BASIC SOFTWARE

USER'S MANUAL

MCX and MCX-Lite/S cards

BASIC SOFTWARE USER'S MANUAL FOR MCX AND MCX-*Lite*/S CARDS

REGISTERED TRADEMARKS ®

- *ACKSYS* is a registered trademark of *ACKSYS*.
- IBM P.C.is a registered trademark of International Business Machines Corporation.
- MS-DOS is a registered trademark of Microsoft Inc.
- TURBO PASCAL is a trademark of Borland International Inc.

NOTICE

ACKSYS ® provides this documentation "as is" without warranty of any kind. In no event shall ACKSYS be held responsible for the profitability and conformity of the hardware compared to the user's requirements.

ACKSYS ® shall not be held responsible for any errors that might be contained in this document, nor for any damages of any amount that the supply, operation or use of this equipment may entail.

ACKSYS ® may revise this document from time to time, or change its contents, without notice.

ACKSYS Company
3 & 5 rue du Stade
BP 80
78302 POISSY CEDEX - FRANCE

Telephone : 33 (1) 39-11-62-81
Fax : 33 (1) 39-11-47-96

English version, February 11, 1997. Revision B-02. Basic software.

English version, February 11, 1997. Revision B-02. Basic software.

TABLE OF CONTENTS

I. INTRODUCTION

I.1 Manual contents

This manual describes the "basic software" included in the MCX series card's EPROM. This software is designed for compatibility with the MCC series cards (thus the designation "MCC emulation mode"). It enables the programmer to use the features of the serial channels in asynchronous transmissions.

The four chapters in this manual provides all the information necessary to the program included in the card's EPROM. Here is a brief description of the manual layout:

The first chapter defines the dialog procedures between the PC and the card using commands and interrupts.

The second chapter explains the procedure for starting the program and initializing the channels.

The third chapter describes the meaning of the LEDs on the card in relation to control operations and transmission or reception of characters.

Finally, the last chapter details the card's commands.

All chapters should be studied carefully; hence we suggest that you read this manual from beginning to end.

I.2 Conventions used

Two major conventions are used in this manual:

The first concerns decimal numbers. All the numbers in this manual are written in decimal format; any hexadecimal numbers are followed by an "h".

The second convention was adopted to simplify the wording. Since this manual is designed for users of the MCX and MCX-*Lite*/S cards, we have decided to not specify the type of card (MCX or MCX-*Lite*/S) when information concerns both cards.

I.3 Compatibility of the basic software between the MCC, MCX and MCX-*Lite*/S cards

This section is important for programmers who have already used an ACKSYS MCC or MCC/II card.

The MCX and MCX-*Lite*/S cards were designed to be compatible to a large extent with the MCC card. They extend the performance capabilities of the ACKSYS communication products, let you select the optimal price/performance solution for a given application, and let you easily migrate existing applications to much more powerful platforms.

The power of its microprocessor (386 SX 25 Mhz, 486 SLC 25 Mhz, ...) makes it a natural choice for applications that handle high-speed lines.
Its structure prevents it from controlling the OCTOPUCE card (8 V23 modems), however. We have included software-controlled mixed RS232D / RS422A interfaces in the MCX and MCX-*Lite*/S cards, making them particularly suitable for industrial applications.

As a result, some commands on the MCC card have been removed or modified, while others have been added to manage the new functions in the MCX and MCX-*Lite*/S cards.

Removed commands:

- MTURN: turn around the OCTOPUCE modem card
- ASYSP: Program the speed in asymmetrical mode
- ASYMD: Switch to asymmetrical mode
- LDIAL: Dial the OCTOPUCE modem card
- MSETP: Initialize the OCTOPUCE modem card
- TANNU: Manage the electronic phone directory

Additional commands:

- RSMDE: Program the RS232D or RS422A mode
- FLASH: Program the 256 Kbyte Flash memory

Modified commands:

- RELRP: Identification of the card and options
- ALLOC: Allocate memory buffers
- BPARM: Read the buffer parameters
- MSIZE: Read the memory configuration
- VINIT: Initialize the line speed (special speeds only)

Please refer to Chapter V for information on the additional or modified commands.

Other important differences:

- The INT 08 software interrupt used for the ADDCM, ADDTX and ADDRX commands is now INT 07.

- The LD_BUFF_PARM (86h) function in INT 08 was modified to handle up to 64 communication lines.

- There is no longer a limited UNIX version with a restricted command set.

- The event queue may now contain 9,000 entries instead of 2,500.

- The amount of memory reserved for communication buffers is increased from 506,000 to 786,420 characters.

- Some useful functions were located at fixed addresses and could be called by a CALL to an absolute address. These functions are now only accessible through the INT 07 software interrupt.

- The mailbox now contains 32 Kbytes instead of 16 or 32 Kbytes on the MCC card; however, its address may now be located beyond the first megabyte of memory.

- Finally, the base address of the transmit and receive counters located at the top of the mailbox had to be moved due to the increase in the number of channels on the MCX card, which is compatible with the MCX-*Lite*/S card.

Base address for the MCC RX counters = 7FC0H	MCX = 7F00H
Base address for the MCC TX counters = 7FE0H	MCX = 7F80H

NOTE

This section only concerns the compatibility of the basic software of the MCC and MCX (or MCX-*Lite*/S) cards. Technically, the cards are completely different.

II. DIALOG PROCEDURE BETWEEN THE CARD AND THE PC

Communications between the card and the host computer in which it is installed use a 32Kbyte dual-ported memory area. The address of this area is programmable (see the "Installation and Technical Specifications Manual for the MCX Range").

The table on the next page describes the use of this memory, also called the mailbox.

The first column in the table contains the zone's offset from the mailbox's address in the PC. The second column indicates the type of operation that can be performed at that address:

WO	Write only
RO	Read only
RW	Read / Write

The third column contains the corresponding mnemonic.

Subdivision of the mailbox

Dec / Hex	Perm	Name	Field Description	
000 / 000h	RW	VALID	Write command validation byte: 01h	**VALIDATION ZONE**
001 / 001h	RW	ENDIT	Write end-of-command acknowledge	**ENDIT ZONE**
002 / 002h	WO	OPCODE	Write command number	**OPCODE ZONE**
003 / 003h	RO	STATUS	End-of-command status / "event" interrupt indicator	**STATUS ZONE**
004 / 004h	RW	PAR. 1	Parameter # 1	**PARAMETER ZONE**
005 / 005h	RW	PAR. 2	Parameter # 2	
006 / 006h	RW	PAR. 3	Parameter # 3	
007 / 007h	RW	PAR. 4	Parameter # 4	
079 / 4Fh	RW	PAR. 76	Parameter # 76	
080 / 50h	RO	I.CHAN	Interrupt, channel #	**INTERRUPT ZONE**
081 / 51h	RO	I.COND	Interrupt, Type	
082 / 52h	RO	I.PAR1	Interrupt, Status # 1	
083 / 53h	RO	I.PAR2	Interrupt, Status # 2	
084 / 54h	RO	I.PAR3	Interrupt, Status # 3	
085 / 55h			Address not used	**RESERVED ZONE**
099 / 63h			Address not used	
100 / 64h	RW	DONNEE 1	Data zone, 1st byte	**DATA ZONE**
101 / 65h	RW	DONNEE 2	Data zone, 2nd byte	
102 / 66h	RW	DONNEE 3	Data zone, 3rd byte	
32511/7EFFh	RW	DONNEE 32412	Data zone, 32,412nd byte	
32512/7F00h	RO	RX Cnt 1	RX counter channel 1	**RX[1] COUNTER ZONE**
32514/7F02h	RO	RX Cnt 2	RX counter channel 2	
32638/7F7Eh	RO	RX Cnt 64	RX counter channel 64	
32640/7F80h	RO	TX Cnt 1	TX counter channel 1	**TX[2] COUNTER ZONE**
32642/7F82h	RO	TX Cnt 2	TX counter channel 2	
32766/7FFEh	RO	TX Cnt 64	TX counter channel 64	

[1] Each counter contains the number of characters in the receive buffer. Only the RX counters associated with installed channels are significant.

[2] Each counter contains the number of characters in the transmit buffer. Only the TX counters associated with installed channels are significant.

II.1 Setting up the program

Before using the "MCC Emulation" mode, make sure the jumpers on your card are positioned as shown below:

ST1 at 1-2	ST2 at 1-2	ST3 at 1-2	ST4 at 1-2	ST5 at 1-2

After the initialization phase, the LED display counts up to eight bits then starts again, address 0 of the mailbox contains 0Fh, and the following character string is written at address 100 (64h):

```
MCX IS READY
```

Now you must send the card the following code to start the "MCC Emulation" program:

```
RUN 01
```

This uppercase character string must be written in the data zone of the dual-ported memory (as of address 100 (64h)).

In reply to this startup sequence, the card resets to zero the byte at address 0 in the mailbox (it was at 0Fh before writing this sequence).
Furthermore, the LEDs stop counting and LED 0 starts to blink once a second.

The card is now ready to receive and execute commands.

II.2 Sending commands to the card

Use the following procedure to send a command to the card:

- Write the command code in the OPCODE zone (Address 2)
- Write the required parameters in the PARAMETER zone (Addresses 3, 4 to 4Fh)
- Write the data required in the DATA zone (Addresses 64h to 7EFFh)
- Write the validation byte (01h) in the VALIDATION zone (Address 0).

An internal mechanism then lets the card execute the command.

Once the command has been executed, the card generates an interrupt on the selected line (see the installation manual) after having updated the following zones:

- The STATUS zone contains an end-of-command interrupt status that can be identified by bit D6 set to 0, an error indicator in bit D7, and the error code sent by the command in bits D5-D4-D3-D2-D1-D0. Bit D7 indicates how the command executed. If the command returned an error, D7 is set to 1 and an error code (different from 0) is written to bits D5-D4-D3-D2-D1-D0. If no error was returned, D7 is set to 0 and an error code of 0 is written to bits D5-D4-D3-D2-D1-D0: so the status byte is null.
- The DATA zone contains the data sent by the command, if appropriate.
- The VALIDATION byte is reset to 0, indicating that a new command may be sent.
- Bit D7 in the ENDIT zone is set to 1.

In any case, you must acknowledge the end-of-command interrupt using the procedure described in section II.3.3.

II.2.1 Error code management

The card returns an error code (also named return code) for each command, which indicates if the command executed normally, if an error was encountered in one of the parameters, or if an error occurred during command execution.

In some cases the error codes may be used as indicators, as described below:

The BTRAN command returns an error if the number of characters to be sent on the channel is greater than the number of characters available in the buffer at that time. If the buffer size was correctly initialized from the beginning with a size greater than the maximum size of information blocks you wish to send, the error indicates that the buffer is currently occupied but will be freed. In this case, you can make the BTRAN command loop until the return code indicates there is no error.

This method of using the BTRAN command is also valid for the ISEND command.

Finally, this feature may also be used with the RDBUF command. You can test the return code which indicates if the number of characters in the buffer is less than the number requested. In that case, your program can loop until all the characters requested have been transferred to the receive buffer.

For the other commands, if the return code indicates an error it means that there was an error in the command parameters or that the channel was not properly initialized.

Chapter V contains information concerning return codes and probable causes for errors. Appendix "**Error code summary**" at the end of the manual provides a summary of all error codes.

II.3 Interrupt management

As described in the preceding section, the card generates interrupts on the machine's bus[1].

Two major elements may cause interrupts:

The first concerns **end-of-command interrupts**.
The second type of interrupt is more complex. The interrupt is triggered by conditions selected with the MINTR command. These are called **"event" interrupts**.

II.3.1 End-of-command interrupts

For these interrupts, bit 6 of the status byte is set to zero.
The PC must absolutely respond to the end-of-command interrupt by sending an ENDIT[2] command.

II.3.2 "Event" interrupts

For these interrupts, bit 6 in the status byte is set to 1.
The card writes additional information in the dual-ported memory (addresses 80, 81, 82, 83, 84 / 50h, 51h, 52h, 53h, 54h respectively).

[1] A sample program under interrupts is provided at the end of the manual (section VI.6).
[2] See the section on acknowledging interrupts (section II.3.3).

We will examine the ten possible cases one by one:

Case #0: IT0

Location	**Interrupt triggered for each character received.**
Address 80 (50h)	Channel # where character was received
Address 81 (51h)	Type of interrupt: 01h
Address 82 (52h)	Least significant byte of the number of characters remaining in the buffer minus this character.
Address 83 (53h)	Most significant byte of the number of characters remaining in the buffer minus this character.
Address 84 (54h)	Character read. The character counter is automatically decremented by 1.

Case #1: IT1 (STCNT Parameter of the STCNT command = 1)

Location	**Interrupt triggered when the first character is received.**
Address 80 (50h)	Channel # where the event was detected.
Address 81 (51h)	Type of interrupt: 02h
Address 82 (52h)	Least significant byte of the number of characters in the buffer.
Address 83 (53h)	Most significant byte of the number of characters in the buffer.
Address 84 (54h)	Not significant.

Case #1': IT1 (STCNT Parameter of the STCNT command > 1)

Location	**Interrupt triggered when the number of characters received is modulo the number of characters programmed by the STCNT command.**
Address 80 (50h)	Channel # where the event was detected.
Address 81 (51h)	Type of interrupt: 02h
Address 82 (52h)	Least significant byte of the number of characters in the buffer.
Address 83 (53h)	Most significant byte of the number of characters in the buffer.
Address 84 (54h)	Not significant.

Case #2[1]: IT2 (Mde parameter of the MINTR command = 0)

Location	Interrupt triggered when the receive buffer is full
Address 80 (50h)	Channel # where the event was detected.
Address 81 (51h)	Type of interrupt: 04h
Address 82 (52h)	Least significant byte of the number of characters in the buffer
Address 83 (53h)	Most significant byte of the number of characters in the buffer
Address 84 (54h)	Not significant

Case #2'[1]: IT2 (Mde parameter of the MINTR command = 1)

Location	Interrupt triggered when the buffer goes from empty to not empty
Address 80 (50h)	Channel # where the event was detected.
Address 81 (51h)	Type of interrupt: 03h
Address 82 (52h)	Not significant.
Address 83 (53h)	Not significant.
Address 84 (54h)	Not significant.

Case #3: IT3

Location	Interrupt triggered when a character string programmed by the CHDEF command is recognized
Address 80 (50h)	Channel # where the event was detected
Address 81 (51h)	Type of interrupt: 08h
Address 82 (52h)	Least significant byte of the number of characters in the buffer, including the string detected
Address 83 (53h)	Most significant byte of the number of characters in the buffer, including the string detected
Address 84 (54h)	Contains the detected character if comparing only one character, or the second character if comparing two characters[2]

[1] See the MINTR command (section V.22)

[2] This feature is useful if you use the CHDEF command to program characters that contain an "Escape (1BH, 27D)" in the first position and another character in the second position (for arrow keys, for example). You can then directly read the second character to determine which key was pressed.

Case #4: IT4

Location	Interrupt triggered by a timeout[1]
Address 80 (50h)	Channel # where the event was detected
Address 81 (51h)	Type of interrupt: 10h
Address 82 (52h)	Least significant byte of the number of characters in the buffer
Address 83 (53h)	Most significant byte of the number of characters in the buffer
Address 84 (54h)	Not significant

Case #5: IT5

Location	Interrupt triggered when a receive error is detected
Address 80 (50h)	Channel # where the event was detected
Address 81 (51h)	Type of interrupt: 20h
Address 82 (52h)	Least significant byte of the number of characters in the buffer
Address 83 (53h)	Most significant byte of the number of characters in the buffer
Address 84 (54h)	Type of error: (SFRP0000) - if P = 1 parity error - if R = 1 and S = 0, a character was lost in the SCC hardware receive buffer - if F = 1 and S = 1, card's software receive buffer overflowed - if F = 1 frame error The flag is available from EPROM version 1.6 on

The bad character is written to the receive buffer even if a receive error occurs.

[1] See the STTMO command in section V.36.

Case #6: IT6

Location	Interrupt triggered by a change in CTS, CD, RI, or BREAK
Address 80 (50h)	Channel # where the event was detected
Address 81 (51h)	Type of interrupt: 40h
Address 82 (52h)	Least significant byte of the number of characters in the buffer
Address 83 (53h)	Most significant byte of the number of characters in the buffer
Address 84 (54h)	Signal status: (B0CRD000) - If D = 1 then CD is high - If C = 1 then CTS is high - If B = 1 then a "BREAK" was detected on the line[1] - If R = 1 then RI is high

Case #7: IT7

Location	Interrupt triggered if the transmit buffer is completely emptied (the entire buffer is sent, including the stop bits)
Address 80 (50h)	Channel # where the event was detected
Address 81 (51h)	Type of interrupt: 80h
Address 82 (52h)	Not significant
Address 83 (53h)	Not significant
Address 84 (54h)	Not significant

Important note

Note that all these conditions may be individually masked or selected using the MINTR command.

Several conditions may be programmed for a single channel, and you may choose different conditions for each communication channel.

"Event" interrupts are sent by the card in chronological order. An "event" interrupt only indicates one event at a time.

If several events occur at the same time for a single character, the card will generate an interrupt for each event.

Just as for end-of-command interrupts, it is absolutely necessary to send an ENDIT[2] command to enable the card to send another interrupt.

[1] A "BREAK" triggers 2 interrupts. The first is sent as soon as the BREAK is detected; the (B) bit is high. The second interrupt is sent when the end of BREAK is detected; the (B) bit is then low (0).
[2] See the section on acknowledging interrupts (section II.3.3).

II.3.3 Acknowledging interrupts

To acknowledge an interrupt, write the byte 01h to address 1 in the mailbox (ENDIT Zone).

Note that the ENDIT command should be sent in the same way when acknowledging "event" and end-of-command interrupts.

II.3.4 Interrupt priorities

Each interrupt condition has its own priority level, depending on the type of interrupt. If several interrupts occur for a single character, their priority level determines the order in which the they are sent to the PC.

The table below illustrates the priorities for the various interrupt conditions:

Priority 0	IT0 IT1 IT4 IT5 IT6
Priority 1	IT7 IT2
Priority 2	IT3

Level 0 priority is the highest.

II.3.5 The interrupt queue

"Event" interrupts as well as end-of-command interrupts are stored in a queue which may contain up to 9,000 entries.

If the machine cannot handle these interrupts fast enough, the queue fills up. If an overflow occurs, the card signals the incident by a conditional interrupt with 0FFH as the parameter code at address 80 (50h).

The character received, end-of-command and end-of-transmit buffer interrupts, however, wait until there is enough room in the queue before being added to it.

When the queue is totally full, the card waits for as many "ENDIT" sequences as there are events in the queue.

II.4 Using the card in "polling" mode

In this mode, the computer ignores interrupts sent by the card.
To detect end-of-command and event interrupts, a status bit is read. This is bit D7 in the ENDIT zone, and is called the STATRDY bit.

In this mode, commands are still sent using the procedure described in section II.2.

The interrupt triggered by the card can be detected because the STATRDY bit goes to 1. This bit should be tested as often as possible to avoid filling up the interrupt queue. The type of interrupt may then be identified by reading the STATUS zone, and the associated information (DATA and INTERRUPT zones) should be processed.

The ENDIT command should finally be sent using the procedure described in section II.3.3, in order to enable the card to send the next interrupt.

Important note:

Because of the design of the hardware, the use of the card in "polling" mode is not recommended. Overly systematic "polling" of the dual-ported memory can slow down the card to some extent and consequently reduce performance.

Managing the card with interrupts provides much better results than using "polling" mode.

III. INITIALIZING CHANNELS THROUGH SOFTWARE

You must initialize the card's operating mode before it can work in an application.

The following command is mandatory:

VINIT: Initialize the communication lines for speed, number of bits, etc.

The following commands are optional; their use depends on your application.

RSMDE: Select the RS232D or RS422A communication mode.
DALOC: Deallocate the communication buffers **if the default sizes are not suitable**.
ALLOC: Allocate the communication buffers.
VMODE: Define the handshake mode and echo.
CHDEF: Define the end-of-string characters.
STCNT: Define the size of a character block.
STIME: Set the time on the card.
STTMO: Define the receive timeout value.
MINTR: Initialize the interrupt mode.
RXENB: Receive enable.

Note:
Appendix VI.5 at the end of the manual provides an initialization sequence that clearly illustrates how to use the communication lines on the MCX and MCX-*Lite*/S cards.

IV. INTERPRETING THE GROUP OF EIGHT LEDS

IV.1 LEDs in common with the MCX and MCX-*Lite*/0 cards

The MCX and MCX-*Lite*/0 cards include eight LEDs to assist during power-on diagnostics, and to provide visual feedback on control operations, and when sending and receiving characters.
At power-on, LEDs 0 through 7 light up in succession to indicate the different phases of the test.
If one or more of the LEDs stays on permanently, an error has occurred. You must turn off the system and contact ACKSYS technical support.

As soon as the start code is sent ("`RUN 01`"), the card is operational, and LED 0 blinks once per second.

The eight LEDs then have the following meaning:

- LED 0 Activity (blinks once per second)
- LED 1 Command being processed
- LED 2 Interrupt towards the PC active
- LED 3 Receiving in asynchronous mode
- LED 4 Transmitting in asynchronous mode
- LED 5 Not significant (always off)
- LED 6 Not significant (always off)
- LED 7 Receive error

IV.2 The RS232/RS422 mode control LEDs

Special case for the MCX-*Lite*/S card

Two yellow LEDs on the Lite-SERIAL extension indicate the operating mode (RS232D or RS422A) for each communication line.

LED DL1 channel 1 in RS422 mode if on, channel 1 in RS232 mode if not.
LED DL2 channel 2 in RS422 mode if on, channel 2 in RS232 mode if not.

Special case for the MCX-XX card

Eight yellow LEDs on the MCX-BP extension indicate the mode for each of the card's 8 channels. The LED is on for channels programmed in RS422 mode, and stays off for channels programmed in RS232 mode.

V. SOFTWARE COMMANDS

V.1 ADDCM (28h): Add a command to the standard command set

OPCODE = 40 (28h)
This command lets you add commands to the card's standard command set, and to create "MACROS".
Warning: The MCXDEBUG[1] program is required to use this command effectively.
Parameters #1 and #2 must contain the size of the command, in bytes. When the installation is finished, the command's opcode and associated error code are returned in parameters 1 and 2, respectively.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	1	0	0	0
PARAMETER 1	Cml	Cml	Cml	Cml	Cml	Cml	Cml	Cml
PARAMETER 2	Cmh	Cmh	Cmh	Cmh	Cmh	Cmh	Cmh	Cmh
DATA ZONE	COMMAND TO ADD TO MEMORY							

Cml: PARAMETER #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

Parameter #1 contains the least significant byte of the size of the command to add, in bytes.
This parameter returns the order number of the command that was just installed. Warning: this order number may vary from one version to the next of the basic firmware; the card provides numbers in sequential order.
You may add up to 20 new commands.

Cmh: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

Parameter #2 contains the most significant byte of the size of the command to add.
This parameter returns the error number for the command that was just installed. Warning: this error number may vary from one version of the firmware to the next. Furthermore, the size of the command should not be greater than the amount of memory available.

DATA ZONE
The data zone in the dual-ported memory must contain the code of the command to install.
The data zone is located at a positive offset of 100 (64h) from the start of the dual-ported memory.

[1] Optional development software from ACKSYS for the MCX and MCX-*Lite*/S cards.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 164 (A4h) is returned to indicate an error in the command.

The cause of the error may be:

- Not enough memory to install the command
- Size of the command equals zero
- Too many commands installed

IMPORTANT NOTE

It is absolutely imperative that the code of the command be compiled with an origin of 0000h.
Standard function and command calls for the card can be made with software interrupt INT 07, where the [AL] register contains the number of the function or command called (see Appendix VI.5).
Local data must be declared in the same segment as the code of the command. No intersegment references may be used.
When the command returns, the [AL] register should indicate if an error has occurred.

[AL]	Meaning
0	No error
1	Error. The error code returned is the one provided when the command was installed

Warning: the card's commands as well as any new commands must be executed with interrupts enabled. If you disable interrupts for too long, you might miss incoming characters.
You may use all the microprocessor's registers for your command.
The command must end with a RETF instruction (opcode 0CBH).
The MS-DOS MCXADDCM utility on the MCX diskette implements the ADDCM command. It is described in Appendix VI.5.

V.2 ADDTX (29h): Add a transmit function

OPCODE = 41 (29h)
This command adds a process to character transmission.
Warning: It is practically impossible to use this command properly without the MCXDEBUG[1] program.
Parameters 1 and 2 must contain the size of the function, in bytes.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	1	0	0	1
PARAMETER 1	Fnl	Fnl	Fnl	Fnl	Fnl	Fnl	Fnl	Fnl
PARAMETER 2	Fnh	Fnh	Fnh	Fnh	Fnh	Fnh	Fnh	Fnh
DATA ZONE	FUNCTION TO ADD TO MEMORY							

Fnl: PARAMETER #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #1 contains the least significant byte of the size of the function to add, in bytes.

Fnh: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #2 contains the most significant byte of the size of the function to add, in bytes.
Warning: the size of the function's code must not be greater than the amount of memory available.

DATA ZONE
The data zone of the dual-ported memory contains the code of the function to add. The data zone is located at a positive offset of 100 (64h) from the start of the dual-ported memory.

ERROR CODE
If the command executes normally, an error code of 0 is returned. If not, code 165 (A5h) is returned to indicate an error in the command.

The cause of the error may be:

- Not enough memory to install the function
- Function size equals zero
- Function already installed

[1] Optional development software from ACKSYS for the MCX and MCX-*Lite*/S cards.

IMPORTANT NOTE

It is absolutely imperative that the code of the command be compiled with an origin of 0000h.
Standard function and command calls for the card can be made with software interrupt INT 07, where the [AL] register contains the number of the function or command called (see Appendix VI.4).
Local data must be declared in the same segment as the function's code.
No intersegment references may be used.
When called, the [AL] register contains the character to send and the [BP] register indicates the channel to which it should be sent. When the function returns, the [AL] register must still contain the character to send.
Furthermore, the [AH] register must be saved.
Warning: the card's transmit function is executed with interrupts enabled. You should not disable interrupts for too long, as this might cause you to lose incoming characters.
You may use all the microprocessor's registers to execute your function except [AH].
The function must terminate with a RETF instruction (opcode 0CBH).

V.3 ADDRX (2Ah): Add a receive function

OPCODE = 42 (2Ah)
This command adds a process to character reception.
Warning: It is practically impossible to use this command properly without the MCXDEBUG program.
Parameters 1 and 2 must contain the size of the function, in bytes.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	1	0	1	0
PARAMETER 1	Fnl	Fnl	Fnl	Fnl	Fnl	Fnl	Fnl	Fnl
PARAMETER 2	Fnh	Fnh	Fnh	Fnh	Fnh	Fnh	Fnh	Fnh
DATA ZONE	FUNCTION TO ADD TO MEMORY							

Fnl: PARAMETER #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #1 contains the least significant byte of the size of the function to add, in bytes.

Fnh: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #2 contains the most significant byte of the size of the function to add.
Warning: the code size of the function must not be greater than available memory.

DATA ZONE
The data zone of the dual-ported memory must contain the code of the function to add. The data zone is located at a positive offset of 100 (64h) from the start of the dual-ported memory.

ERROR CODE
If the command executes normally, an error code of 0 is returned. If not, code 166 (A6h) is returned to indicate an error in the command.

The cause of the error may be:

- Not enough memory to install the function
- Function size equals zero
- Function already installed

IMPORTANT NOTE

It is absolutely imperative that the code of the command be compiled with an origin of 0000h.
Standard function and command calls for the card can be made with software interrupt INT 07, where the [AL] register contains the number of the function or command called (see Appendix VI.4).
Local data must be declared in the same segment as the function's code.
No intersegment references may be used.
When called, the [AL] register contains the character that was just received and the [BP] register indicates the channel on which it was received. When the function returns, the [AL] register must still contain the character to write to the buffer.
Furthermore, the [AH] register must be saved.
Warning: the card's receive function is executed with interrupts disabled. Your function cannot enable interrupts while it executes. You may use all the microprocessor's registers to execute your function except [AH].
The function must terminate with a RETF instruction (opcode 0CBH).

V.4 ALLOC (01h): Initialize buffer size

OPCODE = 1 (01h)
This command initialize the size of the receive, transmit and image buffers.

Parameters #2 and #3 specify the requested size.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	0	0	0	1
PARAMETER #1	0	#B	#B	#B	#B	#B	#B	#B
PARAMETER #2	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl
PARAMETER #3	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh
PARAMETER #4	TpB	TpB	0	0	0	0	0	0

#B: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These 7 bits indicate the buffer number for which memory should be reserved. It is written in hexadecimal, and must be between 1 and the number of lines installed for transmit/receive buffers and between 1 and 50 for screen images.

Lnl: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter contains the least significant byte of the size to reserve for the selected buffer.

Lnh: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter contains the most significant byte of the size to reserve for the selected buffer. Warning: the buffer size cannot exceed 64 Kbytes for TX or RX buffers, and the size of the mailbox minus 1 Kbyte for image buffers (31 Kbytes).

TpB: PARAMETER #4 DB7.DB6

These 2 bits define the type of buffer for which memory will be reserved, as per the table below:

DB7	DB6	Description
0	0	Transmit buffer
0	1	Receive buffer[1]
1	0	Screen image buffer
1	1	Illegal combination

[1] Warning: the minimum size for a receive buffer is 149 bytes. This represents: buffer size + 128 + 20 (where buffer size =1).

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 130 (82h) is returned to indicate an error in the command.

The cause of the error may be:

- transmit buffer # not between 1 and the number of channels installed (max = 64)
- receive buffer # not between 1 and the number of channels installed
- buffer # not between 1 and 50 for images
- Illegal combination on TpB (binary 11) parameter #4
- Image buffer size greater than 31 Kbytes or equal to 0
- TX or RX buffer size greater than 64 Kbytes or equal to 0
- Not enough memory
- Receive buffer size less than 149

IMPORTANT NOTE

At power-on, the buffers are programmed with the following default values:

- one 8192 byte transmit buffers for each installed channel
- one 512 byte receive buffers for each installed channel
- fifty 2000 byte screen image buffers for the whole card

These values are used until the DALOC command or a new allocation command is issued. The buffers are allocated in memory according to the buffer's order number[1].

Buffer order numbers are as follows:

- Transmit buffers: from 1 to 64 for the MCX card
  from 1 to 2 for the MCX-*Lite*/S card
- Receive buffers: from 65 to 128 for the MCX card
  from 3 to 4 for the MCX-*Lite*/S card
- Screen image buffers: from 129 to 178 for the MCX card
  from 5 to 54 for the MCX-*Lite*/S card

Warning: buffer memory should be reserved only once at power-on. This should not be executed again, because pointers may be moved and information lost. It is nonetheless possible to allocate memory for a buffer if its order number is greater than the number of buffers for which allocations have already been made.

[1] Warning: do not confuse the buffer number in parameter #1 with the buffer order number.

V.5 BDELE (06h): Erase TX buffer and stop

OPCODE = 6 (06h)
This command stops transmission of the buffer currently being sent, and totally erases the buffer memory.
This feature is particularly useful for VIDEOTEX servers, where programmers sometimes need to halt quickly cancel the display of previous pages when the user presses the "NEXT" key.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	0	1	1	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 contain the channel number whose buffer will be erased. This number must be coded in hexadecimal, and must be between 1 and the number of lines installed.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 135 (87h) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Corresponding buffer not reserved by the ALLOC command

IMPORTANT NOTE

The BDELE command does not trigger a "buffer completely empty" interrupt, even if this condition is programmed by the MINTR command.

V.6 BPARM (18h): Read buffer parameters

OPCODE = 24 (18h)
This command reads the parameters of a communication buffer.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	1	0	0	0
PARAMETER 1	0	#B	#B	#B	#B	#B	#B	#B
PARAMETER 2	TpB	TpB	0	0	0	0	0	0
DATA ITEM 1	Cnl	Cnl	Cnl	Cnl	Cnl	Cnl	Cnl	Cnl
DATA ITEM 2	Cnh	Cnh	Cnh	Cnh	Cnh	Cnh	Cnh	Cnh
DATA ITEM 3	SGl	SGl	SGl	SGl	SGl	SGl	SGl	SGl
DATA ITEM 4	SGh	SGh	SGh	SGh	SGh	SGh	SGh	SGh
DATA ITEM 5	ADl	ADl	ADl	ADl	ADl	ADl	ADl	ADl
DATA ITEM 6	ADh	ADh	ADh	ADh	ADh	ADh	ADh	ADh
DATA ITEM 7	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl
DATA ITEM 8	LNh	LNh	LNh	LNh	LNh	LNh	LNh	LNh
DATA ITEM 9	IDl	IDl	IDl	IDl	IDl	IDl	IDl	IDl
DATA ITEM 10	IDh	IDh	IDh	IDh	IDh	IDh	IDh	IDh
DATA ITEM 11	Ril	Ril	Ril	Ril	Ril	Ril	Ril	Ril
DATA ITEM 12	Rih	Rih	Rih	Rih	Rih	Rih	Rih	Rih

#B: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 indicate the buffer number. This number must be between 1 and the number of lines installed for transmit/receive buffers and between 1 and 50 for screen image buffers. This parameter must be in hexadecimal.

TpB: PARAMETER #2 DB7.DB6

These two bits determine the type of buffer to be queried, as per the table below:

DB7	DB6	Buffer type
0	0	Transmit buffer
0	1	Receive buffer
1	0	Screen image buffer
1	1	Illegal combination

Cnl: DATA ITEM #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the least significant byte of the FINBUF pointer used for managing transmit buffers.

Cnh: DATA ITEM #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the most significant byte of the FINBUF pointer used for managing transmit buffers.

SGl: DATA ITEM #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #3 represent the least significant byte of the segment where the buffer is located.

SGh: DATA ITEM #4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #4 represent the most significant byte of the segment where the buffer is located.

ADl: DATA ITEM #5 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #5 represent the least significant byte of the address where the buffer is located.

ADh: DATA ITEM #6 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #6 represent the most significant byte of the address where the buffer is located.

Lnl: DATA ITEM #7 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #7 represent the least significant byte of the length allocated by the ALLOC command.

Lnh: DATA ITEM #8 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #8 represent the most significant byte of the length allocated by the ALLOC command.

ID1: DATA ITEM #9 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #9 represent the least significant byte of the buffer index.

IDh: DATA ITEM #10 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #10 represent the most significant byte of the buffer index.

Ril: DATA ITEM #11 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #11 represent the least significant byte of the rotary index used to manage receive buffers.

Rih: DATA ITEM #12 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #12 represent the most significant byte of the rotary index used to manage receive buffers.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 148 (94h) is returned to indicate an error in the command.

The cause of the error may be:

- TX buffer # not between 1 and the number of channels installed (max = 64)
- RX buffer # not between 1 and the number of channels installed (max = 64)
- Image buffer # not between 1 and 50
- Buffer not allocated by ALLOC command
- Illegal combination in TpB (11) parameter 2

IMPORTANT NOTE

The buffer segment and address returned by the BPARM command indicate the buffer's actual address in memory. Use the RMEMO command to easily read the contents of a buffer.

V.7 BREAK (23h): Send a break to a communication line

OPCODE = 35 (23h)
This command sends a "Break" to a communication channel. The "Break" is a series of zeros sent over a certain time period.
There are two types of "Breaks":

Type of break	Duration
short	1 second
long	4 seconds

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	0	0	1	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	0	0	0	0	0	0	0	Tbk

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter indicates the channel number where the "Break" should be sent. It is in hexadecimal and must be between 1 and the number of channels installed.

Tbk: PARAMETER #2 DB0

This bit in parameter #2 indicates the type of "Break" to send:

Tbk	Type of break
0	short
1	long

IMPORTANT NOTE

This command can only be used correctly if the channel in question has already been initialized by the VINIT command.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 158 (9Eh) is returned to indicate an error in the command parameters.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Error in parameter number 2

V.8 BTEST (0Fh): Test the card's memory

OPCODE = 15 (0Fh)

This command performs a test of the card's memory. If you execute this command, all ongoing communications will be stopped and destroyed. The test stops as soon as an error is encountered. The results of this command are only meaningful if the error code returned is different from 0.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	1	1	1	1
DATA ITEM 1	AD1	AD1	AD1	AD1	AD1	AD1	AD1	AD1
DATA ITEM 2	ADh	ADh	ADh	ADh	ADh	ADh	ADh	ADh
DATA ITEM 3	SG1	SG1	SG1	SG1	SG1	SG1	SG1	SG1
DATA ITEM 4	SGh	SGh	SGh	SGh	SGh	SGh	SGh	SGh
DATA ITEM 5	Wr1	Wr1	Wr1	Wr1	Wr1	Wr1	Wr1	Wr1
DATA ITEM 6	Wrh	Wrh	Wrh	Wrh	Wrh	Wrh	Wrh	Wrh
DATA ITEM 7	Ww1	Ww1	Ww1	Ww1	Ww1	Ww1	Ww1	Ww1
DATA ITEM 8	Wwh	Wwh	Wwh	Wwh	Wwh	Wwh	Wwh	Wwh

ADl: DATA ITEM 1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits are the least significant byte of the address where the memory error was found.

AD2: DATA ITEM 2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits are the most significant byte of the address where the memory error was found.

SGl: DATA ITEM 3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits are the least significant byte of the segment where the memory error was found.

SGh: DATA ITEM 4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits are the most significant byte of the segment where the memory error was found.

Wrl: DATA ITEM 5 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits are the least significant byte of the 16 bit word read that was different from the word written.

Wrh: DATA ITEM 6 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits are the most significant byte of the 16 bit word read that was different from the word written.

Ww1: DATA ITEM 7 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits are the least significant byte of the 16 bit word that was written to the card's memory.

Wwh: DATA ITEM 8 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits are the most significant byte of the 16 bit word that was written to the card's memory.

ERROR CODE

If the command executed normally and no errors were encountered, an error code of zero is returned. If not, code 144 (90h) is returned and the information in the data zone is valid.

V.9 BTRAN (08h): Load and send a TX buffer

opcode = 8 (08h)
This command loads a transmit buffer and starts character transmission.
Parameters and #2 and #3 indicate the size of the data block.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	1	0	0	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl
PARAMETER 3	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh
DATA ZONE	DATA TO SEND							

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 indicate the channel number for the transmission. This number must be between 1 and the number of lines installed, and must be in hexadecimal.

Lnl: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter contains the least significant byte of the size of the data block to send.

Lnh: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter contains the most significant byte of the size of the data block to send.

Warning: the number of characters to write must never exceed the size of the buffer allocated by the ALLOC command. Furthermore, if the number of characters to transmit is greater than the size of the mailbox minus 1 Kbyte (or 31 Kbytes), you must execute several writes.

DATA ZONE

The data zone of the dual-ported memory contains the characters to be copied to the card. The data zone is located at a positive offset of 100 (64h) from the start of the dual ported memory.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 137 (89h) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of lines installed (max = 64)
- Size equal to 0 or greater than 31 Kbytes
- Size requested greater than the size allocated
- Size available less than the size requested
- Transmit buffer not reserved by the ALLOC command

IMPORTANT NOTE

You must do the following before executing this command:

- Allocate the buffer with the ALLOC command
- Initialize the communication parameters (VINIT command)
- Initialize the handshake parameters (VMODE command)

After the command has been issued, the card automatically begins sending characters to the requested line, and gradually frees space in the buffer. After a certain amount of time, a new transmission may be started before the previous one has finished, although you must make sure that there is enough free space in the buffer to receive the new block of information.
The transmit buffers are rotary.
Either the TFREE command or the TX COUNTER ZONE give information about the space available in the transmit buffer.
The card automatically manages handshaking using the protocol defined in the VMODE command.

V.10 CHDEF (0Ah): Define end-of-string characters

OPCODE = 10 (0Ah)
This command specifies the end-of-string characters. When these characters are detected, the card generates an interrupt on the PC bus to indicate that an entire phrase has been received. You may enable or disable this condition with the MINTR command.
Note that the end-of-string may be detected over one or two characters. For example, the "SEND" key on the Minitel returns two characters.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	1	0	1	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	0	0	0	NbS	NbS	NbS	NbS	NbS
PARAMETER 3	0	0	0	0	0	0	StL	StL
PARAMETER 4	St0	St0	St0	St0	St0	St0	St0	St0
PARAMETER 5	St1	St1	St1	St1	St1	St1	St1	St1
PARAMETER 6	0	0	0	0	0	0	StL	StL
PARAMETER 7	St0	St0	St0	St0	St0	St0	St0	St0
PARAMETER 8	St1	St1	St1	St1	St1	St1	St1	St1
PARAMETER 48	0	0	0	0	0	0	StL	StL
PARAMETER 49	St0	St0	St0	St0	St0	St0	St0	St0
PARAMETER 50	St1	St1	St1	St1	St1	St1	St1	St1

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 indicate the receive channel to test. This number must be between 1 and the number of channels installed, and must be in hexadecimal.

NbS: PARAMETER #2 DB4.DB3.DB2.DB1.DB0

This parameter specifies the number of character strings that will be used with the MINTR command to indicate to the CPU that a block of characters has been received. You may program up to16 strings of 1 or 2 characters. Characters in the string may, for example, be a CR or a CR, LF sequence, or perhaps a Minitel key that generates two ASCII codes.
The NbS parameter must be in hexadecimal, between 1 and 16.

StL: PARAMETER #3,6,9,12,15,18,21,24,27,30,33,36,39,42,45,48 DB1.DB0

This parameter is the number of characters in the string defined below. Acceptable values are 1 or 2; you must repeat the value for each string defined.

STn: PARAMETER #4+n, ...,49+n DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

The STn parameters contain the ASCII characters that define the end-of-string characters. For example, if StL Parameter #3 defines a length of 1, parameter #4 contains the character, while parameter #5 is not significant.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 140 (8Ch) is returned.

The cause of the error may be:

- Channel number not between 1 and the number of lines installed (max = 64)
- Number of strings equal to 0 or greater than 16
- Number of characters in a string greater than 2

IMPORTANT NOTE

The default values are:

ASCII character: 13 D (Carriage return) for each channel

V.11 CLRRX (1Dh): Erase a receive buffer

OPCODE = 29 (1Dh)
This command erases the entire receive buffer. All pointers concerning that buffer are reset to zero.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	1	1	0	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 represent the channel number whose receive buffer should be erased. This number, in hexadecimal, must be between 1 and the number of channels installed.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 151 (97h) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Specified buffer not reserved by the ALLOC command

V.12 DALOC (15h): Deallocate buffers

OPCODE = 21 (15h)
This command deletes all the channel buffers and the image buffers.
It requires no parameters, and always returns an error code of zero.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	0	1	0	1

IMPORTANT NOTE

After issuing this command, you must allocate buffer memory for each channel.
You may use this command to free all the memory allocated by default when the card is powered on.

V.13 ENDIT (14h): End of interrupt

OPCODE = 20 (14h)
This command acknowledges an interrupt generated by the card.
This command requires no parameters, and no error code is returned in order not to delete the "status" of the previous command.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	0	1	0	0

IMPORTANT NOTE
This command is equivalent to sending the ENDIT sequence in direct mode, as described earlier in this manual (see II.3.3). It is only available to provide compatibility with existing programs written for the MCC card.

No End-of-command interrupt is raised for this command.

V.14 FLASH (2Eh): Program the 256 Kbytes of Flash EPROM

OPCODE = 46 (2Eh)
This command lets you program the card's 256 Kbytes of FLASH memory.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	1	1	1	0
PARAMETER 1	0	0	0	0	0	0	0	Mde

Before programming the FLASH memory, you must transfer the code destined for the FLASH memory to the card's memory.
Use the MBOOT command to load the program into the card.
The FLASH memory is located from 0C0000H to 0FFFFFH in the first megabyte of the card's memory.
Since you cannot write to this memory directly, you must first copy the data into RAM.
The table below indicates which parts of RAM correspond to the FLASH memory which will contain the data.

RAM	FLASH
40000h - 4FFFFh	0C0000h - 0CFFFFh
50000h - 5FFFFh	0D0000h - 0DFFFFh
60000h - 6FFFFh	0E0000h - 0EFFFFh
70000h - 7FFFFh	0F0000h - 0FFFFFh

Also note that the memory from 38000h to 3FFFFh is overwritten when the FLASH command executes.
Warning: the use of this command destroys the contents of the character buffers. It takes approximately 6 seconds to execute. During those six seconds, no other processing is possible, and any commands or characters sent to the card will be ignored.

Mde: PARAMETER #1 DB0

This bit selects one of the two ways of programming the FLASH memory.
When DB0 equals 0, only the RAM between 40000h and 67FFFh will be reprogrammed. In this mode, the previous contents of the FLASH memory is automatically copied to 68000h to 7FFFFh in RAM.
This avoids overwriting all the contents of FLASH memory. If you make a mistake, the card will still be useable.
If DB0 equals 1, however, you may program all the FLASH memory. But beware of accidents!
This feature is useful for updating the basic software without changing the EPROM (using the MCXFLASH program).

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 169 (A9h) is returned to indicate an error in the command.

The cause of the error may be:

- Cannot erase or program the FLASH memory.
- Parameter #1 incorrect.

WARNING

If you update the software with this command (using the MCXFLASH program) the new code will not be used until you RESET the card.

V.15 GOADR (12h): Execute a program in memory

OPCODE = 18 (12h)
This command executes a program in memory at the address-segment specified in parameters 1, 2 , 3 and 4. The program must have been previously loaded with the MBOOT command.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	0	0	1	0
PARAMETER 1	SGl	SGl	SGl	SGl	SGl	SGl	SGl	SGl
PARAMETER 2	SGh	SGh	SGh	SGh	SGh	SGh	SGh	SGh
PARAMETER 3	ADl	ADl	ADl	ADl	ADl	ADl	ADl	ADl
PARAMETER 4	ADh	ADh	ADh	ADh	ADh	ADh	ADh	ADh

SGl: PARAMETER #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #1 contains the least significant byte of the segment where the program to be executed is located.

SGh: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #2 contains the most significant byte of the segment where the program to be executed is located.

ADl: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
These bits in parameter #3 contain the least significant byte of the program's start address.

ADh: PARAMETER #4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
These bits in parameter #4 contain the most significant byte of the program's start address.

WARNING
Parameters 1 to 4 do not make a FAR pointer, since the offset and segment parts are swapped.

IMPORTANT NOTE

Once the program is started, you lose control of the card during its execution. The task for receiving characters is not suspended, however, as long as the program that is executed does not execute a CLI (forbid interrupts) command. The real-time clock is stopped[1], however, but may be controlled by the user program.

The program must end with an RETF (0CBh) instruction in order to return control to the basic software.

Furthermore, we recommend that you do not initialize the STACK POINTER nor the STACK SEGMENT, which authorize from the start a 128 byte stack for the program that is executed. If this is not enough, you should declare a local stack and save the stack position in your first instructions.

ERROR CODE

The GOADR command does not return an error code. It does write a zero in the mailbox's status zone when the program that is executed returns control to the card's command interpreter.

RECOMMENDATIONS

It is relatively easy to develop custom software for the card using the MCXDEBUG[2] tool.

It includes a library of primitives that make it simple to access the card's various peripherals.

You can load .EXE programs if they are less than 32 Kbytes. You can also load programs in INTEL's MCS-86 format, limited only by the amount of memory on the card.

[1] For compatibility reasons with the IRMX-86 loader (INTEL's real-time OS).
[2] A debug tool called MCXDEBUG is available from our product list.

V.16 HNGUP (16h): Hang up modem

OPCODE = 22 (16h)
This command hangs up the modem for the channel in question. This is done by temporarily lowering the DTR signal.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	0	1	1	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	0	0	0	0	0	Tmp	Tmp	Tmp

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 contain the number of the channel which is to be hung up.
This number must be between 1 and 64 (depending on the number of channels installed), and must be in hexadecimal.

Tmp: PARAMETER #2 DB2.DB1.DB0

These three bits indicate the number of seconds during which the DTR signal must go low so the modem's electronics actually hang up the line. This must be a hexadecimal number between 1 and 7.
It may be programmed to meet the needs of different types of modems.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 159 (9Fh) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64).
- Time required outside acceptable limits

V.17 ILOAD (02h): Load a screen image

OPCODE = 2 (02h)
This command loads a screen image into memory. Parameters #2 and #3 contain the image size.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	0	0	1	0
PARAMETER 1	0	0	#I	#I	#I	#I	#I	#I
PARAMETER 2	Ln1	Ln1	Ln1	Ln1	Ln1	Ln1	Ln1	Ln1
PARAMETER 3	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh
DATA ZONE	IMAGE TO BE TRANSFERED							

#I: PARAMETER #1 DB5.DB4.DB3.DB2.DB1.DB0
These bits in parameter #1 contain the number of the screen image buffer to transfer to. This must be a hexadecimal number between 1 and 50.

Lnl: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
This parameter contains the least significant byte of the size of the image to transfer.

Lnh: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
This parameter contains the most significant byte of the size of the image to transfer.

Warning: the image size must not be greater than the size of the mailbox minus 1 Kbytes (or 31 Kbytes). It cannot be greater than the amount of memory reserved by the ALLOC command.

DATA ZONE
The data zone contains the image to transfer to the card's memory. The data zone starts at offset 100 (64h) from the start of the dual ported memory.

ERROR CODE
If the command executes normally, an error code of 0 is returned. If not, code 131 (83h) is returned to indicate an error in the command.

The cause of the error may be:

- Image buffer not reserved by the ALLOC command
- Size requested greater than size reserved
- Image # not between 1 and 50
- Image size equals 0

V.18 IMDWR (1Eh): Immediately send a string of characters

OPCODE = 30 (1Eh)
This command immediately sends a string of characters down a transmission channel. It has priority over normal transmissions initiated by the BTRAN command.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	1	1	1	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	Ln1	Ln1	Ln1	Ln1	Ln1	Ln1	Ln1	Ln1
PARAMETER 3	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh
DATA ZONE	DATA TO SEND							

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 represent the channel number to which the characters will be sent. This is a hexadecimal number between 1 and the number of channels installed.

Lnl: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #2 contain the least significant byte of the number of characters to send.

Lnh: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #3 contain the most significant byte of the number of characters to send.

The value defined by parameters #2 and #3 must be different from zero, and cannot be greater than the size of the mailbox minus 1 Kbyte (or 31 Kbytes).

IMPORTANT NOTE

This command does not terminate until the last character has been sent (including the stop bits). If the communication is blocked, the command will never terminate.
Furthermore, it is absolutely essential to have initialized the transmission channel.
Transmission takes place without the software handshake protocol selected. It is controlled instead by the hardware handshake if it was selected.
Finally, this command does not check if any BREAK is currently sent. Any characters transmitted during a BREAK will not be sent correctly; you should wait until the BREAK sequence is finished before transmitting.

DATA ZONE

The data zone in the dual-ported memory contains the characters to send down the channel. The data zone is located at a positive offset of 100 (64h) from the start of the dual ported memory.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 152 (98h) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Number of characters to send equal to 0 or greater than 31 Kbytes

V.19 ISEND (07h): Send a screen image

OPCODE = 7 (07h)
This command sends a screen image down a transmission channel. Parameter #2 contains the number of the image to send, and parameter #1 contains the channel number.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	0	1	1	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	0	0	#I	#I	#I	#I	#I	#I

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 contain the channel number to which the screen image will be sent. This must be a hexadecimal number between 1 and the number of lines installed.

#I: PARAMETER #2 DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #2 contain the number of the screen image to send. This must be a hexadecimal number between 1 and 50.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 136 (88h) is returned to indicate an error in the command.

The cause of the error may be:

- Image buffer not reserved by the ALLOC command
- Transmit buffer not reserved by the ALLOC command
- Image # not between 1 and 50
- Channel # not between 1 and the number of channels installed (max = 64)
- Available size less than image size

IMPORTANT NOTE

The mechanism used to send the screen image is related to the BTRAN command. The screen image to send is queued after the data already present in the transmit buffer. As a result, several screen images may be sent sequentially on the same transmission line, as long as the transmit buffer is big enough.
Before using the ISEND command, you should allocate the screen image buffer of the transmission channel to the appropriate size.

V.20 LTEST (0Eh): Test communication lines

OPCODE = 14 (0Eh)

This command automatically tests the communication lines (in RS232D and in RS422A). Before executing this command you must install a TEST connector to loop the signals for the test.

You should ignore the results of this command unless the error code returned is different from 0.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	1	1	1	0
DATA ITEM 1	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 2	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 3	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 4	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 5	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 6	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 57	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 58	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 59	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 60	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 61	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 62	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 63	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch
DATA ITEM 64	TxD	RxD	CTS	RTS	DTR	RIg	CDt	Bch

TxD: DATA ITEMS 1 to # channels installed DB7

If this bit equals zero, no error was detected. If not, an error occurred on the TX DATA signal.

RxD: DATA ITEMS 1 to # channels installed DB6

If this bit equals zero, no error was detected. If not, an error occurred on the RX DATA signal.

CTS: DATA ITEMS 1 to # channels installed DB5

If this bit equals zero, no error was detected. If not, an error occurred on the CLEAR TO SEND signal.

RTS: DATA ITEMS 1 to # channels installed DB4

If this bit equals zero, no error was detected. If not, an error occurred on the REQUEST TO SEND signal.

DTR: DATA ITEMS 1 to # channels installed DB3

If this bit equals zero, no error was detected. If not, an error occurred on the DATA TERMINAL READY signal.

RIg: DATA ITEMS 1 to # channels installed DB2

If this bit equals zero, no error was detected. If not, an error occurred on the RING INDICATOR signal.

CDt: DATA ITEMS 1 to # channels installed DB1

If this bit equals zero, no error was detected. If not, an error occurred on the CARRIER DETECT signal.

Bch: DATA ITEMS 1 to # channels installed DB0

If this bit equals zero, no error was detected. If not, the character received is different from the character sent.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 143 (8Fh) is returned to indicate a faulty signal. The bit corresponding to the bad signal is set to 1 in the result word.

IMPORTANT NOTE

The information concerning uninstalled lines is not significant and should be ignored.
After using this command, you must reinitialize all the command channels with the VINIT command. The test destroys any channel configuration information you may have programmed.

V.21 MBOOT (05h): Load a program into memory

OPCODE = 5 (05h)
This command loads a program into the card's memory by blocks.
Put the address where the code block should be loaded (SEG:ADR) in parameters 1, 2, 3 and 4; put the code size in parameters 5 and 6.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	0	1	0	1
PARAMETER 1	SGl	SGl	SGl	SGl	SGl	SGl	SGl	SGl
PARAMETER 2	SGh	SGh	SGh	SGh	SGh	SGh	SGh	SGh
PARAMETER 3	ADl	ADl	ADl	ADl	ADl	ADl	ADl	ADl
PARAMETER 4	ADh	ADh	ADh	ADh	ADh	ADh	ADh	ADh
PARAMETER 5	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl
PARAMETER 6	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh
DATA ZONE	BLOCK OF CODE TO LOAD INTO MEMORY							

SGl: PARAMETER #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #1 contains the least significant byte of the segment where the code block should be loaded.

SGh: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #2 contains the most significant byte of the segment where the code block should be loaded.

ADl: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #3 contains the least significant byte of the address where the code block should be loaded in the card's memory.

ADh: PARAMETER #4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #4 contains the most significant byte of the address where the code block should be loaded in the card's memory.

Lnl: PARAMETER #5 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #5 contains the least significant byte of the size of the code block.

Lnh: PARAMETER #6 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #6 contains the most significant byte of the size of the code block.

Warning: the code block size cannot be greater than the size of the mailbox minus 1 Kbyte (or 31 Kbytes). If not, an error is returned.

DATA ZONE

The data zone of the dual-ported memory contains the code block to copy to the card.
Reminder: the data zone is located at a positive offset of 100 (64h) from the start of the dual ported memory.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 134 (86h) is returned to indicate an error in the command.

The cause of the error is:

- Code block size greater than 31 Kbytes or equal to 0

IMPORTANT NOTE

If the sum of the program size and the start address is greater than 64 Kbytes (0FFFFH), any code with addresses greater than 10000H will be written to address 0 and above in the same segment.

V.22 MINTR (0Ch): "Event" interrupt conditions

OPCODE = 12 (0Ch)
This command enables or masks the card's interrupt conditions.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	1	1	0	0
PARAMETER 1	Mde	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	IT7	IT6	IT5	IT4	IT3	IT2	IT1	IT0

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter indicates the channel number to be initialized. This is a hexadecimal number between 1 and the number of channels installed.

Mde: PARAMETER #1 DB7

This parameter lets you modify the meaning of IT2 (see table below).

ITn: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #2 indicate the events that will trigger an interrupt on the PC bus. The table below describes the interrupt event for each IT #.

IT #	Event that triggers the interrupt	Comments
IT0	Each character received	Except for handshake characters. The character counter is decremented by 1 at each interrupt.
IT1 (STCNT=1)	First character received	See STCNT command
IT1 (STCNT>1)	Received a sequence of STCNT characters	See STCNT command
IT2 (Mde=0)	Receive buffer full	
IT2 (Mde=1)	Receive buffer no longer empty	
IT3	Character string programmed by the CHDEF command recognized	See CHDEF command
IT4	Timeout	See STTMO command
IT5	Receive error detected	Parity error, frame error, or characters overwritten in the controller's FIFO buffer
IT6	CTS, CD, or RI signal changed state, or BREAK sequence detected	
IT7	Transmit buffer completely emptied (including stop bit)	

NOTE

Each of the card's channels can have a different operating mode for interrupts. You can also select several conditions for a given channel.
A new set of conditions simply replaces any previous conditions, however.
Furthermore, by default no interrupts are enabled.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 142 (8Eh) is returned to indicate an error in the parameters.

The cause of the error is:

- Channel # not between 1 and the number of channels installed (max = 64)

V.23 MSIZE (2Ch): Read memory configuration

OPCODE = 44 (2Ch)

This command returns information concerning the card's memory configuration:

- Memory size
- First free address in 1st segment
- Memory available for buffers
- Mailbox segment on card side
- Mailbox size

This command requires no parameters. It has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	1	1	0	0
DATA ITEM 1	Msl	Msl	Msl	Msl	Msl	Msl	Msl	Msl
DATA ITEM 2	Msh	Msh	Msh	Msh	Msh	Msh	Msh	Msh
DATA ITEM 3	Ffl	Ffl	Ffl	Ffl	Ffl	Ffl	Ffl	Ffl
DATA ITEM 4	Ffh	Ffh	Ffh	Ffh	Ffh	Ffh	Ffh	Ffh
DATA ITEM 5	Mal	Mal	Mal	Mal	Mal	Mal	Mal	Mal
DATA ITEM 6	Mah	Mah	Mah	Mah	Mah	Mah	Mah	Mah
DATA ITEM 7	Sbl	Sbl	Sbl	Sbl	Sbl	Sbl	Sbl	Sbl
DATA ITEM 8	Sbh	Sbh	Sbh	Sbh	Sbh	Sbh	Sbh	Sbh
DATA ITEM 9	Bsl	Bsl	Bsl	Bsl	Bsl	Bsl	Bsl	Bsl
DATA ITEM 10	Bsh	Bsh	Bsh	Bsh	Bsh	Bsh	Bsh	Bsh

Msl: DATA ITEM #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This contains the least significant byte of the total amount of memory installed on the card.

Msh: DATA ITEM #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This contains the most significant byte of the total amount of memory installed on the card.

IMPORTANT NOTE

The value provided by Msl and Msh contains a number of paragraphs in hexadecimal (one paragraph = 16 bytes).

Ffl: DATA ITEM #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the least significant byte of the first address available in the card's segment 0. This address does not take into account buffer memory.

Ffh: DATA ITEM #4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the most significant byte of the first address available in the card's segment 0. This address does not take into account buffer memory.

Mal: DATA ITEM #5 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the least significant byte of the amount of memory available for buffers.

Mah: DATA ITEM #6 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the most significant byte of the amount of memory available for buffers.

IMPORTANT NOTE

The value provided by Mal and Mah contains a number of paragraphs in hexadecimal (one paragraph = 16 bytes). This value is updated as memory is allocated by the ALLOC command.

Sbl: DATA ITEM #7 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the least significant byte of the mailbox segment, as seen from the card side.

Sbh: DATA ITEM #8 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the most significant byte of the mailbox segment, as seen from the card side.

Bsl: DATA ITEM #9 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the least significant byte of the size of the card's mailbox.

Bsh: DATA ITEM #10 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the most significant byte of the size of the card's mailbox

ERROR CODE

The MSIZE command always returns 0.

V.24 NOPER (2Bh): NOP command (no operation)

OPCODE = 43 (2Bh)

This command executes a null command on the card, which then generates an end-of-command interrupt.

This command has no parameters and always returns an error code of zero.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	1	0	1	1

USAGE

This command may be used to free the interrupt mechanism when the system is waiting for an interrupt to empty the command stack.

V.25 RDBUF (09h): Read a receive buffer

OPCODE = 9 (09h)
This command reads the contents of a receive buffer.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	1	0	0	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl
PARAMETER 3	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh
PARAMETER 4	0	0	0	0	0	Typ	Typ	Typ
PARAMETER 5	0	0	0	0	NbC	NbC	NbC	NbC
PARAMETER 6	Chr	Chr	Chr	Chr	Chr	Chr	Chr	Chr
PARAMETER 7	Chr	Chr	Chr	Chr	Chr	Chr	Chr	Chr
PARAMETER 8	Chr	Chr	Chr	Chr	Chr	Chr	Chr	Chr
PARAMETER 9	Chr	Chr	Chr	Chr	Chr	Chr	Chr	Chr
PARAMETER 20	Chr	Chr	Chr	Chr	Chr	Chr	Chr	Chr
PARAMETER 21	Chr	Chr	Chr	Chr	Chr	Chr	Chr	Chr
DATA ZONE	DATA READ							

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 contain the number of the channel whose buffer will be read.
This is a hexadecimal number between 1 and the number of channels installed.

Lnl: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter contains the least significant byte of the number of characters to read.

Lnh: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter contains the most significant byte of the number of characters to read. Warning: the number of characters to read must not exceed the size reserved by ALLOC command. If the number of characters received is greater than the size of the mailbox minus 1 Kbyte (or 31 Kbytes), then you must perform several reads.

Typ: PARAMETER #4 DB2.DB1.DB0

These bits in parameter #4 specify the kind of read to perform, as per the following table:

DB2	DB1	DB0	Type of read
0	0	0	Read (n) characters
0	0	1	Non-destructive read of (n) characters
0	1	0	Read entire buffer
0	1	1	Read last character
1	0	0	Read (n) characters and copy to the mailbox even if there are not enough characters
1	0	1	Read all characters up to a string

Other combinations of DB2, DB1 and DB0 are illegal and return an error.

Type 0: Read (n) characters.
In this case, parameters #2 and #3 must contain the number of characters to be read. This command modifies the card's internal pointers: it is therefore a destructive read. If the number of characters requested is greater than the number of characters in the buffer when the command is issued, the card returns error code 139 and parameters #2 and #3 are automatically loaded with the number of characters available in the buffer. In this case, the characters are not copied to the mailbox.

Type 1: Read (n) characters (non destructive).
This operation is identical to the previous one, except the card's internal pointers are not modified. This read is therefore non-destructive: the characters are not removed from the buffer.

Type 2: Read the entire buffer.
In this case, all characters in the buffer are read and removed (up to the mailbox size minus 1 Kbyte). Parameters #2 and #3 are automatically loaded with the total number of characters in the buffer.

Type 4: Read (n) characters and copy them to the mailbox, even if the number of characters requested is not currently present in the buffer.
In this case, parameters #2 and #3 contain the number of characters to be read. This command modifies the card's internal pointers: it is therefore a destructive read. If the number of characters requested is greater than the number of characters in the buffer when the command is issued, the card returns error code 139 and parameters #2 and #3 are automatically loaded with the number of characters available in the buffer. In this case, the data zone of the mailbox contains these characters.

Type 5: Read all characters in the buffer until the programmed string is recognized.

This command reads the contents of a receive buffer until the string programmed with the NbC and Chr parameters is recognized. Characters in the buffer are copied to the mailbox, as well as the stop string.
This read operation is of destructive type; after execution, the number of characters copied is indicated by parameters 2 and 3.
If the string is not found, the number of characters copied equals 0.
You do not have to initialize parameters 2 and 3 when issuing type 5 RDBUF commands.
Warning: searching for the string and therefore copying characters depends on the size of the mailbox.

NbC: PARAMETER #5 DB3.DB2.DB1.DB0

This parameter contains the number of characters in the string to be recognized (type 5 command).
This must be a number between 1 and 16.

Chr: PARAMETERS #6 to 21 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These parameters contain the ASCII codes of the characters in the string to be recognized (type 5 command).

DATA ZONE

The data zone of the dual-ported memory contains the characters read by the command. The data zone is located at a positive offset of 100 (64h) from the start of the dual ported memory.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 138 (8Ah) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Number of characters to read equal to 0 or greater than 31 Kbytes
- Number of characters to read greater than the reserved size
- Receive buffer not reserved by the ALLOC command
- Illegal combination for type of read
- Number of characters in string equal to 0 or greater than 16 (type 5 command)

If the error code equals 139 (8Bh), the number of characters requested is not currently in the buffer, and parameters #2 and #3 indicate the number of characters available. For a type 0 read, the characters are not copied to the mailbox. They are copied to the mailbox for a type 4 read, however.

V.26 RELRP (13h): Read codes, revisions, and identifications

OPCODE = 19 (13h)
This command returns the following information:

- Revision of the firmware
- Confidential code
- Card ID code
- Number of lines installed
- Total memory size
- Microprocessor type
- Microprocessor clock frequency
- If a math coprocessor is installed
- Type of connection unit
- If an auxiliary MCX-PWS power supply is installed

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	0	0	1	1
DATA ITEM 1	VFW	VFW	VFW	VFW	VFW	VFW	VFW	VFW
DATA ITEM 2	Cf1	Cf1	Cf1	Cf1	Cf1	Cf1	Cf1	Cf1
DATA ITEM 3	Cf2	Cf2	Cf2	Cf2	Cf2	Cf2	Cf2	Cf2
DATA ITEM 4	Cf3	Cf3	Cf3	Cf3	Cf3	Cf3	Cf3	Cf3
DATA ITEM 5	Cf4	Cf4	Cf4	Cf4	Cf4	Cf4	Cf4	Cf4
DATA ITEM 6	0	1	0	0	1	1	0	1
DATA ITEM 7	0	1	1	0	0	0	1	1
DATA ITEM 8	0	1	1	1	1	0	0	0
DATA ITEM 9	0	Nlg	Nlg	Nlg	Nlg	Nlg	Nlg	Nlg
DATA ITEM 10	Fpu[1]	Cpu	Pws[2]	Bpt[3]	Mem	Mem	Mem	Mem
DATA ITEM 11	Clk	Clk	Clk	Clk	Clk	Clk	Clk	Clk
DATA ITEM 12	VMD	VMD	VMD	VMD	VMD	VMD	VMD	VMD
DATA ITEM 13	VDG	VDG	VDG	VDG	VDG	VDG	VDG	VDG
DATA ITEM 14	VBI	VBI	VBI	VBI	VBI	VBI	VBI	VBI
DATA ITEM 15	RIN	RIG	RIF	RIE	RID	RIC	RIB	RIA

VFW: DATA ITEM #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

The bits in data item #1 encode the version number of the card's basic software. For example, code 11h indicates software revision 1.1.

Cf1: DATA ITEM #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the card's confidential code. This value is masked, and stored in the card.

[1] Fpu = 0 for the MCX-*Lite*/S card.
[2] Pws = 0 for the MCX-*Lite*/S card
[3] Bpt = 1 for the MCX-*Lite*/S card

Cf2: DATA ITEM #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the card's second confidential code. This value is masked in the card's hardware.

Cf3: DATA ITEM #4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the card's third confidential code. This value is masked, and stored in the card.

Cf4: DATA ITEM #5 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This is the card's fourth confidential code. This value is masked, and stored in the card.
These four bytes are written when the card is manufactured. They represent a confidential code which lets users protect their software by refusing to run if a different confidential code is found.
When the purchaser did not request a confidential code, the command returns code 87654321.

DATA ITEMS #6,7,8 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

Data items 6,7, and 8 contain the following ASCII characters:

Mcx

They provide a quick way to detect if the card is in the machine. They also let the programmer automatically determine the card's address and segment.

Nlg: DATA ITEM #9 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits indicate the number of channels installed on the card. This number ranges from 1 to 64 modulo 8[1].

Mem: DATA ITEM #10 DB3.DB2.DB1.DB0

These bits provide the card's total memory capacity in megabytes.

Bpt: DATA ITEM #10 DB4

This bit indicates the type of MCX-BP connection unit attached to the MCX-00 card:

Bpt = 0 ..... Single oscillator unit
Bpt = 1 ..... Double oscillator unit

[1] This number must always equal 2 for the MCX-*Lite*/S card.

Pws: DATA ITEM #10 DB5

This bit indicates whether or not the MCX-PWS auxiliary power supply is connected to the MCX-00 card.

Pws = 0 ..... No power supply installed
Pws = 1 ..... Power supply installed

Cpu: DATA ITEM #10 DB6

This bit indicates the type of microprocessor installed on the card.

Cpu = 0 ..... 386SX
Cpu = 1 ..... 486 SLC

Fpu: DATA ITEM #10 DB7

This bit indicates whether or not a 80387SX coprocessor is installed on the MCX-00 card.

Fpu = 0 ..... No coprocessor installed
Fpu = 1 ..... Coprocessor installed

Clk: DATA ITEM #11 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #11 indicate the clock speed of the card's processor. The value is returned in hexadecimal, and it represents Mhz.

VMD: DATA ITEM #12 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #12 contain the version of the MCXDOS extension.

VDG: DATA ITEM #13 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in data item #13 contain the version number of the resident part of MCXDEBUG.

VBI: DATA ITEM #14 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

The bits in data item #14 contain the version number of the card's BIOS.

ERROR CODE

The RELRP command always returns an error code equal to 0.

RIN: DATA ITEM #15 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

Each bit, if set to 1, indicates that the corresponding connection unit is able to use the Ring Indicator signals on each of its eight channels.

V.27 RINIT (19h): Reinitialize the card

OPCODE = 25 (19h)
This command resets the card's software.

It has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	1	0	0	1
PARAMETER 1	0	0	0	0	0	0	0	Typ

Typ: PARAMETER #1 DB0

This bit in parameter #1 indicates the type of reinitialization to perform.

If Typ = 1, the card resets all variables to zero. The communication lines and modem cards are reset to their initial state, and will have to be initialized again. After the command, the card is waiting for the "`RUN 01`" start code.

If Typ = 0, the memory zones are reset to zero and the buffers return to their default configurations (see the ALLOC command). The communication controllers are not reset; their status is preserved.

WARNING

Since the card does not return an end-of-command interrupt if Typ = 1, the programmer can determine if the card is ready to receive the "`RUN 01`" start code by reading address 0 in the mailbox. Once the card is ready, it writes byte 0Fh at address 0 and the character string "`MCX IS READY`" at address 100 (64h).

ERROR CODE

If the command executes normally, an error code of 0 is returned (only if Typ = 0).
If not, code 154 (9Ah) is returned to indicate an error in the command.

The cause of the error is:

- Error in parameter #1

V.28 RMEMO (11h): Dump a block of memory

OPCODE = 17 (11h)
This command lists a block of memory.
The block's segment is supplied in parameters #1 and #2, while parameters #3 and #4 contain the block's address.
Parameters #5 and #6 contain the size of the block to read.

The information is transferred into the card's data zone.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	0	0	0	1
PARAMETER 1	SGl	SGl	SGl	SGl	SGl	SGl	SGl	SGl
PARAMETER 2	SGh	SGh	SGh	SGh	SGh	SGh	SGh	SGh
PARAMETER 3	ADl	ADl	ADl	ADl	ADl	ADl	ADl	ADl
PARAMETER 4	ADh	ADh	ADh	ADh	ADh	ADh	ADh	ADh
PARAMETER 5	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl
PARAMETER 6	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh
DATA ZONE	LISTED DATA BLOCK							

SGl: PARAMETER #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #1 contains the least significant byte of the segment of the block to be listed.

SGh: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #2 contains the most significant byte of the segment of the block to be listed.

ADl: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #3 contains the least significant byte of the address of the block to be listed.

ADh: PARAMETER #4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #4 contains the most significant byte of the address of the block to be listed.

Lnl: PARAMETER #5 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #5 contains the least significant byte of the size of the block to be read.

Lnh: PARAMETER #6 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #6 contains the most significant byte of the size of the block to be read.

DATA ZONE

The data zone of the dual-ported memory contains the data block that was read. The data zone is located at a positive offset of 100 (64h) from the start of the dual ported memory.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 146 (92h) is returned.

The cause of the error may be:

- Data block size greater than 31 Kbytes
- Data block size equal to zero

IMPORTANT NOTE

If the sum of the size requested and the start address is greater than 64 Kbytes (0FFFFH), the contents of addresses greater than 10000H will be replaced with the contents of address zero and above in the same segment.

V.29 RSMDE (2Dh): Initialize RS232D or RS422A mode

OPCODE = 45 (2Dh)
This command selects the operating mode for a communication channel.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	1	1	0	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	0	0	0	0	0	0	0	Mde

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 represent the channel number to activate in receive mode. This must be a hexadecimal number between 1 and the number of channels installed.

Mde: PARAMETER #2 DB0

This bit selects either the RS232D or RS422A mode.

DB0	Mode
0	RS232D
1	RS422A

NOTE

By default all communication channels are programmed in RS232D mode.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 168 (A8h) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Parameter #2 incorrect

V.30 RSTAT (0Dh): Read the status of communication channels

OPCODE = 13 (0Dh)
This command dynamically reads the status of all communication channels.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	1	1	0	1
DATA ITEM 1	TXe	RXr	CDt	CTS	RIg	Par	Ovr	Brk
DATA ITEM 2	TXe	RXr	CDt	CTS	RIg	Par	Ovr	Brk
	. . . [1]							
DATA ITEM 61	TXe	RXr	CDt	CTS	RIg	Par	Ovr	Brk
DATA ITEM 62	TXe	RXr	CDt	CTS	RIg	Par	Ovr	Brk
DATA ITEM 63	TXe	RXr	CDt	CTS	RIg	Par	Ovr	Brk
DATA ITEM 64	TXe	RXr	CDt	CTS	RIg	Par	Ovr	Brk

The various states are written consecutively for each channel in the dual-ported memory's data zone. The data zone is located at a positive offset of 100 (64h) from the dual-ported memory base address.

TXe: DATA ITEMS 1 to # of channels installed DB7

If this bit is set to 1, the transmitter for the corresponding channel is ready to receive a new character. Note that the communication processors can store up to four transmit characters internally.

RXr: DATA ITEMS 1 to # of channels installed DB6

If this bit is set to 1, a character was received by the receiver, and may be read by the card's CPU. Note that the communication processors can store up to three receive characters internally.

CDt: DATA ITEMS 1 to # of channels installed DB5

If this bit is set to 1, the "CARRIER DETECT" signal is present on the channel's connector.

CTS: DATA ITEMS 1 to # of channels installed DB4

If this bit is set to 1, the "CLEAR TO SEND" signal is present on the channel's connector.

[1] The data below only concerns the MCX card (depending on the number of channels installed).

RIg: DATA ITEMS 1 to # of channels installed DB3

If this bit is set to 1, the "RING INDICATOR" signal is present on the channel's connector.

PAR: DATA ITEMS 1 to # of channels installed DB2

If this bit is set to 1, a parity or a frame error has been detected on the corresponding channel's receiver.

Ovr: DATA ITEMS 1 to # of channels installed DB1

If this bit is set to 1, characters have been overwritten in the receiver. This may occur when the CPU could not read the incoming characters fast enough, and other characters were received that erased the preceding ones.

Brk: DATA ITEMS 1 to # of channels installed DB0

If this bit is set to 1, a "BREAK" signal was detected on the channel's receiver. The "BREAK" signal is low for approximately 250 ms for a short "BREAK" or 3.5 seconds for a long "BREAK" (DEC VT100 standards).

ERROR CODE

The RSTAT command always returns an error code of 0.

IMPORTANT NOTE

Information returned for uninstalled channels is not significant. The events corresponding to bits DB0, DB1 and DB2 are memorized as soon as they are detected, and only reset to zero by the RSTAT command.

V.31 RXCNT (1Bh): Read number of characters received

OPCODE = 27 (1Bh)
This command returns the number of characters in the receive buffer specified by parameter #1. The information returned is written to the dual-ported memory's data zone.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	1	0	1	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
DATA ITEM 1	Nbl	Nbl	Nbl	Nbl	Nbl	Nbl	Nbl	Nbl
DATA ITEM 2	Nbh	Nbh	Nbh	Nbh	Nbh	Nbh	Nbh	Nbh

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 indicate the channel number. This is a hexadecimal number between 1 and the number of channels installed.

Nbl: DATA ITEM #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits represent the least significant byte of the number of characters in the selected buffer.

Nbh: DATA ITEM #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits represent the most significant byte of the number of characters in the selected buffer.

IMPORTANT NOTE

You may also read the number of characters in the receive buffer by reading the mailbox directly. The character counters are updated in real time in the dual-ported memory. This is a faster method for reading the information, and does not keep the card busy.
Each counter is a 16 bit WORD divided into least significant byte (8 bits) and most significant byte (8 bits).

The first counter is located at address 32512 (7F00H hexadecimal) relative to the segment where the card is located; the second counter is located at address 32514, the third at 32516, etc.

The value of the counter for channels not installed is constant, and always equal to zero.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 150 (96h) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Associated buffer not reserved by the ALLOC command

V.32 RXENB (04h): Receive enable or disable

OPCODE = 4 (04h)
This command enables or disables reception for a channel.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	0	1	0	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	0	0	0	0	0	0	0	A/D

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter 1 represent the channel number to enable in receive mode. This must be a hexadecimal number between 1 and the number of channels installed.

A/D: PARAMETER #2 DB0

This bit determines if the channel should be enabled or disabled.

DB0	Receive
1	Enabled
0	Disabled

IMPORTANT NOTE

No channels are enabled by default.
This command is only valid if the channel to be enabled has already been initialized by the VINIT command. When you enable a receive channel, the timeout value, if it has been programmed, is automatically reloaded into the counter in order to avoid resending the STTMO command after an RXENB = OFF, RXENB = ON sequence.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 133 (85h) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Corresponding buffer not reserved by the ALLOC command

V.33 STCNT (0Bh): Define the size of received blocks

OPCODE = 11 (0Bh)
This command defines the size of character blocks that will trigger an interrupt.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	1	0	1	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	Cnl	Cnl	Cnl	Cnl	Cnl	Cnl	Cnl	Cnl
PARAMETER 3	Cnh	Cnh	Cnh	Cnh	Cnh	Cnh	Cnh	Cnh

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 specify the receive channel number. This number must be between 1 and the number of channels installed, in hexadecimal.

Cnl: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #2 contain the least significant byte of the number of characters that must be in the buffer to generate an interrupt.

Cnh: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #3 contain the most significant byte of the number of characters that must be in the buffer to generate an interrupt.

IMPORTANT NOTE

This function is only operational if the corresponding interrupt has been selected with the MINTR command.
An interrupt is generated each time the number of characters in the buffer equals "modulo" the number of characters programmed by this command.
The number contained in parameters #2 and #3 must be encoded in hexadecimal, and must not exceed the size allocated for the buffer.
The default number of characters for this function is 80.

WARNING

If the programmed value is 1, the interrupt generated will not operate as usual.
In that case the interrupt means: Interrupt following the receipt of the first character.
This feature is useful for determining the start of a session on a communication channel.

ERROR CODE

The system returns an error code of 0 if the command executes normally. If not, error code 141 (8Dh) is returned.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Number of characters greater than the reserved size for this buffer

V.34 STIME (10h): Set the time on the card

OPCODE = 16 (10h)
This command updates the time on the card. This time is used to tell the system when the last communication took place, to determine which terminals or peripherals are in sleep mode (TMRRP command). Time is kept even when the card is powered off.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	0	0	0	0
PARAMETER 1	Hr	Hr	Hr	Hr	Hr	Hr	Hr	Hr
PARAMETER 2	Min	Min	Min	Min	Min	Min	Min	Min
PARAMETER 3	Sec	Sec	Sec	Sec	Sec	Sec	Sec	Sec

Hr: PARAMETER #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #1 contains hours in BCD format.

Min: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #2 contains minutes in BCD format.

Sec: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0
Parameter #3 contains seconds in BCD format.

ERROR CODE
If the command executes normally, an error code of 0 is returned. If not, code 145 (91h) is returned to indicate an error in the command.

The cause of the error may be:

- Invalid hour
- Invalid minutes
- Invalid seconds

V.35 STSIG (24h): Manually control DTR and RTS

OPCODE = 36 (24h)
This command lets you control the DTR and RTS signals manually.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	1	0	0	1	0	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	0	0	0	0	0	0	DTR	RTS

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter indicates the channel number for which you will control the DTR and RTS bits. This is a hexadecimal number between 1 and the number of channels installed.

RTS: PARAMETER #2 DB0

This bit enables or disables the RTS signal (Request to send):

RTS	Status
0	Disabled
1	Enabled

DTR: PARAMETER #2 DB1

This bit enables or disables the DTR signal (Data terminal ready):

DTR	Status
0	Disabled
1	Enabled

IMPORTANT NOTE

This command can only be used correctly if the channel has been initialized previously by the VINIT command. Furthermore, the VINIT and HNGUP commands as well as the hardware handshake dynamically modify the status of DTR and RTS.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 160 (A0h) is returned to indicate an error in the command parameters.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Error in parameter #2

V.36 STTMO (1Fh): Initialize receive timeout

OPCODE = 31 (1Fh)
This command defines the timeout value for receiving characters.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	1	1	1	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	Tmo	Tmo	Tmo	Tmo	Tmo	Tmo	Tmo	Tmo

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 contain the number of the receive channel whose timeout will be configured. This number must be between 1 and the number of channels installed, in hexadecimal.

Tmo: PARAMETER #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

Parameter #2 represents the timeout value for this receive channel.
This is a hexadecimal value, in seconds. The maximum authorized value is 255 seconds.
The default value is 10 seconds.

IMPORTANT NOTE

This function has no effect unless the corresponding interrupt has been programmed by the MINTR command.
An interrupt is triggered each time the corresponding timeout register reaches zero.
The default timeout value is 10 seconds.

Warning: The countdown starts as soon as this value is written, if the channel has been validated in receive mode. If an RXENB = ON sequence is sent, the channel's timeout counter is loaded with the value programmed by the STTMO command.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 153 (99h) is returned.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Timeout value equal to zero

V.37 TFREE (1Ah): Read space available in a TX buffer

OPCODE = 26 (1Ah)

This command returns the space available for the transmit channel indicated in parameter #1. The information returned is written in the dual-ported memory's data zone.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	1	0	1	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
DATA ITEM 1	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl	Lnl
DATA ITEM 2	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh	Lnh

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #1 contain the number of the transmission channel. This is a hexadecimal number between 1 and the number of channels installed.

Lnl: DATA ITEM #1 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This data item contains the least significant byte of the space available in the selected buffer.

Lnh: DATA ITEM #2 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This data item contains the most significant byte of the space available in the selected buffer.

IMPORTANT NOTE

You can also read the space available in a transmit buffer by directly reading the mailbox. Counters are updated in real time in the dual-ported memory. This is a faster method for reading the information, and does not tie up the card. Each counter is a WORD (16 bits) organized with the least significant byte (8 bits) followed by the most significant byte (8 bits). It indicates the space available in the TX buffer.

The address of the first counter is 32640 (7F80H in hexadecimal) relative to the card's segment. The second counter is located at address 32642, the third at address 32644, and so forth.

The counters for uninstalled channels do not vary.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 149 (95h) is returned to indicate an error in the command.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Associated buffer not reserved by the ALLOC command

NOTE

Since the transmit buffers are rotary, space available is automatically updated as characters are transmitted.

V.38 TMRRP (1Ch): Read the time of last communication

OPCODE = 28 (1Ch)

This command tells you the time the last events were recorded (interrupt conditions only). This makes it easy to tell which terminals or peripherals are in sleep mode (a function used in multistation operating systems). Information is provided for each communication channel, and is written in the dual-ported memory.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	1	1	1	0	0
DATA ITEM 1	H01	H01	H01	H01	H01	H01	H01	H01
DATA ITEM 2	M01	M01	M01	M01	M01	M01	M01	M01
DATA ITEM 3	H02	H02	H02	H02	H02	H02	H02	H02
DATA ITEM 4	M02	M02	M02	M02	M02	M02	M02	M02
DATA ITEM 5	H03	H02	H02	H02	H02	H02	H02	H02
DATA ITEM 6	M03	M02	M02	M02	M02	M02	M02	M02
DATA ITEM 7	H04	H02	H02	H02	H02	H02	H02	H02
DATA ITEM 8	M04	M02	M02	M02	M02	M02	M02	M02
DATA ITEM 9	H05	H02	H02	H02	H02	H02	H02	H02
DATA ITEM 10	M05	M02	M02	M02	M02	M02	M02	M02
DATA ITEM 11	H06	H02	H02	H02	H02	H02	H02	H02
DATA ITEM 12	M06	M02	M02	M02	M02	M02	M02	M02

Hxx: DATA ITEM DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This data item contains the hour of the last event on channel xx, in BCD format.

Mxx: DATA ITEM DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This data item contains the minutes of the last event on channel xx, in BCD format.

ERROR CODE

This command always returns an error code equal to 0.

IMPORTANT NOTE

Information returned for uninstalled channels is not significant. Furthermore, the time buffers are only updated by events that trigger interrupts. In other words, characters received do not modify the table contents unless they correspond to a condition programmed by the MINTR command. This is done so as not to degrade the card's performance.

V.39 VINIT (00h): Initialize communication parameters

OPCODE = 0 (00h)
This command initializes a channel's communication parameters.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	0	0	0	0
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	E/O	Par	Stp	Stp	BRx	BRx	BTx	BTx
PARAMETER 3	Clk	0	0	Spd	Spd	Spd	Spd	Spd
PARAMETER 4	Tcl	Tcl	Tcl	Tcl	Tcl	Tcl	Tcl	Tcl
PARAMETER 5	Tch	Tch	Tch	Tch	Tch	Tch	Tch	Tch

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

These 7 bits contain the number of the channel to be initialized. This must be a hexadecimal number between 1 and the number of channels installed.

BTx: PARAMETER #2 DB1.DB0

These 2 bits specify the number of bits per character sent.

DB1	DB0	Number of bits per character sent
0	0	5
1	0	6
0	1	7
1	1	8

BRx: PARAMETER #2 DB3.DB2

These 2 bits specify the number of bits per character received.

DB3	DB2	Number of bits per character received
0	0	5
1	0	6
0	1	7
1	1	8

If 5,6 or 7 bits are selected, the card automatically sets non-significant bits to 0.

Stp: PARAMETER #2 DB5.DB4

These 2 bits specify the number of stop bits per character received.

DB5	DB4	Number of stop bits
0	0	Illegal combination
0	1	1
1	0	1.5
1	1	2

Par: PARAMETER #2 DB6

This bit enables or disables the calculation and check of the parity bit.

DB6	Parity
1	Enable
0	Disable

E/O: PARAMETER #2 DB7

This bit selects the type of parity calculation (odd or even). This is only significant if parity is enabled (Bit D6 in parameter #2 set to 1).

DB7	Parity
1	even
0	odd

NOTE

If all bits in parameter #2 are set to 0, the default initialization is used. In that case you do not need to define parameters 3, 4 and 5. The default values are as follows:

- Length of characters sent/received 8 bits
- Number of stop bits per character 1 bit
- Parity Disabled
- Transmit/receive speed 9600 bps

Spd: PARAMETER #3 DB4.DB3.DB2.DB1.DB0

These bits select the receive/transmit speed for the channel as per the following tables. This is a hexadecimal number between 0 and 17.

Decimal value	Hexadecimal value	Speed in bits/second for 16 MHz clock
0	(00h)	50
1	(01h)	75
2	(02h)	110
3	(03h)	134.5
4	(04h)	150
5	(05h)	300
6	(06h)	600
7	(07h)	1200
8	(08h)	1800
9	(09h)	2000
10	(0Ah)	2400
11	(0Bh)	3600
12	(0Ch)	4800
13	(0Dh)	7200
14	(0Eh)	9600
15	(0Fh)	19200
16	(10h)	38400
17	(11h)	Defined by programmer

Selection #17 lets you program speeds other than those in the table, as well as the frequency of the SCC clock. You can write another counter factor to generate line speed.
Write this new value in parameters #4 and #5 (least significant byte then most significant byte).
This selection can manage speeds up to 38400 bits/second.

Clk: PARAMETER #3 DB7

This parameter determines the base clock used for selection 17 (custom speed) to obtain lower error percentages for certain speeds.

Clk	Clock frequency (in Mhz)
0	16
1	14.7456

IMPORTANT NOTE

Warning: the base clock selected for a communication channel determines the speed for all the channels of the MCX-BP unit or the *Lite*-SERIAL extension. All the channels must then be programmed with speed 17 (the standard speeds are no longer valid).

Tcl: PARAMETER #4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter contains the least significant byte of the new counter value. It is only valid if parameter #3 contains the value 17.

Tch: PARAMETER #5 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter contains the most significant byte of the new counter value. It is only valid if parameter #3 contains the value 17.

To obtain the new counter value based on the communication speed, perform the following calculation:

$$\text{Counter factor} = \frac{\text{clock speed (Hz)}}{32 \times \text{speed (bits / second)}} - 2$$

Example: for 9600 bits per second, calculate the counter factor as follows:

$$\text{Counter factor} = \frac{16.10^6}{32 \times 9600} - 2 = 50$$

This represents 32 hexadecimal. Therefore you write 32 in parameter #4 (least significant byte) and 0 in parameter #5 (most significant byte).

If the operation returns a value that is not an integer, you should round off to the nearest whole number. If the difference between the integer and the decimal part of the number is too large, the transmission may contain a non-negligible percentage of errors.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 129 (81h) is returned to indicate an error in the parameters.

The cause of the error may be:

- Channel # not between 1 and the number of channels installed (max = 64)
- Illegal combination in Stp (00) parameter #2
- Selected speed greater than 17

WARNING

The VINIT command cannot be used during reception or transmission, because it may desynchronize the receivers and transmitters.

V.40 VMODE (03h): Initialize Handshake, Echo and Encryption

OPCODE = 3 (03h)
This command initializes the parameters concerning the handshake, echo mode and the data encoding method used for a communication channel.

This command has the following format:

	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
OPCODE	0	0	0	0	0	0	1	1
PARAMETER 1	0	C#	C#	C#	C#	C#	C#	C#
PARAMETER 2	0	Vid	Dif	Cry	Ech	HS2	HS1	HS0
PARAMETER 3	ch0	ch0	ch0	ch0	ch0	ch0	ch0	ch0
PARAMETER 4	chH	chH	chH	chH	chH	chH	chH	chH
PARAMETER 5	K1	K1	K1	K1	K1	K1	K1	K1
PARAMETER 6	K2	K2	K2	K2	K2	K2	K2	K2

C#: PARAMETER #1 DB6.DB5.DB4.DB3.DB2.DB1.DB0

This parameter indicates the channel number to be initialized. This must be a hexadecimal number between 1 and the number of channels installed.

HSn: PARAMETER #2 DB2.DB1.DB0

These bits in parameter #2 select the handshake, as described below:

DB2	DB1	DB0	Handshake mode
0	0	0	No handshake (1)
0	0	1	XON-XOFF Mode(2)
0	1	0	ACK-NACK[1] Mode(3)
0	1	1	STX-ETX[1] Mode(4)
1	0	0	DTR-CTS Mode (5)
1	0	1	Special character Mode (6)
1	1	0	IXANY Mode (7)
1	1	1	RTS-CTS Mode (8)

NOTE
Mode 8 is available from EPROM version 1.8 on.

In modes 2, 3, 4 and 6, the first character restarts communications, while the second halts the current communication. In mode 7, XOFF is the stop character, but any other character will restart communications.
In modes 5 and 8 mode, only the line control signals are used to synchronize communications. During transmission, the CTS signal must be set to 1 before the card can transmit; transmission is blocked if CTS equals 0. In reception, the DTR (or RTS) signal is set to 1 if the card is ready to receive characters. As soon as the buffer is full, this signal is automatically set to zero.

[1] The ACK-NACK and STX-ETX characters are not used here as string delimiters, but simply as start and stop characters (as in XON-XOFF mode).

In all cases, the card sets the DTR and RTS signals to 1 during the VINIT command.

If you use mode 6 (special characters), you must write the character used to restart communications in parameter #3, and the character to halt communications in parameter #4. These two characters must be ASCII encoded. If you do not write these two characters, XON and XOFF are used by default.

Ech: PARAMETER #2 DB3

This bit in parameter #2 determines if the card should automatically echo characters received. If Ech equals 1, an echo is sent; no echo is sent if this bit equals 0.
Use the DB5 bit in parameter #2 (Dif) to determine if the echo will be immediate or delayed.

Cry: PARAMETER #2 DB4

This bit in parameter #2 determines if the card should encrypt the data sent and received on the channel.

DB4	Status
0	Encryption enabled
1	Encryption disabled

If encryption mode is enabled, parameters #5 and #6 indicate encryption keys 1 and 2; their value may be between 0 and 255.

Warning: the handshake characters are also encoded. If echo mode is selected, however, the character returned as an echo will be decoded.

Information concerning the encryption algorithm is available on request.

Dif: PARAMETER #2 DB5

This bit in parameter #2 determines if characters are echoed immediately or if the echo is delayed.
If Dif equals 0 the echo is immediate, i.e. the character received is immediately resent.
If Dif equals 1, the character received is written to the end of the transmit buffer. It will only be resent after the characters already in the transmit buffer have been sent.
Characters to be echoed in delayed mode are temporarily stored in a 128 byte buffer, in order to free the transmit buffer. If the transmit buffer is completely full, the echo characters cannot be added to it.

As a result, if the transmit buffer remains completely full during the time required to receive 128 characters, some of the echo characters already in the buffer will be overwritten. Warning: the system provides no warning before overwriting the characters!

Furthermore, the character counters available in the transmit buffers are updated by the echo in delayed mode. As a result, be careful when reading these counters. If they indicate that you can send a character, and if an echo character is added at the same time, an error will be generated in the BTRAN command.

Vid: PARAMETER #2 DB6

When using delayed echo mode, this bit in parameter #2 specifies if echo characters should be filtered in Videotex mode. If Vid equals zero, no filtering is performed, and all characters received are resent. If Vid equals one, the system filters characters as follows.

- The two characters following a 1Bh, 3Ah (ESC ":") sequence are deleted, as well as the ESC ":".
- The three characters following a 1Bh, 3Bh (ESC ";") sequence are deleted, as well as the ESC ";".
- The two characters following a 1Fh are deleted, as well as the 1Fh.
- The character following the 13h (CTRL<S>) is deleted, as well as the CTRL<S>.

This filtering removes "undesirable" MINITEL sequences
Note that the Videotex filtering mode only operates if delayed echo is enabled.

chO: PARAMETER #3 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #3 contain the ASCII character that restarts communications.

chH: PARAMETER #4 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #4 contain the ASCII character that halts communications.

K1: PARAMETER #5 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #5 make up key #1 used in the card's encryption algorithm. It must be between 0 and 255.

K2: PARAMETER #6 DB7.DB6.DB5.DB4.DB3.DB2.DB1.DB0

These bits in parameter #6 make up key #2 used in the card's encryption algorithm. It must be between 0 and 255.

NOTE

Note that the handshake mode operates both when receiving and transmitting characters.
Reception is suspended when the number of characters received equals the maximum size of the receive buffer minus a margin of 20 characters.
The preceding configurations may differ for each channel, and may be selected independently.
By default, all channels are programmed in XON-XOFF mode; echo and encryption modes are disabled.
The software handshake modes are only operational if the channels in question have been receive enabled (RXENB command).

WARNING

If flow control is disabled (no handshake protocol, or synch characters not received), the receive buffer may overflow; an internal mechanism avoids overwriting characters in the buffer, and halts any incoming characters.
In that case, the number of characters read by the RXCNT command equals the number set by the ALLOC command, and the data read corresponds to the data that entered the receive buffer first.

ERROR CODE

If the command executes normally, an error code of 0 is returned. If not, code 132 (84h) is returned to indicate an error in the parameters.

The cause of the error is :

- Channel # not between 1 and the number of channels installed (max = 64)
- Illegal combination selected

VI. APPENDICES

VI.1 Command summary sorted by opcode

OPCODES	COMMAND	DESCRIPTION
00 / 00h	VINIT	Initialize communication parameters
01 / 01h	ALLOC	Initialize buffer size
02 / 02h	ILOAD	Load a screen image
03 / 03h	VMODE	Initialize handshake, echo and encryption
04 / 04h	RXENB	Enable or disable reception
05 / 05h	MBOOT	Load a program into memory
06 / 06h	BDELE	Clear the transmit buffer and stop
07 / 07h	ISEND	Send a screen image to a channel
08 / 08h	BTRAN	Load a transmit buffer and send
09 / 09h	RDBUF	Read a receive buffer
10 / 0Ah	CHDEF	Define string parameters
11 / 0Bh	STCNT	Define block sizes
12 / 0Ch	MINTR	Initialize the card's interrupt mode
13 / 0Dh	RSTAT	Read status of communication channels
14 / 0Eh	LTEST	Automatic test of communication channels
15 / 0Fh	BTEST	Automatic test of buffer memory
16 / 10h	STIME	Set time on card
17 / 11h	RMEMO	Read a block of memory
18 / 12h	GOADR	Execute a program in memory
19 / 13h	RELRP	Read codes, revisions, identification
20 / 14h	ENDIT	End-of-interrupt command (acknowledge)
21 / 15h	DALOC	Deallocate buffer memory
22 / 16h	HNGUP	Hang up modem
23 / 17h	XXXXX	Reserved
24 / 18h	BPARM	Read buffer parameters
25 / 19h	RINIT	Reinitialize card
26 / 1Ah	TFREE	Calculate space available in TX buffer
27 / 1Bh	RXCNT	Read number of characters in RX buffer
28 / 1Ch	TMRRP	Read time of last communications
29 / 1Dh	CLRRX	Clear receive buffer
30 / 1Eh	IMDWR	Immediate transmission of character string
31 / 1Fh	STTMO	Define receive timeout
32 / 20h	XXXXX	Reserved
33 / 21h	XXXXX	Reserved
34 / 22h	XXXXX	Reserved
35 / 23h	BREAK	Send a "break" to a transmission channel
36 / 24h	STSIG	Manual control of DTR and RTS signals
37 / 25h	XXXXX	Reserved (Debugging monitor)
38 / 26h	XXXXX	Reserved
39 / 27h	XXXXX	Reserved
40 / 28h	ADDCM	Add a command to standard command set
41 / 29h	ADDTX	Add a transmit character function
42 / 2Ah	ADDRX	Add a receive character function
43 / 2Bh	NOPER	NOP command (no operation)
44 / 2Ch	MSIZE	Read amount of memory available

OPCODE (contd.)	COMMAND (contd.)	DESCRIPTION (contd.)
45 / 2Dh	RSMDE	Initialize RS232D or RS422A mode
46 / 2Eh	FLASH	Program the FLASH memory
50 / 32h	XXXXX	Reserved

VI.2 Error code summary

The table below provides the error codes for each command.[1].

DEC =	Bit 7	Bit 6[2]	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
000	**0**	**0**	0	0	0	0	0	0	No error
129	**1**	**0**	0	0	0	0	0	1	VINIT error
130	**1**	**0**	0	0	0	0	1	0	ALLOC error
131	**1**	**0**	0	0	0	0	1	1	ILOAD error
132	**1**	**0**	0	0	0	1	0	0	VMODE error
133	**1**	**0**	0	0	0	1	0	1	RXENB Error
134	**1**	**0**	0	0	0	1	1	0	MBOOT Error
135	**1**	**0**	0	0	0	1	1	1	BDELE error
136	**1**	**0**	0	0	1	0	0	0	ISEND error
137	**1**	**0**	0	0	1	0	0	1	BTRAN error
138	**1**	**0**	0	0	1	0	1	0	RDBUF error
139	**1**	**0**	0	0	1	0	1	1	RDBUF error
140	**1**	**0**	0	0	1	1	0	0	CHDEF error
141	**1**	**0**	0	0	1	1	0	1	STCNT error
142	**1**	**0**	0	0	1	1	1	0	MINTR error
143	**1**	**0**	0	0	1	1	1	1	LTEST error
144	**1**	**0**	0	1	0	0	0	0	BTEST error
145	**1**	**0**	0	1	0	0	0	1	STIME error
146	**1**	**0**	0	1	0	0	1	0	RMEMO error
147	**1**	**0**	0	1	0	0	1	1	Reserved
148	**1**	**0**	0	1	0	1	0	0	BPARM error
149	**1**	**0**	0	1	0	1	0	1	TFREE error
150	**1**	**0**	0	1	0	1	1	0	RXCNT error
151	**1**	**0**	0	1	0	1	1	1	CLRRX error
152	**1**	**0**	0	1	1	0	0	0	IMDWR error
153	**1**	**0**	0	1	1	0	0	1	STTMO error
154	**1**	**0**	0	1	1	0	1	0	RINIT error
155	**1**	**0**	0	1	1	0	1	1	Reserved
156	**1**	**0**	0	1	1	1	0	0	Reserved
157	**1**	**0**	0	1	1	1	0	1	Reserved
158	**1**	**0**	0	1	1	1	1	0	BREAK error
159	**1**	**0**	0	1	1	1	1	1	HNGUP error
160	**1**	**0**	1	0	0	0	0	0	STSIG error
161	**1**	**0**	1	0	0	0	0	1	Reserved
162	**1**	**0**	1	0	0	0	1	0	RELRP error
163	**1**	**0**	1	0	0	0	1	1	Reserved
164	**1**	**0**	1	0	0	1	0	0	ADDCM error
165	**1**	**0**	1	0	0	1	0	1	ADDTX error
166	**1**	**0**	1	0	0	1	1	0	ADDRX error
167	**1**	**0**	1	0	0	1	1	1	DEBUG error
168	**1**	**0**	1	0	1	0	0	0	RSMDE error

[1] This return code is written in the mailbox's STATUS zone (address 03).
[2] Note that this bit is always set to "0" in an end-of-command STATUS.

DEC =	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	
169	**1**	**0**	1	0	1	0	0	1	FLASH error
189	**1**	**0**	1	1	1	1	0	1	Unauthorized command[1]
190	**1**	**0**	1	1	1	1	1	0	Unknown command
191	**1**	**0**	1	1	1	1	1	1	Command not implemented

If bit 6 is set to 1, the interrupt received is not an end-of-command interrupt, but an event interrupt.
If bit 7 is set to zero and bit 6 is also at 0, the command that was issued previously executed with no errors.

[1] This code means a reserved command was used

VI.3 Using the MCXADDCM program (MS-DOS only)

The MCXADDCM program is only available under MS-DOS.
It lets you add new commands to the card's standard command set.
New commands may themselves call any of the standard commands, as well as any of the card's functions.
It is thus possible to create MACRO commands which include some of the card's standard commands and which perform special processing.
The commands must be compiled with base address 0000h; no INTERSEGMENT references may exist in the program.
.BIN files are loaded; i.e. .EXE files that have been converted with the MS-DOS EXE2BIN utility.
The EXE2BIN utility verifies that the program makes no INTERSEGMENT references.
Finally, after installing the new command, the MCXADDCM program returns the new OPCODE and its associated error code.

Examples provided on the diskette:

STCNT.ASM	Retroactive version of the STCNT command
CHDEF.ASM	Retroactive version of the CHDEF command
INIMAC.ASM	Initialize command sequence
MCXCALC.ASM	Calculation command that may be executed by the card

The STCNT.BAS program illustrates how to load new commands in a program, or into your driver, without using the MCXADDCM program.

Note:

You may create new commands using other programming languages than assembly language, as long as you can control the command's start address and as long as you can make sure that you code and data reside in the same segment.
The ASMCMD.BAT program lets you assemble, link and generate your own .BIN with the MASM, LINK et EXE2BIN programs sold by Microsoft Inc.

Important:

The MCXADDCM program must be started when the card has not yet been initialized (before the « `RUN 01` »), and therefore without using the MS-DOS driver.
Warning: the RINIT command deletes the commands previously installed.

VI.4 List of INT 07 interrupt functions

The table below lists the commands and functions that may be executed using the INT 07 interrupt with the ADDCM, ADDTX and ADDRX commands.
The commands may be called internally without creating an end-of-command interrupt. Parameters are passed through the mailbox, as for external commands. Note that the mailbox segment as seen from the card is different from the segment seen from the PC.
For functions (function number greater than or equal to 80h), you must load values in the microprocessor's registers.

List of INT 07 commands

OPCODE	COMMAND	DESCRIPTION
00 / 00h	VINIT	Initialize communication parameters
01 / 01h	ALLOC	Initialize buffer sizes
02 / 02h	ILOAD	Load a screen image
03 / 03h	VMODE	Initialize handshake, echo and encryption
04 / 04h	RXENB	Enable or disable a receive channel
05 / 05h	MBOOT	Load a program into memory
06 / 06h	BDELE	Clear a transmission buffer and stop
07 / 07h	ISEND	Send an image to a channel
08 / 08h	BTRAN	Load a transmit buffer and send
09 / 09h	RDBUF	Read a receive buffer
10 / 0Ah	CHDEF	Define string parameters
11 / 0Bh	STCNT	Define block size
12 / 0Ch	MINTR	Initialize the card's interrupt mode
13 / 0Dh	RSTAT	Read the status of the communication channels
14 / 0Eh	LTEST	Automatic test of communication channels
15 / 0Fh	BTEST	Automatic test of buffer memory
16 / 10h	STIME	Set time on card
17 / 11h	RMEMO	Read a block of memory
18 / 12h	GOADR	Execute a program in memory
19 / 13h	RELRP	Read codes, revisions, identification
21 / 15h	DALOC	Deallocate memory buffers
22 / 16h	HNGUP	Hang up modem
24 / 18h	BPARM	Read buffer parameters
25 / 19h	RINIT	Reinitialize card
26 / 1Ah	TFREE	Calculate space available in TX buffer
27 / 1Bh	RXCNT	Read number of characters in RX buffer
28 / 1Ch	TMRRP	Read time of most recent communications
29 / 1Dh	CLRRX	Clear receive buffer
30 / 1Eh	IMDWR	Immediately send a character string
31 / 1Fh	STTMO	Define receive timeout values
35 / 23h	BREAK	Send a "break" to a transmission channel
36 / 24h	STSIG	Manually control DTR and RTS signals
41 / 29h	ADDTX	Add a character transmit function
42 / 2Ah	ADDRX	Add a character receive function

OPCODES (contd.)	COMMANDS (contd.)	DESCRIPTION (contd.)
43 / 2Bh	NOPER	NOP command (no operation)
44 / 2Ch	MSIZE	Read the amount of memory available
45 / 2Dh	RSMDE	Initialize RS232D or RS422A mode
46 / 2Eh	FLASH	Program the FLASH memory

List of INT 07 interrupt functions

QUEUE_LOAD Function (80h)
This function loads the parameters of the interrupt spooler in order to generate an interrupt in the host system.

Input:

[BP] Register = Channel number (between 1 and the number of channels installed)
[DL] Register = IT condition (will be written to address 81 / 51h)
[BX] Register = Number of char. (will be written to 82 / 52h and 83 /53h)
[DH] Register = IT parameter (will be written to 84 / 54h)

Output:

[BP] Register = Unchanged
[DL] Register = Unchanged
[BX] Register = Unchanged
[DH] Register = Unchanged
Other registers = Unchanged

REGISTER_WRITE Function (81h)
This function writes a value in an internal SCC register.

Input:

[CH] Register = SCC unit number (between 1 and number of channels installed)
[CL] Register = Register number where write will be performed
[BL] Register = Value to write to register defined in [CL]

Output:

[CH] Register = Unchanged
[CL] Register = Unchanged
[BL] Register = Unchanged
Other registers = Unchanged

Warning: for this sub-program, you must disable interrupts so that not interrupt handler can access the SCC between the writing of the register number and the writting of the register contents.

REGISTER_READ Function (82h)
This function reads the contents of one of the SCC's internal registers.

Input:

[CH] Register = SCC unit number (between 1 and the number of channels installed)
[CL] Register = Register number to be read

Output:

[CH] Register = Unchanged
[CL] Register = Unchanged
[BL] Register = Contents of internal SCC register
Other registers = Unchanged

Warning: for this sub-program, you must disable interrupts so that not interrupt handler can access the SCC between the writing of the register number and the reading of the register contents.

WRITE_SCC_EOI Function (83h)
This function acknowledges interrupts generated by the SCC.

Input:

[CH] Register = SCC unit number (between 1 and the number of channels installed)

Output:

[CH] Register = Unchanged
Other registers = Unchanged

CHARACTER_IN Function (84h)
This function reads a character in one of the SCCs.

Input:

[CH] Register = SCC unit number (between 1 and the number of channels installed)

Output:

[CH] Register = Unchanged
[CL] Register = 00 if the character was read, 255 if not (no characters ready)
[BL] Register = Character read or register contents for read #0
Other registers = Unchanged

Since this sub-program calls the REGISTER_READ sub-program, the same precautions are necessary concerning interrupts.

CHARACTER_OUT Function (85h)
This function sends a character to one of the SCCs.

Input:

[CH] Register = SCC unit number (between 1 and the number of channels installed)
[BL] Register = Character to send to the SCC

Output:

[CH] Register = Unchanged
[CL] Register = 00 if the character was sent, 255 if not (SCC not ready)
[BL] Register = Contents of register 0 if character not sent, or else unchanged
Other registers = Unchanged

Since this sub-program calls the REGISTER_READ sub-program, the same precautions are necessary concerning interrupts.

LD_BUFF_PARM Function (86h)
This function provides the descriptor address of the specified buffer.

Input:

[AH] Register = Buffer number
[BL] Register = Buffer type

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	B#	B#	B#	B#	B#	B#	B#

Buffer number

TX or RX buffers (1-number of channels installed)
Image buffers (1-50)

DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
BTp	BTp	0	0	0	0	0	0

Buffer type

0	0	Transmit buffer
0	1	Receive buffer
1	0	Image buffer
1	1	Illegal combination

Output:

[AH] Register	=	Unchanged
[BX] Register	=	Buffer descriptor address
Other registers	=	Unchanged

The buffer descriptor contains 13 bytes. Its address is always relative to segment 0000h.

The 13 bytes have the following structure:

Byte 1	:	Buffer allocation flag
Bytes 2 & 3	:	Index 1
Bytes 4 & 5	:	Buffer segment
Bytes 6 & 7	:	Buffer address
Bytes 8 & 9	:	Buffer size
Bytes 10 & 11	:	Index 2
Bytes 12 & 13	:	Index 3

If the buffer allocation flag equals 1, the specified buffer is allocated and therefore the following information is valid.
Bytes 2 & 3, 10 & 11, 12 & 13 are pointers to the buffer. Their use varies for transmit, receive or image buffers.

Transmit buffers

Bytes 1 & 2 (INDEX 1)	.....	Output pointer
Bytes 10 & 11 (INDEX 2)	.....	Input pointer
Bytes 12 & 13 (INDEX 3)	.....	Not used

Receive buffers

Bytes 1 & 2 (INDEX 1)	.....	Character counter
Bytes 10 & 11 (INDEX 2)	.....	Output pointer
Bytes 12 & 13 (INDEX 3)	.....	Input pointer

Image buffers

Bytes 1 & 2 (INDEX 1)	.....	Not used
Bytes 10 & 11 (INDEX 2)	.....	Not used
Bytes 12 & 13 (INDEX 3)	.....	Not used

VI.5 Sample initialization sequence

The following example illustrates a typical initialization sequence for the card:

```
WRITE RUN 01                        ; 1st level initialization
ISSUE RELRP                         ; Read number of channels
ISSUE STIME (HH:MM:SS)              ; Set time on card
ISSUE VINIT (01,00)                 ; Channel #1, default value
ISSUE VINIT (02,00)                 ; Channel #2, default value
ISSUE RSMDE (01,00)                 ; Channel #1, RS232D mode
ISSUE RSMDE (02,00)                 ; Channel #2, RS232D mode
ISSUE DALOC                         ; Deallocate all channel buffers
ISSUE ALLOC (T,01,30000)            ; Init buffer 1, TX, 30000 characters
ISSUE ALLOC (T,02,30000)            ; Init buffer 2, TX, 30000 characters
ISSUE ALLOC (R,01,20000)            ; Init buffer 1, RX, 20000 characters
ISSUE ALLOC (R,02,20000)            ; Init buffer 2, RX, 20000 characters
ISSUE ALLOC (I,11,2000)             ; Image #11, 2000 characters
ISSUE ALLOC (I,12,2000)             ; Image #12, 2000 characters
ISSUE ILOAD (11,MENU1)              ; Load image 11
ISSUE ILOAD (12,MENU2)              ; Load image 12
ISSUE VMODE (01,XON/XOFF)           ; Channel 1, XON/XOFF, echo off
ISSUE VMODE (02,XON/XOFF)           ; Channel 2, XON/XOFF, echo off
ISSUE CHDEF (01,01,CR,LF)           ; Channel 1, end-of-string = CR, LF
ISSUE CHDEF (02,01,CR,LF)           ; Channel 2, end-of-string = CR, LF
ISSUE STCNT (01,1000)               ; Channel 1, block size 1000 char
ISSUE STCNT (02,1000)               ; Channel 2, block size 1000 char
ISSUE STTMO (01,60)                 ; Channel 1, timeout 60 sec.
ISSUE STTMO (02,10)                 ; Channel 2, timeout 10 sec.
ISSUE MINTR (01,CNT,DEF,TMO)        ; INT on Count, string, T-O.
ISSUE MINTR (02,CNT,DEF,TMO)        ; INT on Count, string, T-O.
ISSUE RXENB (01)                    ; Receive enable channel 1
ISSUE RXENB (02)                    ; Receive enable channel 2
END                                 ; End of initialization
```

After this initialization sequence, the card is ready to receive characters on channels #1 and #2; it can also send characters on the same channels.
Screen images #11 and #12 have been loaded with menus 1 and 2 respectively; they may be sent to lines #1 and #2.
The following conditions have been programmed in receive mode for each of the first two channels:

- Interrupt if the number of characters in the buffer is modulo 1000 characters
- Interrupt if the CR, LF character string is detected
- Interrupt channel 1 if no character has been received after 60 seconds
- Interrupt channel 2 if no character has been received after 10 seconds

The transmit / receive parameters are programmed with the default values: 9600 bits/second, 8 bits/character, no parity, 1 stop bit.

VI.6 Sample program under interrupts

The following example illustrates how to use the card's command processor under interrupts with Borland's TURBO PASCAL version 5.

SET_INT_VEC, GET_INT_VEC, MEM and PORT are standard TURBO PASCAL functions.
This example uses an 8 channel MCX card whose mailbox address is D0000H and which uses IRQ3.

```
unit mcx;

interface

procedure mcx_open;
procedure mcx_close;
procedure mcx_command(can:byte; comm:byte; param1:byte);

implementation

uses dos;


                        { Card parameters }


const  irq_vec    = $0b;     { Interrupt vector           }
       irq_mask   = $08;     { Interrupt mask             }
       irq_base   = $20;     { Address of 1st 8259        }
       mcx_adr    = $d000;   { Address of card            }
       nb_canal   = 8;       { Number of channels         }


                       { Mailbox structure }


       start      = 0;       { Offset to validate cmd     }
       endit      = 1;       { Offset for ENDIT function  }
       opcode     = 2;       { Offset of command code     }
       status     = 3;       { Offset of command status   }
       canal      = 4;       { Offset of channel number   }
       mcx_p1     = 5;       { Offset of 1st parameter    }
       it_chan    = 80;      { Offset of IT channel       }
       it_cond    = 81;      { Offset of IT description   }
       mcx_data   = 100;     { Offset of data zone        }
```

```
                          { Card commands }

       vinit      = 0;       { VINIT command                    }
       alloc      = 1;       { ALLOC command                    }
       iload      = 2;       { ILOAD command                    }
       vmode      = 3;       { VMODE command                    }
       rxenb      = 4;       { RXENB command                    }
       mboot      = 5;       { MBOOT command                    }
       bdele      = 6;       { BDELE command                    }
       isend      = 7;       { ISEND command                    }
       btran      = 8;       { BTRAN command                    }
       rdbuf      = 9;       { RDBUF command                    }
       chdef      = 10;      { CHDEF command                    }
       stcnt      = 11;      { STCNT command                    }
       mintr      = 12;      { MINTR command                    }
       rstat      = 13;      { RSTAT command                    }
       ltest      = 14;      { LTEST command                    }
       btest      = 15;      { BTEST command                    }
       stime      = 16;      { STIME command                    }
       rmemo      = 17;      { RMEMO command                    }
       goadr      = 18;      { GOADR command                    }
       relrp      = 19;      { RELRP command                    }
       daloc      = 21;      { DALOC command                    }
       hngup      = 22;      { HNGUP command                    }
       bparm      = 24;      { BPARM command                    }
       rinit      = 25;      { RINIT command                    }
       tfree      = 26;      { TFREE command                    }
       rxcnt      = 27;      { RXCNT command                    }
       tmrrp      = 28;      { TMRRP command                    }
       clrrx      = 29;      { CLRRX command                    }
       imdwr      = 30;      { IMDWR command                    }
       sttmo      = 31;      { STTMO command                    }
       break      = 35;      { BREAK command                    }
       stsig      = 36;      { STSIG command                    }
       addcm      = 40;      { ADDCM command                    }
       addtx      = 41;      { ADDTX command                    }
       addrx      = 42;      { ADDRX command                    }
       noper      = 43;      { NOPER command                    }
       msize      = 44;      { MSIZE command                    }
       rsmde      = 45;      { RSMDE command                    }
       flash      = 46;      { FLASH command                    }

var    sauv_it    :  pointer;     { Address of old IT handler   }
       irq_done   :  boolean;     { Command acknowledged?       }
```

```
                { Write a character string to mailbox }


procedure mcx_write(adr:word; x:string);

 begin
 if x[0]= #1     then mem[mcx_adr:adr]:=byte(x[1])
            else move(x[1],mem[mcx_adr:adr],length(x));
end;


                      { Send a command to card }


procedure cli; inline($fa);
procedure sti; inline($fb);

procedure mcx_command(can:byte; comm:byte; param1:byte);

 begin
      repeat until irq_done;
      cli;
      irq_done:=false;
      mem[mcx_adr:opcode]:=comm;
      mem[mcx_adr:canal]:=can;
      mem[mcx_adr:mcc_p1]:=param1;
      mem[mcx_adr:start]:=$01;
      sti;
 end;
```

```
    { End-of-command and event interrupt sub-program }


procedure int_proc; interrupt;
var it_can, stat:  byte;

 begin
      stat:=mem[mcx_adr:status];
      if stat and $40<>0 then
           begin
                it_can:=mem[mcx_adr:it_chan];
                case mem[mcx_adr:it_cond] of
                     $01:  char_rdy(it_can);
                     $02:  modulo(it_can);
                     $03:  empty_notempty (it_can);
                     $04:  rx_full(it_can);
                     $08:  mcxstring(it_can);
                     $10:  timeout(it_can);
                     $20:  rx_error (it_can);
                     $40:  signals(it_can);
                     $80:  tx_empty(it_can);
                     $c0:  numbering (it_can);
                end;
           end else irq_done:=true;
      if stat and $80<>0 then error;
      mem[mcx_adr:endit]:=1; port[irq_base]:=$20;
 end;
```

The CHARACTER_RDY, MODULO, etc. functions represent the interrupts programmed by the MINTR command. They should be expanded into an actual program.
The IRQ_DONE variable lets you avoid sending a command before the previous command has been acknowledged.

```
          { Install the IT vector, initialize the 8259 }
         { and send a few commands to the 8 channel card }


procedure mcx_open;
var i:byte;

 begin
      get_int_vec(irq_vec,sauv_it);
      set_int_vec(irq_vec,@int_proc);
      port[irq_base+1]:=port[irq_base+1] and ($ff-irq_mask);
      mcx_write(mcc_data,'RUN 01');
      irq_done:=true;

      for i:=1 to nb_canal do
           begin
                mcx_command(i,vinit,#0);
                mcx_command(i,rxenb,#1);
           end;
 end;
              { Finished with card, restore old }
             { vector and address of associated process }


procedure mcx_close;

 begin
      set_int_vec(irq_vec,sauv_it);
      port[irq_base+1]:=port[irq_base+1] or irq_mask;
 end;
```

VII. NOTES

VIII. COMMAND INDEX

IX. EVALUATION SHEET

We appreciate your comments and suggestions for improving the quality and ease of use of our documentation.

We would be grateful if you could take a few moments to fill out this evaluation sheet and return it to us. Thanks in advance.

COMPANY:	Telephone:
User:	Position:
Address:	
Zip Code:	Country

Clearly indicate the version of the card, the software and the documentation:

MCX Card	
MCX-Lite/S Card	
EPROM version	
BASIC SOFTWARE Documentation version	

Please give a grade from 0 (worst) to 10 (best) for each of the following characteristics:

Format	☐	Presentation	☐	Organization	☐
Clear	☐	Precise	☐	Explanations	☐

Errors:

If you find any errors in this document, please note the page number and describe the type of error, or write the modifications on the pages in question and send photocopies of those pages along with this report.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

ACKSYS
3-5 rue du Stade
BP 80
78302 POISSY Cedex
FRANCE

- Telephone: +33 (1) 39 11 62 81
- Fax: +33 (1) 39 11 47 96