occurred

The getstat routine can be used universally by the HL driver for obtaining device-specific parameters.

It serves to determine a hardware-specific **code** parameter and return the obtained value via the ***value** pointer.

There are no pre-defined status codes.

If the channel number is required, it can be obtained by means of **CH_CURRENT**. Unknown or unsupported status codes result in the **ERR_UnkCode** error code being returned.

Possible errors:

ERR_UnkCode unknown status code or status code that is not supported

A.4.8 Block read from device

Table A.8 - Block read from device - <dev>_block_read()

Call	Bit32 **<dev>_block_read(mod_ptr,base,buf)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 base	base address
	void *buf	pointer to buffer
Returns	>=0	number of read bytes
	err(code)	if an error occurred

The block read routine can be used both by the HL driver and by the interrupt routine of the LL driver, to read all channels simultaneously into a provided buffer.

The routine has the task of reading all input channels of the device in ascending order and writing them into the buffer **buf**, whereby the conventions regarding buffer organization must be taken into account, i.e. the width of each (channel) value must be identical with the channel width **CH_WIDTH** of 1, 2 or 4 bytes defined in the descriptor structure. The number of input channels must correspond to the number in **CH_INCNT**.

Thus a maximum of (**CH_WIDTH** * **CH_INCNT**) bytes may be written into the buffer. The function returns the number of read bytes (i.e. bytes written to the buffer).

Possible errors:

ERR_Read general read error

A.4.9 Block write to device

Table A.9 - Block write to device - <dev>_block_write()

Call	Bit32 **<dev>_block_write(mod_ptr,base,buf)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 base	base address
	void *buf	pointer to buffer
Returns	>=0	number of written bytes
	err(code)	if an error occurred

The block write routine can be used both by the HL driver and by the interrupt routine of the LL driver, to write values from a provided buffer to all channels.

The routine has the task of writing the values from the buffer **buf** to all output channels of the device in ascending order. The buffer is organized in accordance with the conventions concerning buffer organization, i.e. the width of each (channel) value from the buffer is the channel width **CH_WIDTH** of 1, 2 or 4 bytes defined in the descriptor. The number of channel values is identical with the number of output channels **CH_OUTCNT**.

Thus a maximum of (**CH_WIDTH** * **CH_OUTCNT**) bytes may be taken from the buffer. The function returns the number of written bytes (i.e. bytes taken from the buffer).

Possible errors:

ERR__Write general write error

A.4.10 Interrupt routine

Table A.10 - Interrupt routine - <dev>_irq_c()

Call	void **<dev>_irq_c(mod_ptr,base)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 base	base address
Return	-	

The HL driver installs the interrupt routine in the system and jumps to it only when a hardware interrupt is triggered.

It can fulfill one or several of the following tasks:

- interrupt-triggered reading from the device: writing to the input buffer;

- interrupt-triggered writing to the device: output of values from the output buffer;

- handling of events: transmitting signals to a user process.

For reading or writing data via the input/output buffers the LL functions **<dev>_block_read()** or **<dev>_block_write()** can be used. Transmission of signals is done via the HL function **M_sig_cond()**.

The function of the interrupt routine (reading, writing or transmitting a signal) depends on the device:

a) **Reading** all channels (into the input buffer):

For this purpose, the interrupt routine must obtain the target address by means of the HL function **M_get_inbuf()** (i.e. the entry to be initialized in the input buffer). Only if this address is not ZERO can data be written to it by calling **<dev>_block_read()**. Then the HL driver must be signaled via **M_rdy_inbuf()** that the block transfer has been completed.

```
if ((buf = M_get_inbuf(mod_ptr)) != NULL)          /* get destination address */
{
      <dev>_block_read(mod_ptr, base, buf);        /* block read to destination */
      M_rdy_inbuf(mod_ptr);                        /* block read ready */
}
```

Figure A.1 - Reading all channels

b) **Writing** all channels (from the output buffer):

For this purpose the interrupt routine must obtain the source address by means of the HL function **M_get_outbuf()** (i.e. the entry to be used in the output buffer). Only if this address is not ZERO can data be read from it by calling **<dev>_block_write()**. Then the HL driver must be signaled via **M_rdy_outbuf()** that the block transfer has been completed.

```
if ((buf = M_get_outbuf(mod_ptr)) != NULL)         /* get source address */
{
      <dev>_block_write(mod_ptr, base, buf);       /* block write from source */
      M_rdy_outbuf(mod_ptr);                       /* block write ready */
}
```

Figure A.2 - Writing all channels

c) **Transmitting** a signal (signal condition) to the user process:

For this purpose the interrupt routine must call the HL function **M_sig_cond()**, stating the signal condition (1..4). This makes it possible for the HL driver to transmit a signal to the user process.

```
if ( condition )              /* condition 2 occurred ? */
  M_sig_cond(mod_ptr,2);      /* handle signal condition 2 (send signal) */
```

Figure A.3 - Transmitting a signal

If several read accesses to the same hardware address are made one after the other in the interrupt routine, the HL function **FlushCache()** must be called before each access. If the used processor type has a read cache, this function effects a clearing of the cache.

A.4.11 Referencing the functions

The following table shows the conditions under which certain LL functions may be called by the HL driver:

Table A.11 - Function reference

	init	exit	read	write	getstat	setstat	block read	block write	irq_c
device is initialized	*	-	-	-	-	-	-	-	-
device is de-initialized	-	*	-	-	-	-	-	-	-
device is already initialized	-	-	*	*	*	*	*	*	*
channel supports input	-	-	*	-	-	-	-	-	-
channel supports output	-	-	-	*	-	-	-	-	-
input buffer exists	-	-	-	-	-	-	*	-	-
output buffer exists	-	-	-	-	-	-	-	*	-
interrupt is installed	-	-	-	-	-	*1)	-	-	*
interrupt is enabled	-	-	-	-	-	-	-	-	*

Note:

1) concerns only the "IRQ_enable" status code

Example:

The **read** routine of the LL driver may only be called if the device is initialized and the channel to be read is an input or input/output channel.

A.4.12 Error handling

In order to enable the HL driver to convert error codes, these must be returned via the HL function **err()**.

Example:

error "ERR_Read" occurred:

*return(**err(**ERR_Read**)**);*

A.5 Service functions of the high-level driver

The HL driver must provide the following service functions for the LL driver:

A.5.1 Request buffer entry

Table A.12 - Request buffer entry - M_get_in/outbuf

Calls	void ***M_get_inbuf(mod_ptr)**	request input buffer
	void ***M_get_outbuf(mod_ptr)**	request output buffer
Parameter	MODUL_DATA *mod_ptr	pointer to descriptor structure
Returns	>0	address of the next, free entry in the input buffer (M_get_inbuf) or the next output buffer entry to be used (M_get_outbuf)
	NULL	no buffer entry available

M_get_in/outbuf() requests the use of the next entry of the input/output buffer from the HL buffer management.

The next buffer entry is returned and internal buffer pointers are incremented.

This function may only be applied in the interrupt routine.

One or more requests must be terminated by the M_rdy_in/outbuf() termination function.

If there is no buffer entry available, the function returns a zero pointer.

A.5.2 Terminate buffer entry

Table A.13 - Terminate buffer entry - M_rdy_in/outbuf

Calls	Bit32 **M_rdy_inbuf(mod_ptr)**	terminate input buffer (block read)
	Bit32 **M_rdy_outbuf(mod_ptr)**	terminate output buffer (block write)
Parameter	MODUL_DATA *mod_ptr	pointer to descriptor structure
Return	0	

M_rdy_in/outbuf() signals the HL buffer management that a block read in the input buffer or a block write from the output buffer is complete.

The M_rdy_inbuf() function checks for a process waiting for data to be read. If the requested data is available in the input buffer, it will be woken up.

The M_rdy_outbuf() function checks for a process waiting for data to be written into the output buffer. If the buffer is empty enough to absorb the data, the process will be woken up.

The function may only be applied in the interrupt routine and only in connection with the buffer request M_get_in/outbuf().

A.5.3 Handle signal condition

Table A.14 - Handle signal condition - M_sig_cond

Calls	Bit32 **M_sig_cond(mod_ptr,i)**	
Parameters	MODUL_DATA *mod_ptr	pointer to descriptor structure
	Bit32 i	Condition (1..4)
Returns	0	OK, signal transmitted
	-1	signal condition is inactive
	-2	error during signal transmission

M_sig_cond() signals the HL driver that signal condition 'i' occurred. The HL driver can then transmit a signal to the user process.

If the signal condition has not been handled, the function returns with a value less than zero.

A.5.4 Delay

Table A.15 - Delay - Delay

Call	void **Delay(msec,mod_ptr)**	
Parameters	Bit32 msec	delay time [msec]
	MODUL_DATA *mod_ptr	pointer to descriptor structure
Returns	-	

Delay() waits for the stated time in milliseconds. If the system is not able to work with this resolution, the function appropriately rounds up the delay time.

A.5.5 Empty processor cache

Table A.16 - Empty processor cache - FlushCache

Call	void **FlushCache()**
Parameters	-
Returns	-

FlushCache() empties the read cache of the processor if one exists. This function is necessary for polling in interrupt routines.

A.5.6 Error return

Table A.16 - Error return - err

Call	Bit32 **err(code)**
Parameter	Bit32 code error code
Returns	(depends on driver mode)

This function must be used by LL drivers for return of errors. It enables the HL driver to convert error codes.

A.6 Conventions

A.6.1 Global variables

It is not permitted to use global variables in LL drivers, since this causes conflicts when several devices are controlled by the same LL driver.

If global variables are required, the device options field, which represents a buffer for global device parameters, must be used. This guarantees that there is an individual options field for each device and therefore individual global variables.

A.6.2 Types of data

Bit32 is a signed long word (32 bits)

Bit16 is a signed word (16 bits)

Bit8 is a signed byte (8 bits)

UnBit32 is an unsigned long word (32 bits)

UnBit16 is an unsigned word (16 bits)

UnBit8 is an unsigned byte (8 bits)

A.6.3 Hardware access

For hardware accesses, the **HwReg32**, **HwReg16** and **HwReg8** macros must be used:

HwReg32(addr) for accesses to a long-word register (32 bits) at address "addr"

HwReg16(addr) for accesses to a word register (16 bits) at address "addr"

HwReg8(addr) for accesses to a byte register (8 bits) at address "addr"

Example:

```
#define CTRL_REG HwReg8(base+0x01)

#define DATA_REG HwReg16(base+0x04)

ctrlbyte = *CTRL_REG;

dataword = *DATA_REG;
```

Use of the above-mentioned types guarantees that the driver remains portable in terms of hardware access. These macros are defined in an external include file depending on the operating system and bus architecture used.

A.6.4 Buffer structure

The input and output buffers are structured according to the following convention:

A buffer consists of **depth** entries (buffer depth). Each of these entries is **width** bytes wide (buffer width) and contains the values of **ch_cnt** channels in ascending order. All channels of an entry, in turn, have the same width **ch_wid** (channel width). The width can be one byte, word or long word and depends on the device.

A buffer starts at **bufptr** and has the size **size**. The buffer depth **depth** can be calculated from the buffer size and the buffer width **width**.

Example:

A device has 3 input channels (1..3) and 4 output channels (4..7). The channel width is 1 (byte). As a result, each entry of the input buffer has a width of 3 bytes and contains the values of the input channels in the order {1, 2, 3}. Therefore, each entry of the output buffer must have a width of 4 bytes and contain the values of the output channels in the order {4, 5, 6, 7}.

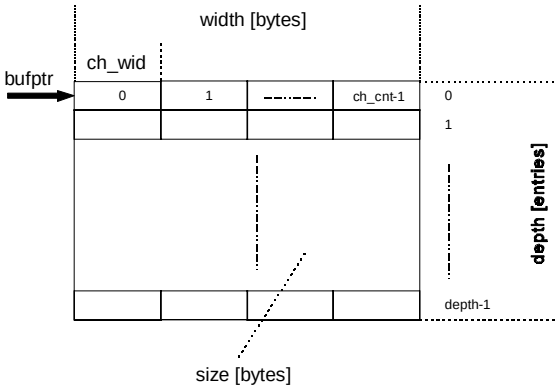


Figure A.4 - Buffer structure

Table A.18 - Calculation of buffer width and depth

	Input buffer	Output buffer
Width	CH_WIDTH * CH_INCNT	CH_WIDTH * CH_OUTCNT
Depth	size / width	size / width

A.6.5 Status codes

In addition to the pre-defined status codes each LL driver can define individual status codes. These "device-specific" codes must be in the range 0x1100..0x11ff.

Convention: Device-specific status codes have the prefix "<dev>_".

Device-specific status codes are defined in an external include file.

Example:

#define M5_ExtPin 0x1101 / get status of EXT pin */*

A.7 Accessing descriptor structure parameters

The HL driver uses the descriptor structure as a management structure for the controlled device.

The LL driver can reference some elements of the descriptor structure by the use of the following macros:

Table A.19 - Parameters that can be referred to by the LL driver

Macro	Type	R/W	Description	Range
MODULE DATA:				
CH_NUMBER	UnBit8	R	no. of channels on device*	1..256
CH_CURRENT	UnBit8	R	current channel*	0..255
OPT_PTR	void*	R	device options location**	0=none, >0=address
OPT_LEN	UnBit16	R	device options size [byte]**	0=none, >0=size
USE_MID	UnBit8	R	module-id data exist*	0=no, 1=yes
CHANNEL DATA:				
CH_OPTION(i)	UnBit16	R	channel option (channel i)**	0x0..0xffff
CH_TYPE(i)	UnBit16	R,W	channel type (channel i)*	0x0..0xffff
IRQ DATA:				
IRQ_VECT	UnBit8	R	irq vector number	0=none, 1..255
IRQ_LEVEL	UnBit8	R	irq level	0..255
IRQ_INSTALLED	UnBit8	R	irq is installed*	0=no, 1=yes
IRQ_ENABLED	UnBit8	R	irq is enabled*	0=no, 1=yes
IRQ_MODE	UnBit16	R	irq mode**	0x0..0xffff
COMMON DATA:				
CH_WIDTH	UnBit8	R	channel width in buffer [byte]*	1,2,4
CH_INCNT	UnBit8	R	no. of input channels*	1..16
CH_OUTCNT	UnBit8	R	no. of output channels*	1..16
MODULE ID DATA:				
MID_REV	UnBit16	R	hardware revision number*	0x0..0xffff
MID_ATT	UnBit16	R	module attributes	0x0..0xffff
MID_IMP	UnBit16	R	intermodule port*	0x0..0xffff
MID_MANUFACT	UnBit16	R	manufacturer data (8 words) *	0x0..0xffff

Notes:
1 R/W: R=read, W=write access allowed
2 *=items of interest
3 **=optional use

A.7.1 MODULE DATA parameters

Global parameters of a device are described by MODULE DATA parameters.

CH_NUMBER total number of device channels

CH_CURRENT current channel number

OPT_PTR points to the device options field

OPT_LEN indicates the size of the device options field

USE_MID indicates whether a module identification exists

The **CH_NUMBER** element gives the total number of channels (and therefore of CHANNEL_DATA structures) of the device.

The HL driver must always set the current channel **CH_CURRENT** to the channel that is being served by the current LL function. With the LL functions Init, Exit, Read and Write, the current channel corresponds to the value of the function parameter "ch". This parameter is not available for the LL functions setstat and getstat, so that these functions have to fetch the channel numbers via the CH_CURRENT macro.

If an LL driver requires global parameters, the HL driver must provide it with a data buffer, the device options field. The **OPT_PTR** pointer points to this buffer, the size of which is defined in **OPT_LEN**.

The **USE_MID** flag signals that the module identification has been read out by the HL driver and that, therefore, the **MID_xxx** parameters have been initialized.

A.7.2 CHANNEL DATA parameters

Each channel of a device is described by CHANNEL DATA parameters.

CH_OPTION channel-specific parameter

CH_TYPE channel description

The **CH_OPTION** element has been reserved for arbitrary use in the LL driver. If the channel option is altered by the HL driver at runtime, this must be signaled to the LL driver by calling the setstat function via the **CH_option** status code.

The **CH_TYPE** channel description defines type, direction and width of a channel:

The channel type (CH_TYPE) may be modified if the configuration of the channel width or direction can be altered on the device at runtime (e.g. TTL I/O with configurable ports).

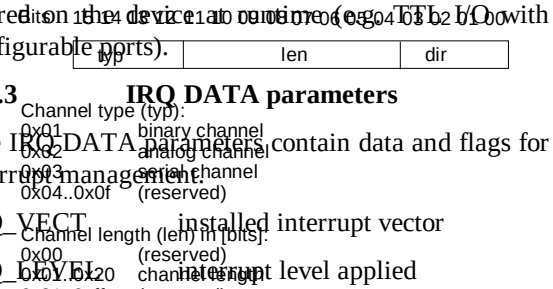


Figure A.4 - Channel description CH_TYPE

A.7.3 IRQ DATA parameters

The IRQ DATA parameters contain data and flags for interrupt management:

IRQ_VECT installed interrupt vector

IRQ_LEVEL interrupt level applied

IRQ_INSTALLED indicates whether an interrupt is installed

IRQ_ENABLED indicates whether an interrupt is enabled

IRQ_MODE interrupt specific parameter

The **IRQ_MODE** element has been reserved for arbitrary use in the LL driver. Its task is to provide the interrupt routine with a user-definable parameter. At runtime, the LL driver is not signaled when the interrupt mode is changed.

The **IRQ_INSTALLED** flag is set by the HL driver as soon as the interrupt (**IRQ_VECT**, **IRQ_LEVEL**) is installed in the system.

The **IRQ_ENABLED** flag is set by the HL driver while the interrupt is enabled on the device.

A.7.4 COMMON DATA parameters

The COMMON DATA parameters contain further global device parameters.

CH_WIDTH	channel width used for buffer entries
CH_INCNT	number of input channels of the device
CH_OUTCNT	number of output channels of the device

The value of CH_WIDTH represents the width of the "widest" channel of the device as defined in the descriptor.

The value of CH_INCNT/CH_OUTCNT represents the number of input and output channels on the device. Channels of the "input/output" type are counted twice, i.e. a device with 4 I/O channels has 4 input and 4 output channels.

A.7.5 MODULE ID DATA parameters

If a module identification exists for a device (USE_MID), the MODULE ID DATA parameters contain the contents of the EEPROM (according to the hardware specification).

MID_REV	hardware revision number
MID_ATT	hardware attribute flags
MID_IMP	intermodule port flags
MID_MANUFACT	start of manufacturer data field (8 words)

(For detailed information on the module identification please refer to the hardware specification.)

A.8 Appendix

A.8.1 Error codes

The following error codes are available for the LL driver:

0
(no error)

ERR_Init	general initialization error
ERR_Exit	general termination error
ERR_Read	general read error
ERR_Write	general write error
ERR_IllParam	illegal parameter

ERR_UnkCode
unknown status code

A.8.2 Status codes

The following status codes are pre-defined:

CH_option	channel option has been changed
IRQ_enable	interrupt enable/disable
SIG_set_cond1	activate signal condition 1
SIG_set_cond2	activate signal condition 2
SIG_set_cond3	activate signal condition 3
SIG_set_cond4	activate signal condition 4

SIG_clr_cond1
de-activate signal condition 1

SIG_clr_cond2
de-activate signal condition 2

SIG_clr_cond3
de-activate signal condition 3

SIG_clr_cond4
de-activate signal condition 4

A.8.3 Macros

The LL driver requires the following macros:

Table A.20 - Macros

Macro	Example	Function
HwReg8(addr)	(UnBit8*)(addr)	access to an 8-bit register at address "addr"
HwReg16(addr)	(UnBit16*)(addr)	access to a 16-bit register at address "addr"
HwReg32(addr)	(UnBit32*)(addr)	access to a 32-bit register at address "addr"
CH_NUMBER	mod_ptr->ch_number	access to MODULE DATA
CH_CURRENT	mod_ptr->ch_current	access to MODULE DATA
OPT_PTR	mod_ptr->opt_ptr	access to MODULE DATA
OPT_LEN	mod_ptr->opt_len	access to MODULE DATA
USE_MID	mod_ptr->use_mid	access to MODULE DATA
CH_OPTION(i)	mod_ptr->ch_ptr[i]->option	access to CHANNEL DATA
CH_TYPE(i)	mod_ptr->ch_ptr[i]->ch_type	access to CHANNEL DATA
IRQ_VECT	mod_ptr->irq_ptr->vect	access to IRQ DATA
IRQ_LEVEL	mod_ptr->irq_ptr->level	access to IRQ DATA
IRQ_INSTALLED	mod_ptr->irq_ptr->installed	access to IRQ DATA
IRQ_MODE	mod_ptr->irq_ptr->mode	access to IRQ DATA
CH_WIDTH	mod_ptr->com_ptr->ch_width	access to COMMON DATA
CH_INCNT	mod_ptr->com_ptr->ch_incnt	access to COMMON DATA
CH_OUTCNT	mod_ptr->com_ptr->ch_outcnt	access to COMMON DATA
MID_REV	mod_ptr->mid_ptr->rev	access to MODULE ID DATA
MID_ATT	mod_ptr->mid_ptr->att	access to MODULE ID DATA
MID_IMP	mod_ptr->mid_ptr->imp	access to MODULE ID DATA
MID_MANUFACT	mod_ptr->mid_ptr->manufact	access to MODULE ID DATA

Annex B

(informative)

Conformance test specification

B.1 Introduction

B.1.1 Objectives

This document provides the manufacturers and users of M-Modules with a detailed technical description of the conformance testing process: the parameters under test, the methods to be used and the conditions under which they will be tested.

The conformance test applies only to the basic electrical, the mechanical and thermal specifications. Interpretation of the specification is discussed, if necessary.

B.1.2 Test conditions

Unless otherwise noted, empirical testing is done by plugging the module under test onto a base board with proven M-Module-compliant signal timing.

The tests are carried out at room temperature with the +5V and the +12V and -12V supplies set to the nominal value (accuracy: 1%).

B.1.3 Rule for worst-case timing analysis

All worst-case timing analysis is done based on each part's data book parameter.

The line loading effect on the device propagation delays is corrected with the following units:

- max. capacitive load is assumed to be 100pF;
- add 10nsec to the propagation delay (as specified at 30pF load),
- add 4nsec to the turn-on delay.

If the data book does not specify a minimum propagation delay, a value equal to one-sixth of the maximum specified delay is used.

B.1.4 Additional testing

Worst-case analysis is a method to verify the design against a set of rules. This is very useful for detecting timing-related problems, but will not always uncover problems related to insufficient power and ground distribution (e.g. ground bounce and glitches).

Therefore empirical testing might be carried out as an addition to worst-case timing analysis.

B.1.5 Signal nomenclature

The signal named "/SELECT" refers to the three signals /CS, /DACK and /IACK, of which only one is asserted during a cycle.

B.2 Timing

B.2.1 Rule 1

A module must operate with a minimum setup time of address and control signals with respect to the /SELECT signal.

Method of verification:

Use the schematics of the M-Module to be tested to determine whether the module will operate properly (input port select signals, output port write, etc.) with a minimum setup time of 10nsec (timing parameter #1).

Criteria for non-conformance:

Data corruption is possible when applying address and control signals with minimum setup time.

B.2.2 Rule 2

A module must operate with a minimum hold-time of address and control signals with respect to the /SELECT signal.

Method of verification:

Use the schematics to determine whether the module will operate properly (input port select signals, output port write, etc.) with a minimum hold time of 10nsec (timing parameter #6).

Criteria for non-conformance:

Data corruption is possible when applying address and control signals with minimum hold time.

B.2.3 Rule 3

A module must operate with a minimum setup time of data signals with respect to the /SELECT signal.

Method of verification:

Use the schematics to determine whether the module will operate properly (input data vs. output device write signal, etc.) with a minimum setup time of 0nsec on the base board interface (timing parameter #1a).

Criteria for non-conformance:

Data corruption is possible when applying address and control signals with minimum setup time.

B.2.4 Rule 4

A module must operate with a minimum hold time of data signals with respect to the /SELECT signal.

Method of verification:

Use the schematics to determine whether the module will operate properly during a write cycle (input data vs. output device write signal, etc.) with a minimum hold time of 10nsec on the base board interface (timing parameter #6).

Criteria for non-conformance:

Data corruption is possible when applying data and control signals with minimum hold time.

B.2.5 Rule 5

A module must keep its read data lines stable during the transfer acknowledge period.

Method of verification:

Use the schematics to determine whether the module will maintain a stable level on its data-lines not later than 25nsec after the assertion of /DTACK (timing parameter #4) until the /SELECT signal is released.

Comment:

When reading data from an input device (e.g. 74BCT244) two parameters are critical:

- data-out active vs. /DTACK asserted (in most cases no problem);

- the data-lines must stay valid during the remaining part of the read cycle. This cannot be guaranteed when asynchronous signals are connected to an input device without a latching function.

Advice: use a 74373 with pin LE connected to OE.

Criteria for non-conformance:

Read data may change during the acknowledge period when applying a changing data pattern to the input device on the module.

B.2.6 Rule 6

A module must assert the /DTACK signal not later than 10µsec after being selected.

Method of verification:

Use the schematics and device data book to determine whether the module will assert the /DTACK signal not longer than 10µsec after the /SELECT signal is asserted (timing parameter #3).

Criteria for non-conformance:

The /DTACK signal is not generated within a period of 10µsec after the assertion of the /SELECT signal.

B.2.7 Rule 7

A module may assert an "early" /DTACK.

Method of verification:

Use the schematics and component data book to determine whether the module will maintain a stable level on its data lines not later than 25nsec after the assertion of /DTACK (timing parameter #4).

Comment:

When implementing simple input/output modules, the /SELECT signal can be used to generate /DTACK and enable input buffers. The number of buffers in series in the data path must be limited to guarantee the maximum delay between /DTACK and the data lines.

Criteria for non-conformance:

The read data cycle has stable data later than 25nsec after the assertion of /DTACK.

B.2.8 Rule 8

A module must release the /DTACK signal within a specified interval after the de-assertion of the /SELECT signal.

Method of verification:

Use the schematics and component data book to determine whether the module will release /DTACK in the interval of 0 to 100nsec after the de-assertion of /SELECT (timing parameter #7).

Comment:

/DTACK is switched off after a module is de-selected. No harm is done when the /DTACK signal is stretched for more than 100nsec, but the system performance is reduced as a new cycle may only start after a release of /DTACK.

Criteria for non-conformance:

1. The /DTACK signal is switched off before the /SELECT signal is de-asserted.

or

2. /DTACK is held active for more than 100nsec after the de-assertion of the /SELECT signal.

B.2.9 Rule 9

A module must switch its data line drivers to the high impedance state within 10nsec from the release of /DTACK.

Method of verification:

Use the schematics and component data book to determine whether the module switches its data line driver(s)/transceiver(s) within 10nsec after the de-assertion of /DTACK (timing parameter #9).

Comment:

The module must switch off the data line drivers in time as a new write cycle may be started on the base board in a back-to-back timing relationship. If this requirement is not met, high current peaks can be found on the devices connected to the data path.

Criteria for non-conformance:

The data transceiver(s) are switched to the high-impedance state later than 25nsec after the de-assertion of /DTACK.

B.2.10 Rule 10

A module must respond on the byte lanes as defined by the DS0 and DS1 signals.

Method of verification:

Use the schematics to determine whether the module controls its devices connected to the data path as coded by the /DS0 and /DS1 signal. Further testing can be done by executing a mix of byte or word transfers on the module, or both.

Criteria for non-conformance:

The data patterns written to the output devices and the data read from the input devices do not correspond to the expected values.

B.2.11 Rule 11

If a module has interrupt capabilities it must respond on the IACK cycle as defined by its Interrupt-Acknowledge type.

Method of verification:

1) Use the schematics and component data book to determine whether the module releases its /IRQ signal during the IACK cycle within the time interval specified by timing parameters #20a and #20b.

2) Plug the module on the test base board and observe timing of the /IRQ signal during the IACK cycle.

Comment:

The module must release its /IRQ line in a defined time interval in order to let the interrupt handling terminate its actions and not being restarted by a still pending /IRQ signal.

When the interrupt requesting condition on the module disappears before the IACK cycle is performed, the /IRQ line must stay asserted. This is to prevent the IACK cycle from ending up in a time-out condition, which is very difficult to handle in most systems.

Criteria for non-conformance:

1) The /IRQ signal is released before the start of the IACK cycle (parameter #20a).

2) The module releases its /IRQ line later than 1µsec after the end of the IACK cycle (parameter #20b).

B.2.12 Rule 12

A module must come in a defined state when the /RESET signal is asserted.

Method of verification:

Use the schematics and component data book to determine whether the module is put in an idle state and releases all active signals.

Criteria for non-conformance:

The module does not release all active signals on the base board connector.

B.2.13 Rule 13

If a module has the module identification function, then the access protocol and the layout of the data in the EEPROM must correspond to the definition in the basic electrical specification.

Method of verification:

Place the module on the test base board and use the read algorithm to verify the data with the properties specified by the supplier of the module.

Criteria for non-conformance:

1) The module does not return the ID data as described in the specification.

or

2) The contents of the ID field is not in conformance with the supplier data.

B.3 Electrical

B.3.1 Rule 1

The interface devices of a module must have properties as specified in the "Driver Characteristics".

Method of verification:

Use the schematics and the device data books to verify the electrical specification with the properties specified in the "Driver Characteristics".

Criteria for non-conformance:

The devices interfacing with the base board do not match the electrical characteristics specified for each of the signal types.

B.3.2 Rule 2

The total capacitive load on any signal pin connected to the base board shall not exceed 25pF.

Method of verification:

Use the schematics, PCB artwork and the device data books to calculate the line loading of the signals interfacing with the base board. The trace capacitance of the lines must also be taken into account.

Comment:

The total line capacitance is determined by the architecture of the base board, so the driver characteristics of the base board must match the line loading.

The data bus connection is bidirectional, so the M-Module must be able to drive an interface which is loaded with much more than 25pF. In many cases the timing characteristics of a device are given at a relatively low capacitive load. The timing of the module must be derated to meet the required data setup and hold times as indicated in the introduction.

Criteria for non-conformance:

1) The total capacitance on any signal line exceeds 25pF.

or

2) The module is not capable of driving the data lines within the specified data setup and hold times.

B.3.3 Rule 3

The total power dissipation of a module shall not exceed 10W.

Method of verification:

Place the module on a base board and measure the average current drawn from the system +5V, +12V and -12V supply and the current supplied via the application connector. This measurement must be carried out while exercising the module with a typical application program.

Criteria for non-conformance:

1) The total power dissipation exceeds 10W.

or

2) The current drawn from one of the three system supplies exceeds 1A (5V) or 200mA (12V).

B.3.4 Rule 4

The thermal properties of a module shall match the data as given in the thermal specification.

Method of verification:

Place the module on a base board on which the component area under the M-Module is filled up to its maximum height and volume.

This measurement must be carried out while exercising the module with a typical application program.

Criteria for non-conformance:

The temperature on the module exceeds the maximum operating value of any component.

B.4 Mechanical

B.4.1 Rule 1

The dimensions of the module and the position of the mounting holes shall fit within the limits as defined in figure B.1.

Method of verification:

Compare the dimensions given in figure B.1 with the following list.

Criteria for non-conformance:

One of the measured values [1,4..8] is not within the limits as defined in the table.

Table B.1 - Module dimensions

No.	Dimensions (mm)	Tolerance (mm)
1	52.9	-0.5
2	52	-
3	131	-
4	146.4	±0.1
5	60.7	±0.1
6	124.5	±0.1
7	134.6	±0.1
8	3.81	-0.2

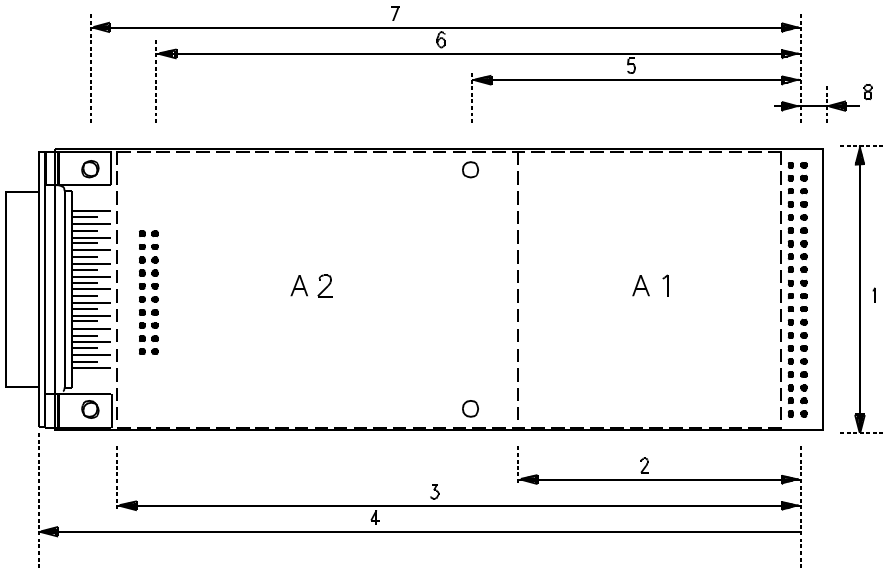


Figure B.1 - M-Module dimensions

B.4.2 Rule 2

The component height in the areas A1 and A2 (c.f. figure B.1) shall fit within the specified limits.

Method of verification:

Measure the height of the largest component in area A1 and A2; compare the data with:

- area A1 < 5.25mm;
- area A2 < 10.5mm.

Criteria for non-conformance:

One of the sizes is not within the limits.

B.4.3 Rule 3

When mounting a module on a base board, the distance between the two boards shall be 12.0mm (±0.1mm).

Method of verification:

Measure the distance between the base board and the module after mounting and securing. The value must be 12.0mm (±0.1mm).

The base board must have a plug header with a maximum height of 10mm over the component surface of the PCB.

Criteria for non-conformance:

The measured value is not within the limit.

B.4.4 Rule 4

Components and component lead(s) on the solder side of a module may project by a maximum of 1.2mm over the surface.

Method of verification:

Measure the length of the leads and the height of the components on the solder side of the module. This value should not exceed 1.2mm.

Criteria for non-conformance:

The measured value is higher than 1.2mm.

B.4.5 Rule 5

When the 24-pin option connector is not implemented, the area may be populated by components with a maximum height of 1.0mm.

Method of verification:

Measure the height of the component or components in the indicated area.

Criteria for non-conformance:

The measured value is higher than 1.0mm.