System Interfaces and Libraries

RS232/CL Interface

Programmer Guide

MOS

olivetti

PUBLICATION ISSUED BY:

Ing. C. Olivetti & C., S.p.A.
Direzione Documentazione
77, Via Jervis - 10015 Ivrea (Italy)

UPDATING STATUS

LEVEL	DATE	UPDATED PAGES	PAGES	CODE
3				4004450 H (3)
4	May 88	Front-cover, back-cover, title-page, v, vi, 1-1 ÷ 1-11, 2-1, 2-3, 2-16, 3-1, 4-1, 5-1, 6-1, 8-1.	30	4004454 F
2				
3				
4				
5				
6				
7				
8				
9				

Pages marked * should be removed

System Interfaces and Libraries

RS232/CL Interface

Programmer Guide

olivetti

PREFACE

This manual describes the asynchronous serial communication RS232/Current Loop interface for connecting serial peripherals to L1 and LSX MOS Systems.

The manual is directed towards programmers and system specialists who intend to handle peripheral units, using this interface in application programs written in the COBOL, Compiled BASIC, FORTRAN (only for L1 MOS systems), PASCAL+ and Interpreted BASIC (only for L1 MOS systems) languages.

SUMMARY

The manual is divided into eight chapters:

- The first introduces the RS232/CL interface, describing the possible connections.
- The second outlines the programmability of the software driver which handles the interface.
- The third, fourth, fifth and sixth chapters contain the description of the COBOL, Compiled BASIC, FORTRAN and PASCAL+ interfaces respectively giving the data formats for all parameters.
- The seventh chapter gives the description of the RS232/CL functions in alphabetical order giving the calls for every programming language.
- The eighth chapter describes the Interpreted BASIC interface and the relevant statements.

REFERENCES

Read first ...

For L1 MOS Systems

Introduction to MOS (in MOS Features and Functions - Code 4041600 F)

For LSX MOS Systems

Introduction to MOS (in MOS Features and Functions - Code 4053440 C)

For further information, read ...

Glossary/Glossario - Code 4002140 H

For L1 MOS Systems

COBOL Language Reference Manual
(in COBOL Programming Manual - Code 4040470 V)

COBOL Program Preparation and Examples
(in COBOL Programming Manual - Code 4040470 V)

Compiled BASIC Language Reference Manual
(in Compiled BASIC Programming Manual - Code 4041840 L)

Compiled BASIC Program Preparation
(in Compiled BASIC Programming Manual - Code 4041840 L)

FORTRAN Language Reference Manual
(in FORTRAN Programming Manual - Code 4042040 G)

FORTRAN Program Preparation and Execution
(in FORTRAN Programming Manual - Code 4042040 G)

PASCAL+ Language Reference Manual
(in PASCAL+ Programming Manual - Code 4041770 U)

PASCAL+ Program Preparation
(in PASCAL+ Programming Manual - Code 4041770 U)

BASIC Environment User Guide - Code 4041920 L

OLINK Linker User Guide - Code 4042120 X

ZLOC Linker User Guide - Code 4043900 E

For LSX MOS Systems

COBOL Language Reference Manual
(in COBOL Programming Manual - Code 4040470 V)

COBOL Program Preparation and Examples - Code 4053500 N

Compiled BASIC Language Reference Manual
(in Compiled BASIC Programming Manual - Code 4041840 L)

Compiled BASIC Program Preparation - Code 4053490 R

PASCAL+ Language Reference Manual
(in PASCAL+ Programming Manual - Code 4041770 U)

PASCAL+ Program Preparation - Code 4053480 Q

MLOC Linker User Guide - Code 4040700 G

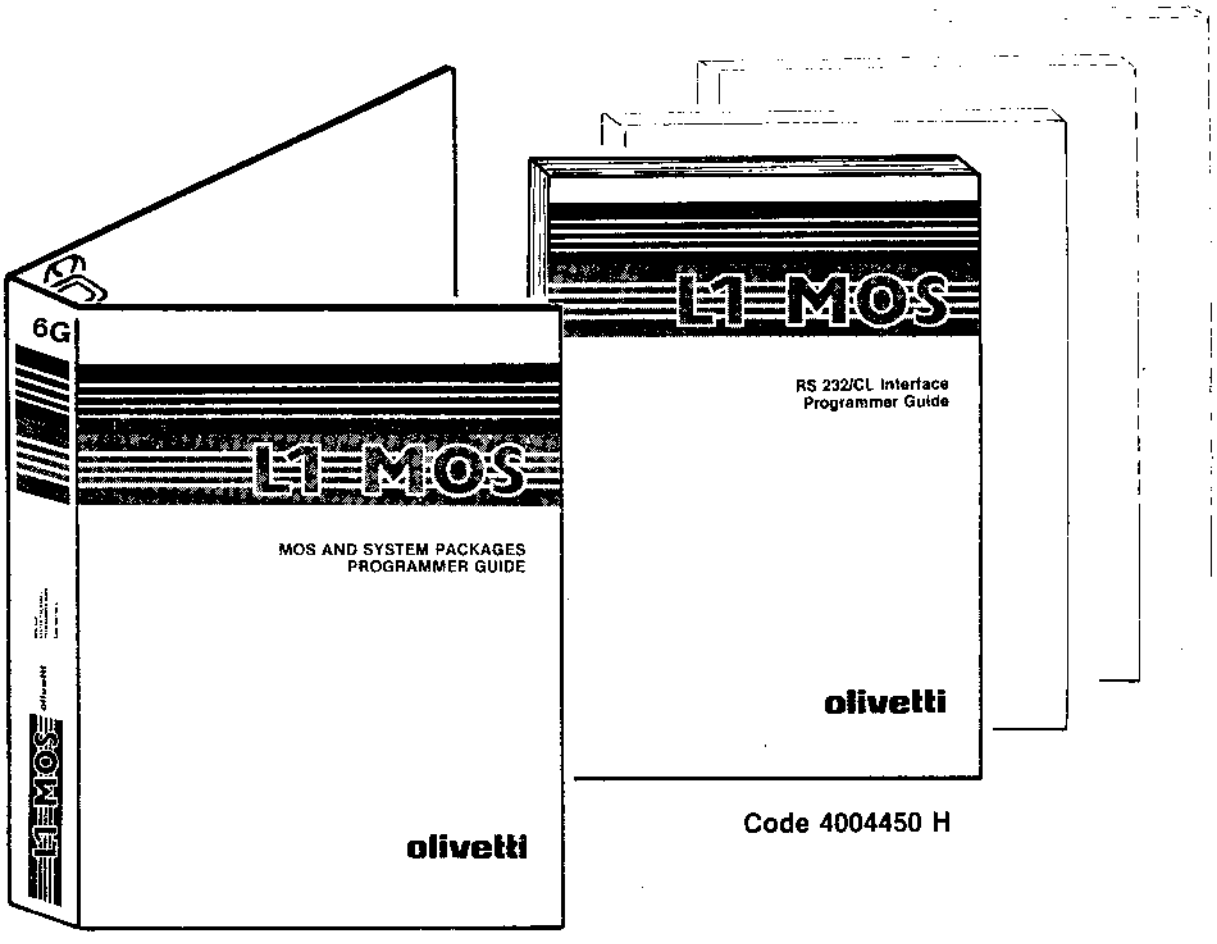
FIRST EDITION: September 1983 - Release 2.0

SECOND EDITION: February 1985 - Release 4.2

UPDATE: June 1985 - Release 5.0

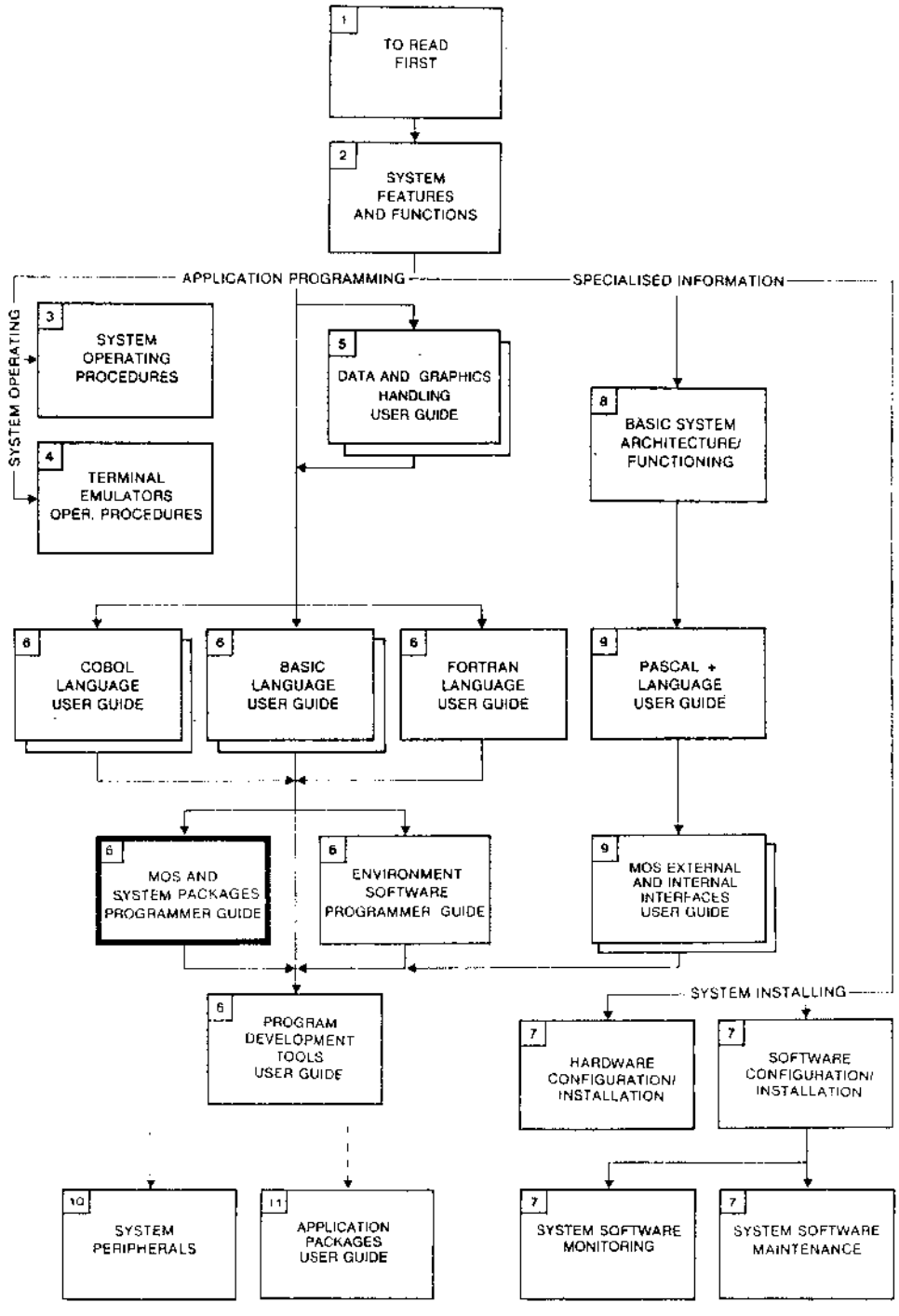
THIRD EDITION: January 1987 - Release 5.2

UPDATE: May 1988 - Releases 6.0 (L1 MOS) and 1.0 (LSX MOS)



Code 4001190 A

Code 4004450 H


1
TO READ FIRST
2
SYSTEM FEATURES AND FUNCTIONS
APPLICATION PROGRAMMING
SPECIALISED INFORMATION
SYSTEM OPERATING
3
SYSTEM OPERATING PROCEDURES
4
TERMINAL EMULATORS OPER. PROCEDURES
5
DATA AND GRAPHICS HANDLING USER GUIDE
8
BASIC SYSTEM ARCHITECTURE/ FUNCTIONING
6
COBOL LANGUAGE USER GUIDE
6
BASIC LANGUAGE USER GUIDE
6
FORTRAN LANGUAGE USER GUIDE
9
PASCAL + LANGUAGE USER GUIDE
6
MOS AND SYSTEM PACKAGES PROGRAMMER GUIDE
6
ENVIRONMENT SOFTWARE PROGRAMMER GUIDE
9
MOS EXTERNAL AND INTERNAL INTERFACES USER GUIDE
SYSTEM INSTALLING
6
PROGRAM DEVELOPMENT TOOLS USER GUIDE
7
HARDWARE CONFIGURATION/ INSTALLATION
7
SOFTWARE CONFIGUHATION/ INSTALLATION
10
SYSTEM PERIPHERALS
11
APPLICATION PACKAGES USER GUIDE
7
SYSTEM SOFTWARE MONITORING
7
SYSTEM SOFTWARE MAINTENANCE

CONTENTS

1. THE RS232 AND CURRENT LOOP INTERFACE

INTRODUCTION

This manual describes the RS232 and CURRENT LOOP (CL) interface for connecting serial peripherals.

An interface is a means of communication between the MOS systems and external input or output equipment of a different type, inasmuch as the standards define the necessary conventions for the functioning and the mechanical and electrical specifications of the connection.

Olivetti has provided the MOS systems with all the software and hardware necessary for supporting the peripheral equipment in accordance with these standards.

The RS232/CL interface pilots the transmission of digital data in asynchronous mode, character by character.

Transmission can be "half duplex", that is, in both directions (to and from the MOS systems), but this means that the information is only transmitted in one direction at a time, or "full duplex" when information can be transmitted in both directions at the same time.

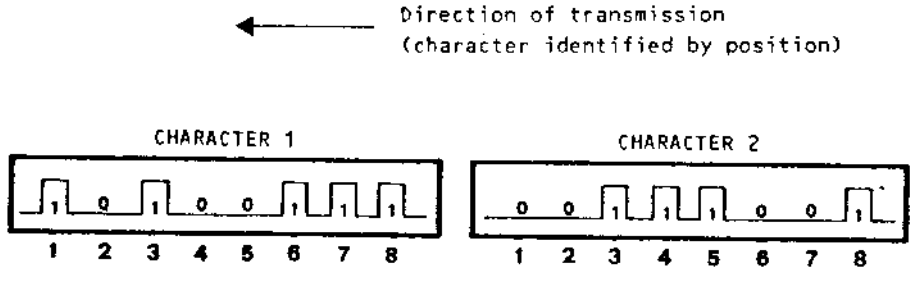


Fig. 1-1 Serial Transmission

SERIAL TRANSMISSION

The serial interface has been projected according to the Recommended Standard RS232 of the Electronic Industries Association in Washington, DC, USA and therefore is known as the EIA RS232, or RS232 for short (the international equivalent is Recommendation V.24 of the International Telecommunications Union - Comite' Consultif International pour la Telephonie et la Telegraphie - which is better known as CCITT V.24). (See Appendix C.)

DEFINITIONS

No distinction is made in this manual between the various types of equipment which can be connected together via the interface.

Equipment is a generic term to indicate a measuring instrument (for example, a thermometer), a peripheral (that is, a device to be externally connected to a computer, for example plotter), or a machine (that is, a device providing specific functions via a logical interface, for example a numerically controlled lathe).

As this manual refers to the interface of a particular system (the MOS system) but does not discuss any specific piece of equipment the words: equipment, peripheral, instrument, and machine will be used indifferently, except where indicated.

A peripheral which conforms to the EIA RS232 and CURRENT LOOP 20 mA serial interfaces is generally used.

THE EIA RS232 AND CURRENT LOOP 20 mA INTERFACE

In the MOS system the controllers CPU, TWIN and MUX handle 1, 2 or 4 lines respectively. The lines handled via TWIN and via MUX (partly) can be either RS232 or Current loop lines, but the line handled via CPU can be only a RS232. It is also possible, by connecting the ELB controller to one of the 4 MUX channels, to obtain 2 RS232 channels of work station.

The MOS system implements:

- The DCE side of the RS232, interface by means of the RS232 PERIPH cable (Standard 13 Olivetti) with which it can be connected to equipment which implements the DTE side of the interface.

  This type of connection is used for transferring data to a distance of no more than 7 meters at a speed of up to 19200 baud.

- The DTE side of the RS232 interface, by means of the RS232 MODEM cable with which it can be connected to an asynchronous modem or to a serial peripheral that implements the DCE side of the interface.

  This type of connection is used for transferring data at a speed of up to 19200 baud.

  The system must not be more than 15 meters apart from the modem.

The MOS system creates a CURRENT LOOP 20 mA interface by means of the RS232 TWIN or MUX board and the CURRENT LOOP cable for exchanging information with compatible peripherals.

This type of connection is used for transferring data to a distance of no more than 300 meters at a speed of up to 19200 baud.

The CURRENT LOOP connection galvanically isolates the systems and, therefore, allows operations or different (and not equipotential) grounds.

The following example shows a possible configuration with three serial peripherals, one of which is a CURRENT LOOP 20 mA.

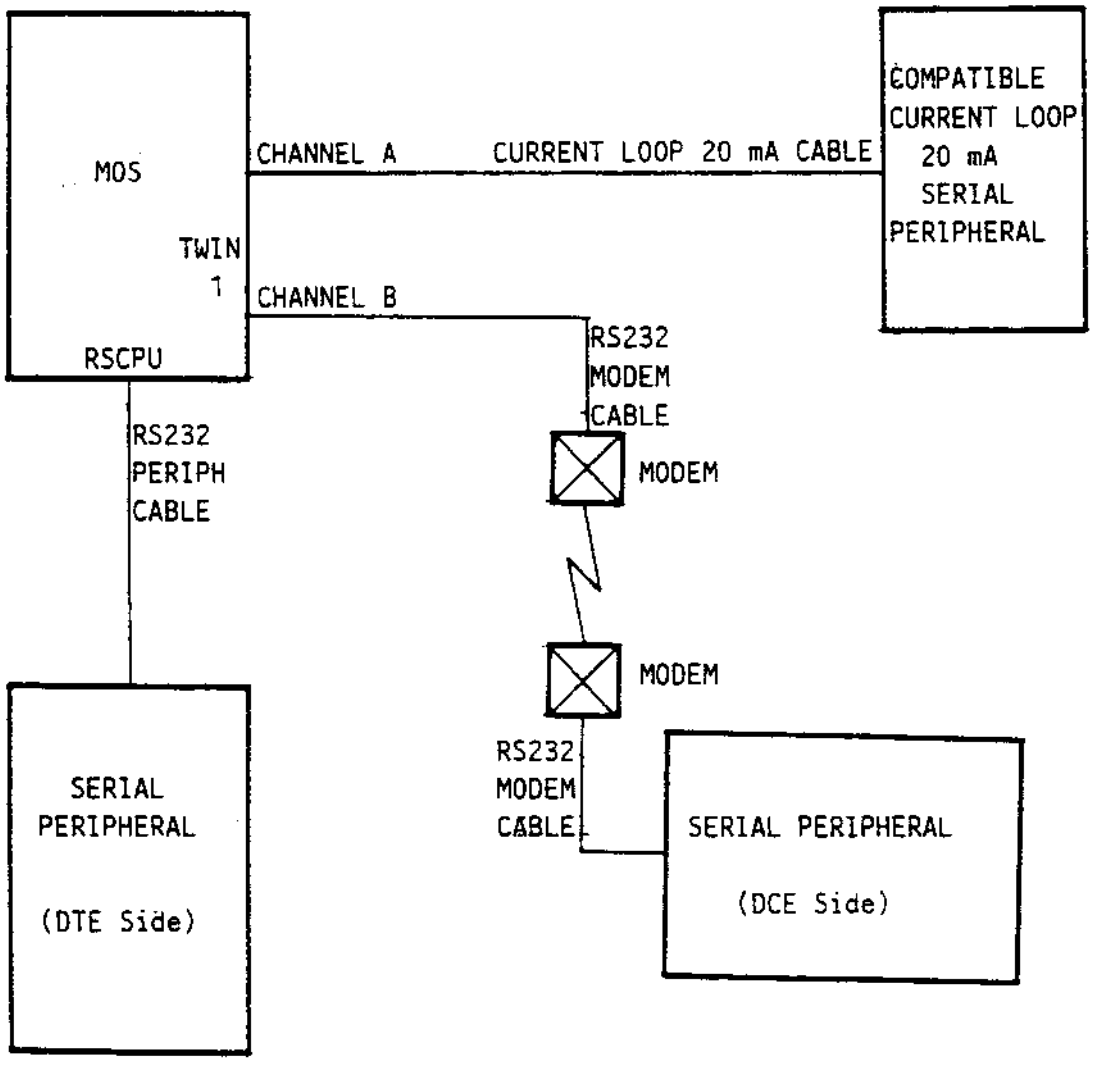


Fig. 1-2 Example of Configuring a MOS System

TYPES OF CONNECTION

MOS SYSTEM TO A PERIPHERAL

The signals which the MOS system exchanges with a serial peripheral conform to the EIA RS232 Standard.

The interface signals and their functions are represented in Fig. 1-3 and are described below. The signal names also conform to the EIA RS232 Standard.

As the electronic interpretation of each signal may vary according to the type of device connected to the MOS system, the relative manual should be consulted for greater detail about the signals. The signals are classified in three categories: ground, data and control. The expressions "Level ON" and "Level OFF" are used here and have the following voltages:

Level ON > 3 volt
Level OFF < -3 volt

Ground Signals:

Protective Ground - (Circuits: CCITT 101, EIA AA): this is a conductor which is electrically linked to the frame of the machine or device. It can also be connected, if necessary, to an external ground.

Signal Ground / Common Return - (Circuits: CCITT 102, EIA AB): this is a conductor which stabilises the common reference potential of all the signals described in this chapter. It is connected to the peripheral unit in the same place as it is connected tonked to the protection ground via a metal jumper. This jumper can be detached to reduce electronic disturbance.

Data Signals:

Transmitted Data - (Circuits: CCITT 103, EIA BA): this signal indicates the data to be transmitted on-line from the peripheral to the MOS system.

Received Data - (Circuits: CCITT 104, EIA BB); this signal indicates the data transmitted to the peripheral from the MOS system.

Control Signals:

Request To Send - (Circuits: CCITT 105, EIA CA): the signals on this circuit, sent from the peripheral, regulate the transmission function of the data channel. Level ON indicates that the peripherals have requested transmission, whilst Level OFF indicates that there are no transmission requests from the peripherals. This signal is not always significant, its use depends on the characteristics of the peripheral.

Clear To Send - (Circuits: CCITT 106, EIA CB): the signals on this circuit are sent from the MOS systems and indicate whether the MOS system can receive data on the data channel. Level ON means that it can, and Level OFF indicates that the MOS system is not ready to receive data.

Data Set Ready - (Circuits: CCITT 107, EIA CC): the signals on this circuit are sent from the MOS system to indicate whether it is ready to operate. Level ON indicates that it is ready, and Level OFF that it is not.

Data Terminal Ready - (Circuits: CCITT 108/2, EIA CD): the signals on this circuit are transmitted by the peripheral and indicate whether it is ready to exchange information with the MOS system. Level ON indicates that the unit is ready, and Level OFF that it is not. This signal is not always significant. Its use depends on the type of peripheral involved.

Received Line Signal Detector - (Circuits: CCITT 109, EIA (CF): the signals on this circuit are transmitted by the MOS system, and indicate whether it is ready to transmit data to the peripheral. Level ON indicates that it is ready, and Level OFF that it is not.

Busy - (Non standard circuit): the meaning of this signal, transmitted by peripheral, depends on the type of peripheral connected to the MOS system. It is used to transmit the "busy" or "anomaly" state to the MOS system.

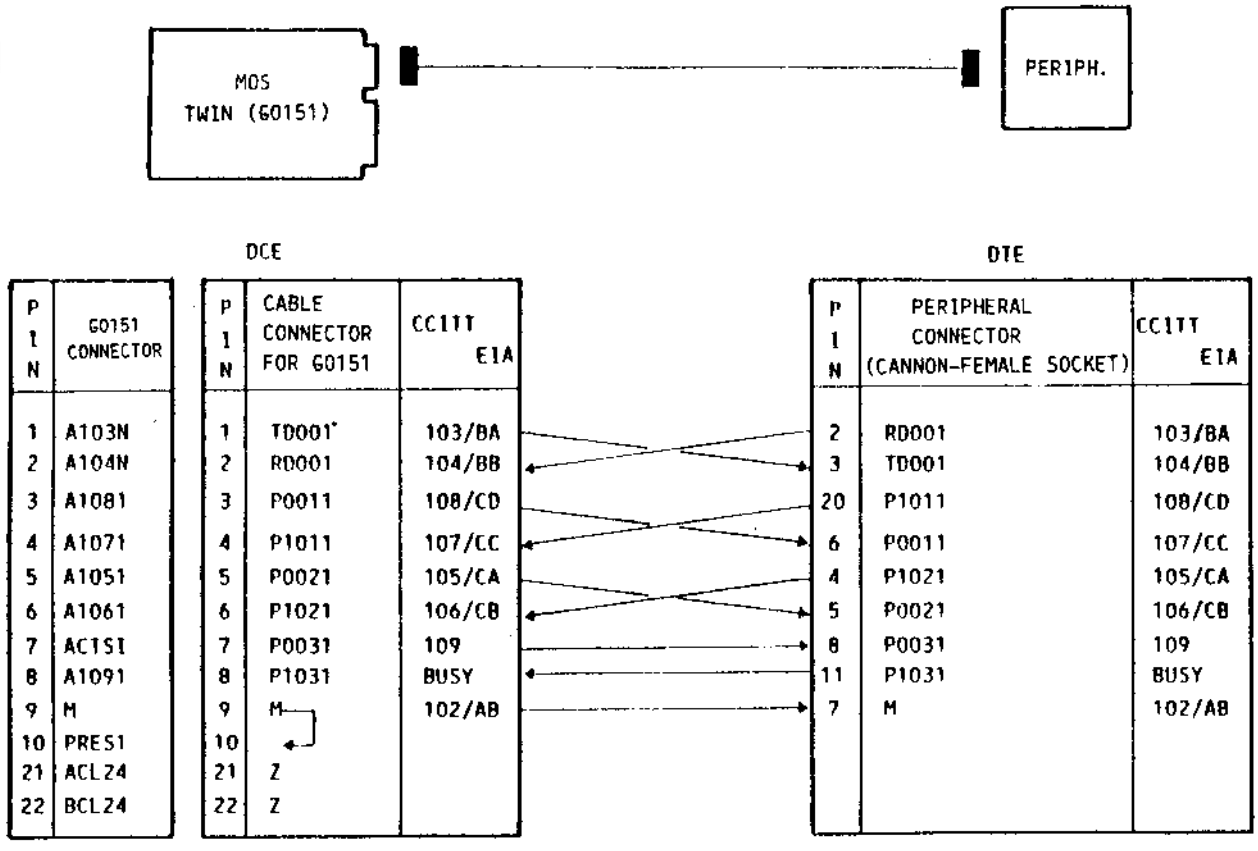


Fig. 1-3 Signals Transmitted on the RS232 PERIPH Cable

MOS SYSTEM TO MODEM

The signals that the MOS system transmits and receives on the same cable via the modem, conform to the EIA RS232 standard.

The interface signals and their functions are shown in Fig. 1-4 and are now described. The names of the signals conform to the EIA RS232 standard. According to this, assignment of the pins, as shown in Fig. 1-4, is valid for all connectors, of both the RS232 mounted on the CPU board and the TWIN interfaces.

The signals are classified in three categories: ground, data and control. In the part of this chapter that follows the expressions "Level ON" and "Level OFF" are used, to which the following voltages correspond:

Level ON > 3 volt
Level OFF < -3 volt

Ground Signals:

Protective Ground - (Circuits: CCITT 101, EIA AA): this is a conductor connected electrically to the chassis of the machine or of the device. It can also be connected, if necessary, to an external ground.

Ground Signal / Common Return - (Circuits: CCITT 102, EIA AB): this is a conductor that establishes the common reference potential between the signals described in this chapter; in the DCE it is fixed to a pin which is connected to a protection ground by means of a metal jumper. This jumper can be disconnected to reduce electronic disturbance.

Data Signals:

Transmitted Data - (Circuits: CCITT 103, EIA BA): this is a binary signal indicating the data to be transmitted by the MOS system to the DCE.

Received Data - (Circuits: CCITT 104, EIA BB); this is a binary signal indicating the data transmitted by the DCE to the MOS system.

Control Signals:

Request To Send - (Circuits: CCITT 105, EIA CA): the signals on this circuit are transmitted by the MOS system and control the transmission function of the data channel. The ON level allows the DCE to transmit. The OFF level stops transmission by the DCE when all the data in circuit 103 (Transmitted Data) have been transmitted.

Clear To Send - (Circuits: CCITT 106, EIA CB): the signals on this circuit are transmitted by the DCE, and indicates if the DCE can receive data on the appropriate channel. The ON level specifies that the DCE is ready to receive data. The OFF level shows that the DEC is ready to receive data.

Data Set Ready - (Circuits: CCITT 107, EIA CC): The signals on this circuit are transmitted by the DCE to indicate if it is ready to operate. The ON level indicates that the signal converter (or some other device) is connected on-line and that the DCE is ready to exchange further control signals with the MOS system, that allows data communication to commence. The OFF level shows that the DCE is not ready; it must be remembered that this signal is not always significant and that sometimes the wrong signal is received by the control unit.

Data Terminal Ready - (Circuits: CCITT 108/2, EIA CD): the signals in this circuit, which are transmitted by the MOS system, control switching the signal converter or similar devices, on and off the line. The ON level indicates that the MOS system is ready to operate, and sets up the DCE to connect the signal converter (or some similar device) to the line; it also allows this connection (which is made successively) to be maintained until further notice. The MOS system is enabled to transmit level ON over the circuit whenever it is ready to transmit or receive data. Level OFF allows the DCE to pass to ready

Received Line Signal Detector - (Circuits: CCITT 109, EIA CF): The signals on this circuit, which are transmitted by the DCE to indicate if the signal received from the channel line is within the limits that are accepted for data transmission. The ON level indicates that the signal that has been received is within these limits, and the OFF level that it is not.

Transmit Clock - (Non standard circuit): this is a signal that controls the transmission speed of a character.

Receive Clock - (Non standard circuit): this is a signal that controls the reception speed of a character.

Ring Indicator - (Circuits : CCITT 125, EIA CE): the signals on this circuit, which are send by the DCE, indicate that a call indicator has

been received by the DCE. The ON level indicates that it has been received, the OFF level indicates that no signal has been received.

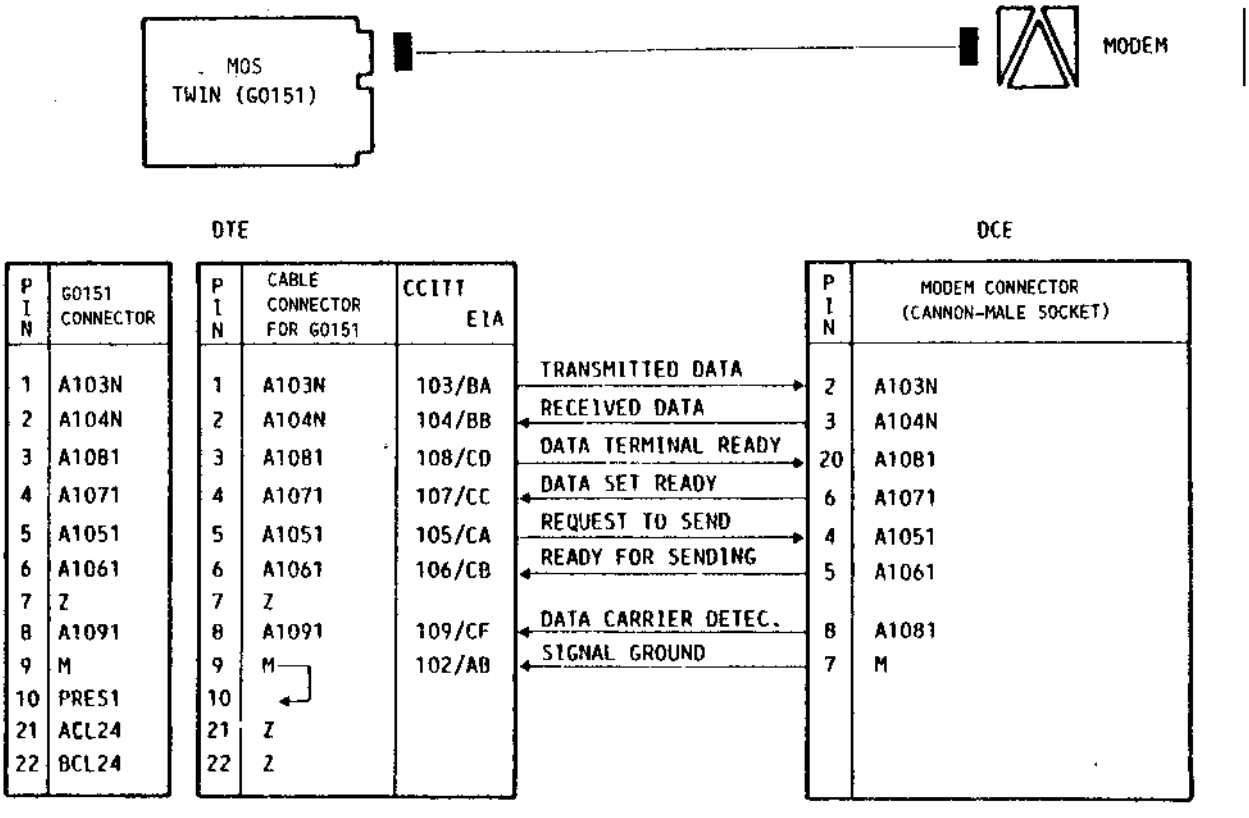


Fig. 1-4 Signals Transmitted on the RS232/CL MODEM Cable

CURRENT LOOP

As indicated in Fig. 1-5, a CURRENT LOOP cable contains four conductors; two are assigned to the transmitter and two to the receiver. The cables must be connected as indicated in the figure. The meaning of each signal depends on the type of peripheral connected.

Note

All the makers use conductors with colours and names specifically defined for compatible Current LOOP 20 mA devices.

The electric feed can be provided by the MOS system ("active" use) or by the connected device ("passive" use).

For greater detail see the manual on the device constructor.

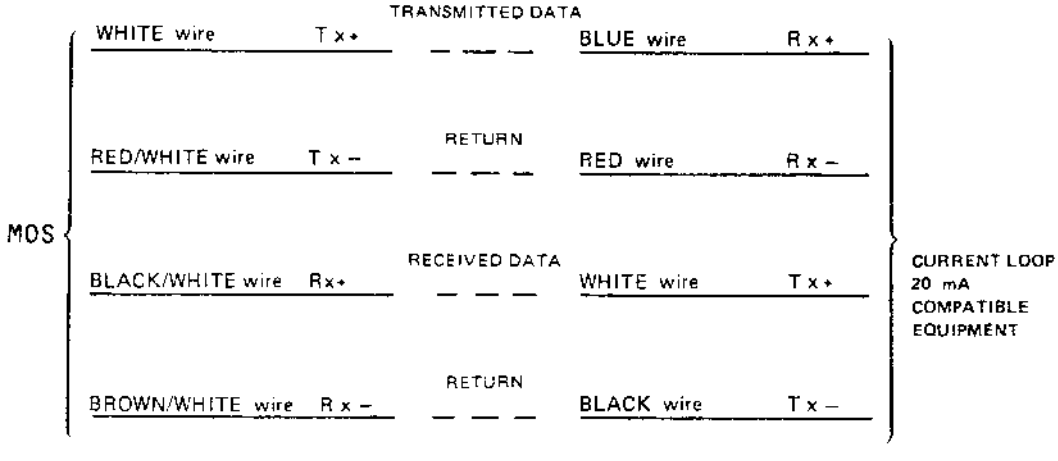


Fig. 1-5 Signals Transmitted on the Conductors of a CURRENT LOOP Cable

The CURRENT LOOP interface can be connected to one or two compatible CURRENT LOOP peripherals. Only point-to-point connections are possible.

The MOS System as Receiver

The standard connection between MOS system as receiver and a compatible piece of equipment as transmitter is illustrated in Fig. 1-6.

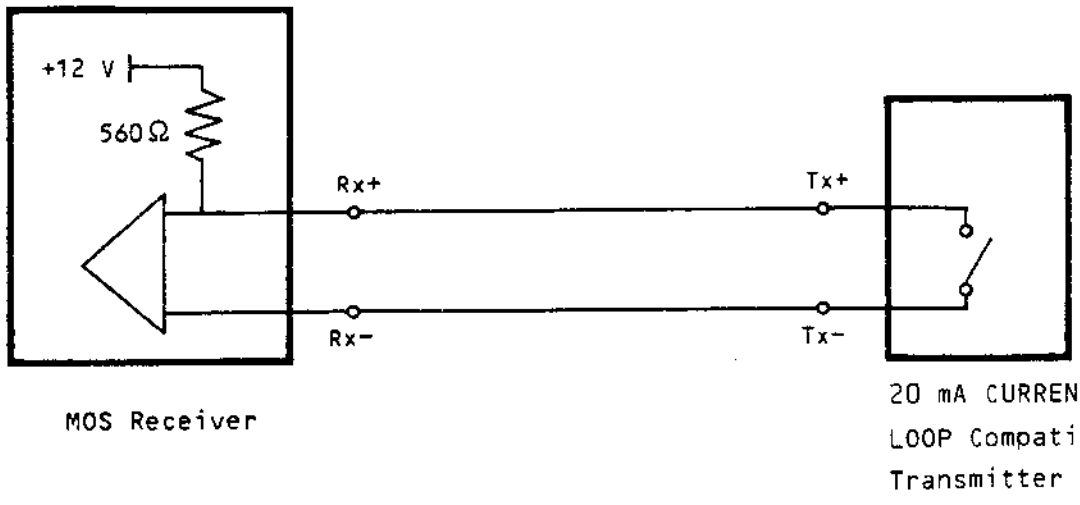


Fig. 1-6 Serial Connection with a MOS System as Receiver in Current Loop

The MOS System as Transmitter

The normal connection between the MOS system as transmitter and a compatible piece of equipment as receiver is illustrated in Fig. 1-7.

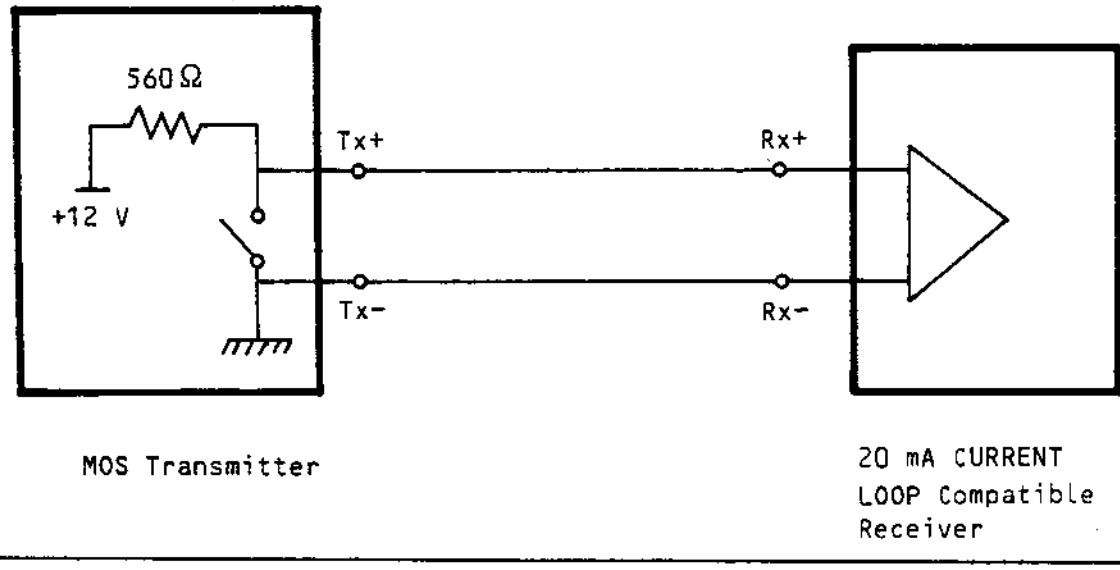


Fig. 1-7 Serial Connection with a MOS system as Transmitter in Current Loop

2. INTERFACE SOFTWARE

With the interface software the user can utilise the RS232/CL interface working in one of the languages that handle it: COBOL, Compiled BASIC, FORTRAN (only for L1 MOS systems), PASCAL+ and Interpreted BASIC (only for L1 MOS systems). All these languages provide the programmer with a set of functions or instructions with which to carry out processing, as well as defining the programming characteristics of each of the lines being used.

Whilst the compiled languages access the interface software functions via simple calls, the Interpreted BASIC application environment access them by means of specific BASIC statements. These statements are described in Chapter 8. The functions which can be called by compiled languages are described in Chapter 7. Chapters 3, 4, 5 and 6 provide the interface and the format of the parameters for each compiled language.

FUNCTIONS AND ERRORS HANDLING

This section contains both a brief introduction to the operations that can be carried out using the RS232/CL serial interface, plus two tables. The first table shows the correspondence between the global name identifying the function and the actual name of the each function in the various compiled languages (described in detail in Chapter 7). The second contains the reply codes for the compiled languages. (For reply codes in Interpreted BASIC see APP.A). A line errors handling example is also given.

GLOBAL NAME DESCRIPTION	COBOL NAME	COMPILED BASIC NAME	FORTRAN NAME	PASCAL+ NAME
ATTACHLINE Attaches a line to a pool of lines	RS_RSATTLN	RS_RSATTLN	ATTACH	ATTACHLN
CLOSELINE Closes data exchange on a line.	RS_RSCLOSELN	RS_RSCLOSELN	CLOSLN	CLOSELN
DETACHLINE Detaches a line from a pool of lines	RS_RSDETLN	RS_RSDETLN	DETACH	DETACHLN
OPENLINE Exchanges data on a line	RS_RSOPENLN	RS_RSOPENLN	OPENLN	OPENLN
READALLSTLINE Reads the reception buffer in 15 lines.	RS_RSRASTLN	RS_RSRASTLN	RDSTLN	READALLSTLN
READINFOLINE Reads the status of all the lines.	RS_RSRINFOLN	RS_RSRINFOLN	RDINFO	READINFOLN
READLINE Reads data from a line.	RS_RSREADLN	RS_RSREADLN	READLN	READLN

Tab. 2-1 Functions for Handling the RS232/CL Interface (cont.)

GLOBAL NAME DESCRIPTION	COBOL NAME	COMPILED BASIC NAME	FORTRAN NAME	PASCAL+ NAME
READSYSLINE Reads the reception buffer of all the lines.	RS_READSYSLN	RS_READSYSLN	RDSYLN	READSYSLN
WAITONPOOL The function waits for data on any of the lines attached to the pool.	RS_RSWONP	RS_RSWONP	WAITPL	WAITONPOOL
WRITEINFOLINE Defines or updates the programming status of the line and driver modes.	RS_RSWINFOLN	RS_RSWINFOLN	WRINFO	WRITEINFOLN
WRITELINE Transmits data on a line.	RS_RSWRITELN	RS_RSWRITELN	WRTLN	WRITELN

Fig. 2-1 Functions for Handling the RS232/CL Interface

Note: From release 5.0 of L1 MOS on, READSYSLINE replaces READALLSTLINE, which is maintained for compatibility. Infact READALLSTLINE inquires the status of the reception buffer of 15 lines, while READSYSLINE inquires the status of the 113 possible lines.

This table contains:

- the reply codes that can be returned by the COBOL, Compiled BASIC, FORTRAN and PASCAL+ functions
- the meaning of each code.

REPLY CODES	MEANINGS
0	Operation terminated correctly.
1	Invalid identifier.
2	One or more invalid parameters have been supplied.
4	Line busy.
8	Illegal operation requested.
16	Line error caused by: - power failure - reception buffer limits exceeded - overrun - parity error - framing error - break in input (only significant if the protocol does not handle the break).
32	Reading interrupted.
64	Time out expired, or all the data on the line have been read.

Tab. 2-2 Reply Codes

Line Errors Handling Procedure

When using the RS232/Current Loop driver, the need may arise to handle line errors, usually due to cables being either not correctly connected or disconnected, or to peripheral devices being switched off or not enabled to transmit.

Here follows the description of a procedure to handle this type of error.

The steps to be performed are:

1. Open the line via an Openline

2. Perform a Readline with datasize = 0. This allows getting back control of the line, so as to be able to check whether any anomaly has occurred:

   - If no errors are signalled, then program execution may be carried on.

   - If line errors (code 16) are signalled, then the following must be done:

3. Execute a Readinfoline with Whatinfo = 4. This allows reading of the reception status of the line:

   - If the error is not present any more, it allows resetting of the line and continuation of the execution of the application program.

   - If the line error is still present, then:

4. Send a message to the operator which describes the possible cause of the error. This can be:

   - Disconnected cable or peripheral device in BREAK status (transmission disabled)

   - Peripheral device switched off or cable not correctly connected.

   Now, the operator must check the device and cable status and remove the cause of the error condition. An answer from the operator is requested to decide whether to abort the operation or to carry it on.

5. In the latter case, a Readinfoline is performed again to make sure the line error has been actually removed, allowing continuation of the execution of the application program.

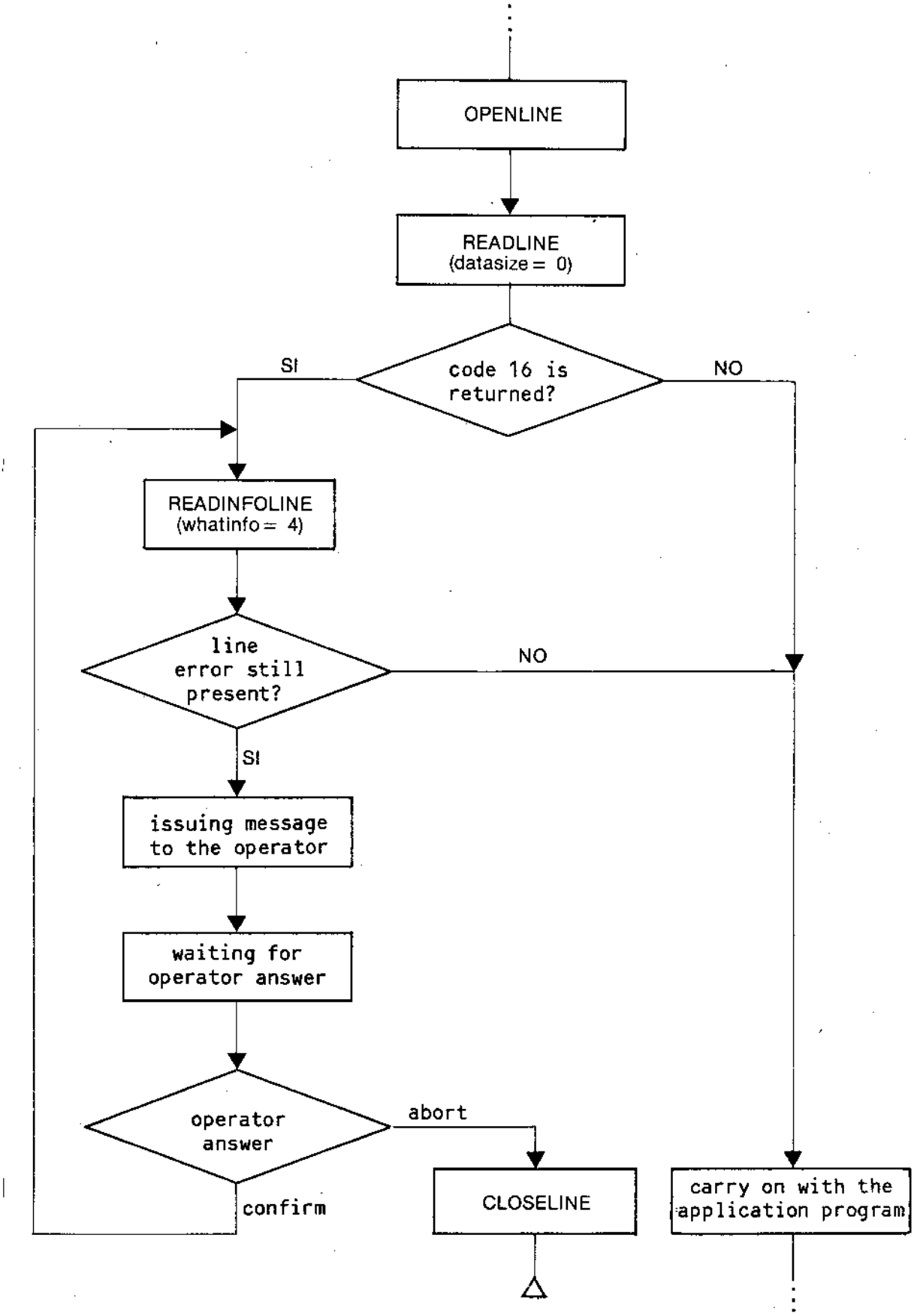


Fig. 2-3 Example of Line Errors Handling

PROGRAMMING CHARACTERISTICS OF THE LINE

When the line is opened, using the "OPENLINE" function or the Interpreted BASIC "OPEN" instruction, the user can define the programming characteristics of the line, by assigning the lineConf parameter the appropriate values (the parameter must have 17 characters).

The accepted values for the parameter have the following meaning:
(Note: the values must be expressed in character form)

POSITION OF THE CHARACTER	ADMITTED VALUES	MEANING	DEFAULT VALUE
1	0	The transmission wire is placed in the 'mark' state (transmission enabled).	
	1	The transmission wire is placed in the 'break' state (transmission disabled).	
	?	The programming characteristics of the line are the same as those specified in the configuration record and the driver default values.	
2	0	Value of the RTS (Request To Send) = 0.	
	1	Value of the RTS = 1.	
3	0	Value of the DTR (Data Terminal Ready) = 0.	
	1	Value of the DTR = 1.	
4	0	Value of the signal in output from the Peripheral Dependent = 0.	
	1	Value of the signal in output from the Peripheral Dependent = 1.	
5	0	Line used in half duplex. (Not for RSCPU)	
	1	Line used in full duplex.	1
6	0	Number of the stop bits = 1.	
	1	Number of the stop bits = 1,5. (Not for RSCPU)	
	2	Number of the stop bits = 2.	

(cont.)

POSITION OF THE CHARACTER	ADMITTED VALUES	MEANING	DEFAULT VALUE
7	0	No parity.	
	1	Odd parity.	
	2	Even parity.	
8		Length of the character being received:	
	0	5 bits (Not for RSCPU)	
	1	6 bits (Not for RSCPU)	
	2	7 bits	
	3	8 bits	
9		Length of the character being transmitted:	
	0	5 bits (Not for RSCPU)	
	1	6 bits (Not for RSCPU)	
	2	7 bits	
	3	8 bits	
10		Reception speed: (baud)	
	0	50	
	1	75	
	2	110	
	3	134,5	
	4	150	
	5	200	
	6	300	
	7	600	
	8	1200	
	9	1800	
	:	2400	
	;	4800	
	<	9600	
	=	19200	
	>	38400	

(cont.)

POSITION OF THE CHARACTER	ADMITTED VALUES	MEANING	DEFAULT VALUE
11		Transmission speed: (baud)	
	0	50	
	1	75	
	2	110	
	3	134,5	
	4	150	
	5	200	
	6	300	
	7	600	
	8	1200	
	9	1800	
	:	2400	
	;	4800	
	<	9600	
	=	19200	
	>	38400	

Note: If the line controller only supports one speed for reception and transmission, that indicated by the character in position 10 is adopted.

12		Protocol type:	
	0	No protocol	0
	1	Protocol which handles the xon and xoff characters	
	2	Protocol which handles the break (not valid for the CPURS board)	
	3	Protocol which handles the busy signal (not valid for the CPURS board)	

13	0	Protocol used both in reception and transmission.
	1	Protocol used only in reception.
	2	Protocol used only in transmission.

Note: Consult lineConf[12] for the RS232 on the CPU board

14	0	The line is not closed for diagnostics.
	1	The line is closed again: the output is sent back as input for checking.

(cont.)

POSITION OF THE CHARACTER	ADMITTED VALUES	MEANING	DEFAULT VALUE
15	0	The DSR status change from 1 to 0 is interpreted by the driver as a line down.	
	1	The DSR status changes are ignored.	1
16	0	The CTS status change from 1 to 0 is interpreted by the driver as a line down. (Not for RSCPU)	
	1	The CTS status changes are ignored.	1
17	0	The DCD status change from 1 to 0 is interpreted by the driver as a line down. (Not for RSCPU)	
	1	The DCD status changes are ignored.	1

The values relative to the other parameters referring to the driver states can be updated (transmission methods with relative time-outs; reception methods with relative time-outs, in match or transparent mode, with or without enabled echo) using the "WRITEINFOLINE" function, or the Interpreted BASIC "CMD" instruction.

For further details, see the function descriptions given later in this manual.

IDENTIFICATION OF THE LINE

The TWIN, CPU, and MUX controllers allow to handle 2, 1 and 4 lines respectively.
As to the connection via MUX it is also possible, by connecting the ELB 3683 controller to one of the 4 MUX channels, to obtain 2 RS232 channels each of which allow to handle any peripheral.

The "lnid" parameter, if supplied with the proper value, allow to identify a line. The possible values and the correspondence between lines and channels are given in the table below.

Note

Regarding the connection via MUX, the channels referred to as T1, T2, T3, T4 are transparent lines (direct connection), but the channels referred to as 1A, 1B, 2A, 2B, 3A, 3B, 4A, 4B are lines connected to MUX via the ELB 3683 controller. Connections via MUX are shown in the following figures.

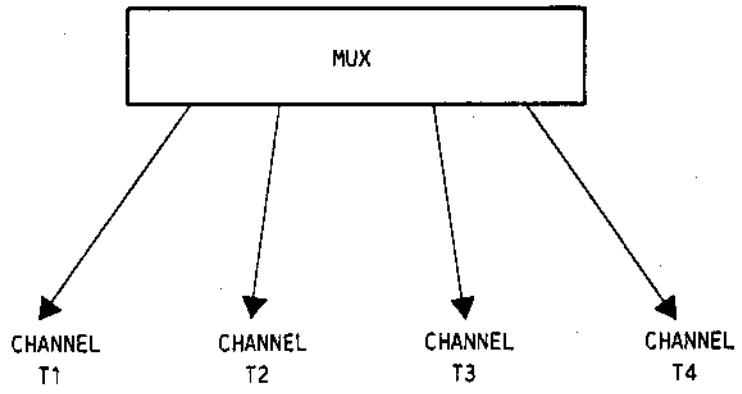


Fig. 2-4 Direct Connection

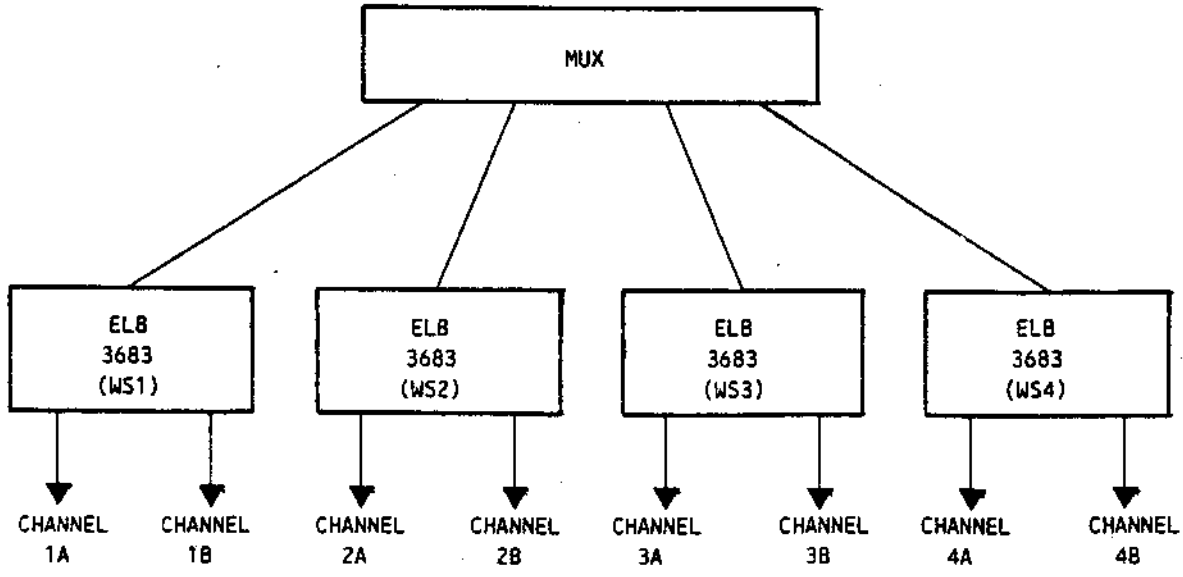


Fig. 2-5 Connection via ELB 3683

LNID VALUE	MEANING	CHANNEL
1	line 1 : connection via CPU	CPU
2	line 2 : connection via Twin	Twin 1 channel A
3	line 3 : " " "	Twin 1 channel B
4	line 4 : " " "	Twin 2 channel A
5	line 5 : " " "	Twin 2 channel B
6	line 6 : " " "	Twin 3 channel A
7	line 7 : " " "	Twin 3 channel B
8	line 8 : " " "	Twin 4 channel A
9	line 9 : " " "	Twin 4 channel B
10	line 10 : " " "	Twin 5 channel A
11	line 11 : " " "	Twin 5 channel B
12	line 12 : " " "	Twin 6 channel A
13	line 13 : " " "	Twin 6 channel B
14	line 14 : " " "	Twin 7 channel A
15	line 15 : " " "	Twin 7 channel B
16	line 16 : " " "	Twin 8 channel A
17	line 17 : " " "	Twin 8 channel B
18	line 18 : connection via MUX	MUX1 channel T1
19	line 19 : " " "	MUX1 channel T2
20	line 20 : " " "	MUX1 channel T3
21	line 21 : " " "	MUX1 channel T4
22	line 22 : " " "	MUX1 channel 1A
23	line 23 : " " "	MUX1 channel 1B
24	line 24 : " " "	MUX1 channel 2A
25	line 25 : " " "	MUX1 channel 2B
26	line 26 : " " "	MUX1 channel 3A
27	line 27 : " " "	MUX1 channel 3B
28	line 28 : " " "	MUX1 channel 4A
29	line 29 : " " "	MUX1 channel 4B
30	line 30 : " " "	MUX2 channel T1
31	line 31 : " " "	MUX2 channel T2
32	line 32 : " " "	MUX2 channel T3
33	line 33 : " " "	MUX2 channel T4
34	line 34 : " " "	MUX2 channel 1A
35	line 35 : " " "	MUX2 channel 1B
36	line 36 : " " "	MUX2 channel 2A
37	line 37 : " " "	MUX2 channel 2B
38	line 38 : " " "	MUX2 channel 3A
39	line 39 : " " "	MUX2 channel 3B
40	line 40 : " " "	MUX2 channel 4A
41	line 41 : " " "	MUX2 channel 4B
42	line 42 : " " "	MUX3 channel T1
43	line 43 : " " "	MUX3 channel T2
44	line 44 : " " "	MUX3 channel T3
45	line 45 : " " "	MUX3 channel T4
46	line 46 : " " "	MUX3 channel 1A
47	line 47 : " " "	MUX3 channel 1B

Tab. 2-6 Correspondence Between Lines and Channels (cont.)

LNID VALUE	MEANING	CHANNEL
48	line 48 : " " "	MUX3 channel 2A
49	line 49 : " " "	MUX3 channel 2B
50	line 50 : " " "	MUX3 channel 3A
51	line 51 : " " "	MUX3 channel 3B
52	line 52 : " " "	MUX4 channel 4A
53	line 53 : " " "	MUX3 channel 4B
54	line 54 : " " "	MUX3 channel T1
55	line 55 : " " "	MUX4 channel T2
56	line 56 : " " "	MUX4 channel T3
57	line 57 : " " "	MUX4 channel T4
58	line 58 : " " "	MUX4 channel 1A
59	line 59 : " " "	MUX4 channel 1B
60	line 60 : " " "	MUX4 channel 2A
61	line 61 : " " "	MUX4 channel 2B
62	line 62 : " " "	MUX4 channel 3A
63	line 63 : " " "	MUX4 channel 3B
64	line 64 : " " "	MUX4 channel 4A
65	line 65 : " " "	MUX4 channel 4B
66	line 66 : connection via MUX	MUX5 channel T1
67	line 67 : " " "	MUX5 channel T2
68	line 68 : " " "	MUX5 channel T3
69	line 69 : " " "	MUX5 channel T4
70	line 70 : " " "	MUX5 channel 1A
71	line 71 : " " "	MUX5 channel 1B
72	line 72 : " " "	MUX5 channel 2A
73	line 73 : " " "	MUX5 channel 2B
74	line 74 : " " "	MUX5 channel 3A
75	line 75 : " " "	MUX5 channel 3B
76	line 76 : " " "	MUX5 channel 4A
77	line 77 : " " "	MUX5 channel 4B
78	line 78 : connection via MUX	MUX6 channel T1
79	line 79 : " " "	MUX6 channel T2
80	line 80 : " " "	MUX6 channel T3
81	line 81 : " " "	MUX6 channel T4
82	line 82 : " " "	MUX6 channel 1A
83	line 83 : " " "	MUX6 channel 1B
84	line 84 : " " "	MUX6 channel 2A
85	line 85 : " " "	MUX6 channel 2B
86	line 86 : " " "	MUX6 channel 3A
87	line 87 : " " "	MUX6 channel 3B
88	line 88 : " " "	MUX6 channel 4A
89	line 89 : " " "	MUX6 channel 4B
90	line 90 : connection via MUX	MUX7 channel T1
91	line 91 : " " "	MUX7 channel T2
92	line 92 : " " "	MUX7 channel T3
93	line 93 : " " "	MUX7 channel T4
94	line 94 : " " "	MUX7 channel 1A
95	line 95 : " " "	MUX7 channel 1B

Tab. 2-6 Correspondence Between Lines and Channels (cont.)

LNID VALUE	MEANING	CHANNEL
96	line 96 : " " "	MUX7 channel 2A
97	line 97 : " " "	MUX7 channel 2B
98	line 98 : " " "	MUX7 channel 3A
99	line 99 : " " "	MUX7 channel 3B
100	line 100: " " "	MUX7 channel 4A
101	line 101: " " "	MUX7 channel 4B
102	line 102: connection via MUX	MUX8 channel T1
103	line 103: " " "	MUX8 channel T2
104	line 104: " " "	MUX8 channel T3
105	line 105: " " "	MUX8 channel T4
106	line 106: " " "	MUX8 channel 1A
107	line 107: " " "	MUX8 channel 1B
108	line 108: " " "	MUX8 channel 2A
109	line 109: " " "	MUX8 channel 2B
110	line 110: " " "	MUX8 channel 3A
111	line 111: " " "	MUX8 channel 3B
112	line 112: " " "	MUX8 channel 4A
113	line 113: " " "	MUX8 channel 4B

Tab. 2-6 Correspondence Between Lines and Channels

EXAMPLE OF OPENING A LINE

The following example shows how to open a line (identified by the number 2) which is to connect the MOS system to a **tablet,** supposing that:

1. the reception and transmission buffers have been assigned 4096 bytes in the configuration record

2. the user (after opening a line) has assigned the following values, using either the "WRITEINFOLINE" function or the Interpreted Basic "CMD" instruction:

   - Tx timeout (timeout in transmission) = 0
   - Rx timeout (timeout in reception) = 0
   - Echo = 0 (disabled)
   - Rx mode (reception mode) = match characters are not used.

COBOL Call

```
MOVE 0 TO REPLY.
MOVE 2 TO LIND.
MOVE "000010133<<010111" TO LINECONF.
MOVE 0 TO TIMEOUT.
MOVE 0 TO OPENMODE.
CALL "RS_RSOPENLN" USING LNID
                         LINECONF
                         TIMEOUT
                         OPENMODE
                         REPLY.
```

Compiled BASIC Call

```
30 DECLARE NUMERIC R*4
40 R=H"0000"
50 CALL RS_RSOPENLN (2,"000010133<<010111",0,0,R)
```

FORTRAN Call

```
reply = OPENLN (2,"000010133<<010111",0,0)
```

PASCAL+ Call

```
T_LINECONF = array [0..16] of char;
VAR LINECONF: T_LINECONF;
.
.
.
LINECONF: = "000010133<<010111";
REPLY:= OPENLN (2,LINECONF,0,0);
```

Interpreted BASIC Call

```
90 OPEN S,#2,PORT1,"000010133<<010111"
```

The meaning of the values assigned to each parameter in the previous example is given hereby.

The value that has been assigned to the "lnid" parameter, in this case "2", opens line "2".

The string of characters which follows the value "2" is the "lineconf" parameter which is used to define the programming characteristics of the line. The meaning of each character is explained in the following table:

POSITION OF THE CHARACTER	VALUE	MEANING
1	0	Transmission enabled
2	0	RTS = 0
3	0	DTR = 0
4	0	Value of the signal in output from the Peripheral Dependent = 0
5	1	Line used in full duplex
6	0	Number of the stop bits = 1

Tab. 2-7 Example Values of "lineconf" (cont.)

POSITION OF THE CHARACTER	VALUE	MEANING
7	1	Odd parity
8	3	Length of the character being received = 8 bits
9	3	Length of the character being transmitted = 8 bits
10	<	Reception speed = 9600 bps
11	<	Transmission speed = 9600 bps
12	0	No protocol
13	1	Protocol used only in reception
14	0	The line is not closed for diagnostic
15	1	The DSR status changes are ignored
16	1	The CTS status changes are ignored
17	1	The DCD status changes are ignored

Tab. 2-7 Example Values of "lineconf"

The other parameters, timeout and openmode, can assume the value 0 only.

SETTING INTERFACE SOFTWARE PARAMETERS

The initial state of the RS232/CL interface software (after the IPL phase) is partly specified by the parameters used for the channel configuration and partly by the default values.

The **default values** are:

- Line use mode = full duplex
- Protocol used = none
- Transmission timeout = 32767 (equivalent to an infinite time)
- Reception timeout = 32767 (equivalent to an infinite time)
- Echo = disabled
- Reception mode = using the match characters
- Received characters control = all the characters with a hexadecimal value from 00 to 1F inclusive are interpreted as match characters.
- type of line down handling = DSR

The parameters which specify the **channel configuration** are reported in the following list. Each parameter is followed by its default value.

- Number of stop bits = 0 (equivalent to 1 stop bit)
- Parity type = 2 (even parity)
- Length of the characters in reception and transmission = 2 (equivalent to 7 bits)
- Speed in reception and transmission = 0a (2400 bps)
- Length of the reception buffer = 128
- Length of the transmission buffer = 256
- Levels of the reception and transmission buffers (expressed in numbers of characters) for protocols which handle the xon-xoff characters: XON level = 64, XOFF level = 96
- Initial state of the break signals in output, of the RTS, the DTR and the Peripheral Dependent: RTS=1, DTR=1, Peripheral Dependent=0
- Protocol modes XON/XOFF = 1
- Protocol support XON/XOFF =1

3. COBOL INTERFACE

This chapter contains the following information:

- The general structure of a COBOL call for the RS232/CL interface; detailed information on the call for each specific procedure is given in alphabetical order in Chapter 7 of this manual.

- The description of all the parameters that are used in the COBOL procedure.

COBOL CALL

The RS232/CL interface can be called via the following COBOL instruction:

CALL "literal" USING parameter list

---

where "literal" can assume one of the COBOL names given in the Tab. 2-1 in Chapter 2.

The RS232/CL COBOL procedures are contained in the COBOL run-time. The application program must specify the following files in the RTLINK file of linker directives:

for L1 MOS systems: TUB_UI.LIB

for LSX MOS systems: RS_UI.O and
TUB_ui.o or TUB_ui.lib

DESCRIPTION OF THE PARAMETERS

All the parameters used in the COBOL procedures are listed in alphabetical order under the name of the parameter and the following information is given for each of them:

- The parameter structure

- The description of the parameter

chararray

```
01  chararray      PIC X(256).
```

Character vector. The values assigned to this vector depend on the type of information that the user requests.

data$

```
01  data$          PIC X(512).
```

The meaning of this parameter depends on the function it is used in:

- For the "READLINE" function, it is a character vector whose dimensions are user-specified and that contains data that are read from the line identified by "lnid".
- For the "WRITELINE" function, it is a character vector whose dimensions are user-specified and that contains data that are transmitted on a line identified by "lnid".

datasize

```
01  datasize       PIC S9(4) COMP.
```

Integer that gives the maximum number of characters that the user intends reading.

intarray

```
01  intarray
    02  longint    PIC S9(9) COMP OCCURS 4.
```

longint is an integer vector. The values that are assigned depend on the type of parameter on which the user has decide to operate.

lengthread

```
01  lengthread     PIC S9(4) COMP.
```

Gives the number of characters that have been read.

linebuffsts

```
01  linebuffsts
    02  linebuffsts    PIC 9 COMP OCCURS 129
```

Linebuffsts is a boolean array of 129 booleans. The first 113 bits inquires the status of the reception buffer of all the 113 lines.

lineconf

```
01  lineconf.
    02 lineelem   PIC X OCCURS 17.
```

lineelem is a 17-character vector with which the user can define the programming characteristics of the line. (For the values that are allowed and their meaning see Chapter 2.)

lnid

```
01  lnid          PIC S9(4) COMP.
```

Number identifying the line.

openmode

```
01  openmode      PIC S9(9) COMP.
```

Parameter reserved for future use. The present value must be 0.

readmode

```
01  readmode      PIC S9(9) COMP.
```

Parameter reserved for future use. The present value must be 0.

reply

```
01   reply        PIC S9(4) COMP.
```

Integer containing the reply code of the function.

startindex

```
01  startindex    PIC S9(4) COMP.
```

The meaning of this parameter depends on the function it is used in:

- For the "READLINE" function, it contains the index of the user-buffer which indicates where to start loading the data that has been read.
- For the "WRITELINE" function, it contains the index that indicates where the record containing the data to be transmitted begins in the user buffer.

status$

01 status$ PIC S9(4) COMP.

Integer that shows the reception buffer status.

timeout

01 timeout PIC S9(9) COMP.

Specifies the amount of time the user is willing to wait for an event. The present value must be 0.

whatinfo

01 whatinfo PIC S9(4) COMP.

The meaning of this parameter depends on the function in which it is used:

- For the "WRITEINFOLINE", this indicates the type of parameters on which the user wishes to operate.
- For the "READINFOLINE", this indicates the information that the user wishes to obtain.

writemode

01 writemode PIC S9(9) COMP.

Parameter reserved for future use. The present value must be 0.

4. COMPILED BASIC INTERFACE

This chapter contains the following information:

- The general structure of a Compiled BASIC call for the RS232/CL interface; detailed information on the call for each specific procedure is given in alphabetical order in Chapter 7 of this manual.
- The description of all the parameters that are used in the Compiled BASIC procedure.

COMPILED BASIC CALL

The request for using the RS232/CL interface called by the following Compiled BASIC instruction:

CALL "literal" list of parameters

---

where the value that can be assumed by "literal" is one of the Compiled BASIC names shown in Tab. 2-1 in Chapter 2.

The following instruction must be inserted in order to call the Compiled BASIC functions:

DECLARE SYSTEM PROCEDURE "literal" (list of parameters)

The application program must specify the following files in the link.cmd file of linker directives:

for L1 MOS systems: TUB_UI.LIB

for LSX MOS systems: RS_UI.O and
TUB_ui.o or TUB_ui.lib

The complete list of DECLARE SYSTEM PROCEDURE needed to call the BASIC functions is:

```
DECLARE SYSTEM PROCEDURE RS_RSATTLN (LNID%I,REPLY%I)
DECLARE SYSTEM PROCEDURE RS_RSCLOSELN (LNID%I,REPLY%I)
DECLARE SYSTEM PROCEDURE RS_RSDETLN (LNID%I,REPLY%I)
DECLARE SYSTEM PROCEDURE RS_RSOPENLN (LNID%I,LINECONF$%17,TIMEOUT%9,    &
&                                     OPENMODE%9,REPLY%I)
DECLARE SYSTEM PROCEDURE RS_RSRASTLN (STATUS%I)
DECLARE SYSTEM PROCEDURE RS_RSREADLN (LNID%I,DATA$%512,STARTINDEX%I,    &
&                                     DATASIZE%I,READMODE%9,            &
&                                     LENGHTREAD%I,REPLY%I)
DECLARE SYSTEM PROCEDURE RS_RSRINFOLN (LNID%I,WHATINFO%I,INTARRAY()%9,  &
&                                      CHARARRAY$%256,REPLY%I)
DECLARE SYSTEM PROCEDURE RS_RSWINFOLN (LNID%I,WHATINFO%I,INTARRAY()%9,  &
&                                      CHARARRAY$%256,REPLY%I)
DECLARE SYSTEM PROCEDURE RS_RSWONF (REPLY%I)
DECLARE SYSTEM PROCEDURE RS_RSWRITELN (LNID%I,DATA$%512,STARTINDEX%I,   &
&                                      DATASIZE%I,WRITEMODE%9,REPLY%I)
DECLARE SYSTEM PROCEDURE RS_READSYSLN (STATUS%, LINBUFFSTS$%129)
```

DESCRIPTION OF THE PARAMETERS

All the parameters used in the COMPILED BASIC procedures are listed in alphabetical order according to the name of the parameter, and the following information is given for each of them:

- The parameter structure
- The description of the parameter

chararray$

chararray$*256

Character vector. The values assigned to this vector depend on the type of parameter that the user wishes to operate on.

data$

data$512

The meaning of this parameter depends on the function that it is used in:

- For the "READLINE" function, this is a character vector whose dimensions are user-specified and that contains the data that are read from the line identified by "lnid".
- For the "WRITELINE" function, this a character vector whose dimensions are user-specified and that contains the data that are transmitted on a line identified by "lnid".

datasize

datasize*i

Integer that indicates the maximum number of characters that the user intends reading.

intarray

intarray()*9

where the array variable given as the argument must be declared as follows:

DECLARE NUMERIC intarray(4)*9

integer vector, the values assigned depend on the type of parameter on which the user has decided to operate.

lengthread

lengthread*i

Indicates how many parameters have been read.

linebuffsts

linebuffsts $*129

linebuffsts is a booolean array of 129 booleans. The first 113 bits inquires the status of the reception buffer of all the 113 lines.

lineconf$

lineconf$*17

17-character vector used by the user to define the programming characteristics of the line. (For the values that are allowed and their meaning see Chapter 2 in this manual.)

lnid

lnid*i

Number identifying the line.

openmode

openmode*9

Parameter reserved for future use. The present value must be 0.

readmode

readmode*9

Parameter reserved for future use. The present value must be 0.

reply

reply*i

Integer that contains the reply code of the function.

startindex

startindex*i

The meaning of this parameter depends on the function it is used in:

- For the "READLINE" function, it contains the index of the user-buffer which indicates where to start loading the data that has been read.
- For the "WRITELINE" function, it contains the index that indicates where the record containing the data to be transmitted begins in the user buffer.

status

status*i

Integer that indicates the status of the reception buffer.

timeout

timeout*9

Specifies how long the user is willing to wait for an event. The present value must be 0.

whatinfo

whatinfo*i

The meaning of this parameter depends on the function in which it is to be used:

- For the "WRITEINFOLINE" function, this indicates the type of parameter the user wishes to operate on.
- For the "READINFOLINE" function, this indicates the type of information that the user wishes to obtain.

writemode

writemode*9

Parameter reserved for future use. The present value must be 0.

5. FORTRAN INTERFACE (ONLY FOR L1 MOS SYSTEMS)

This chapter contains the following information:

- The general structure of a FORTRAN call for the RS232/CL interface; detailed information on calling each specific procedure is given in alphabetical order in Chapter 7 of this manual.
- The description of all the parameters that are used in FORTRAN functions.

FORTRAN CALL

The RS232/CL interface can be called via the following FORTRAN instruction:

```
reply = literal (parameter list)
```

where "reply" is an integer FORTRAN variable and "literal" can assume one of the FORTRAN names given in Tab. 2-1 in Chapter 2.

To access the FORTRAN functions, the calling program must contain the instruction:

```
INCLUDE '/IPL/FORTRAN_77.P/INCL/RS232.H'
```

Furthermore, the following files must be linked to the application program:

```
/IPL/FORTRAN_77.P/LIB/RS232LIB
/IPL/DPC/COMMON/INTERF/TUB_ui.obj
```

DESCRIPTION OF THE PARAMETERS

All the parameters used in the FORTRAN functions are given in alphabetical order under the name of the parameter, and the following information is given for each of them:

- The parameter structure
- The description of the parameter

chararray

character chararray (0:255)

Character arrays. The values assigned to this array depend on the type of information that the user requests.

data

dimension data (512)
character data

The meaning of this parameter depends on the function it is used in:

- For the "READLINE" function, it is a character array whose dimensions are user-specified and that contain data that are read from a line identified by "lnid".
- For the "WRITELINE" function, it is a character array whose dimensions are user-specified and that contain the data that are transmitted on the line indicated by "lnid".

datasize

integer datasize

Integer that indicates the maximum number of characters that the user intends reading.

intarray

integer intarray (0:3)

Integer array. The meaning of the values assigned to this array depends on the type of parameter on which the user has decided to operate.

lengthread

integer lengthread

Parameter that indicates the number of characters that have been read.

linebuffsts

logical linebuffsts(1:129)

Array of 129 booleans. The first 113 bits inquires the status of the reception buffer of all the 113 lines.

lineconf

character lineconf(17)

17-character array with which the user defines the programming characteristics of the line. (For the permitted values and their meaning, see Chapter 2.)

lnid

integer lnid

Number identifying the line.

openmode

integer openmode

Parameter reserved for future use. The present value must be 0.

readmode

integer readmode

Parameter reserved for future use. The present value must be 0.

reply

integer reply

Integer containing the reply code of the function.

startindex

integer startindex

The meaning of this parameter depends on the function in which it is used:

- For the "READLINE" function it contains the index of the user-buffer which indicates where to start loading the data that has been read.
- For the "WRITELINE" it contains the index that indicates where the record containing the data to be transmitted starts in the user buffer.

status

integer status

Integer indicating the reception buffer status.

timeout

integer timeout

The amount of time the user is willing to wait for an event. The present value must be 0.

whatinfo

integer whatinfo

The meaning of this parameter depends on the function in which it is used:

- For the "WRITELINE" function, it indicates the type of parameter on which the user wishes to operate.
- For the "READINFOLINE" function, it indicates the type of information that the user wishes to obtain.

writemode

integer writemode

Parameter reserved for future use. The present value must be 0.

6. PASCAL+ INTERFACE

This chapter contains the following information:

- The general structure of a PASCAL+ call for the RS2323 interface; detailed information on the call for each specific procedure is given in alphabetical order in Chapter 7 in this manual.

- The description of all the parameters used in the PASCAL+ primitives.

PASCAL+ CALL

The RS232/CL interface can be called via the following PASCAL+ instruction:

```
reply := literal (parameter list);
```

where "literal" can assume one of the PASCAL+ names given in Tab. 2-1 in Chapter 2.

To access the PASCAL+ primitives, the program must include the following modules:

ln_t.i (contains all types of RS232 drivers)
ln_p.i (contains all the definitions of the primitives)

Furthermore, either the following interface file:

for L1 MOS systems: TUB_ui.obj

for LSX MOS systems: TUB_ui.o

or, alternately (for L1 and LSX MOS systems), the library TUB_ui.lib

must be linked to the program.

chararray

The parameter must be passed as a variable and defined as follows:

```
T_chararray = array [0..255] of char;

chararray : T_chararray;
```

chararray is a character array. The value assigned to this depends on the type of parameter that the user wishes to operate on.

data

The parameter must be passed as a variable and defined as follows:

```
data : packed array [min..max : integer] of dataltem;
```

The meaning of this parameter depends on the primitive in which it is used:

- For the "READLINE" function, it is a character array whose dimensions are user-specified and that contains the data to be received from the line indicated by "lnid".
- For the "WRITELINE" function, it is a character array whose dimensions are user-specified and that contains the data to be transmitted on the line indicated by "lnid".

datasize

The parameter must be passed as a value and defined as follows:

```
datasize : integer;
```

datasize is an integer that indicates the maximum number of characters that the user wants to read.

intarray

The parameter must be passed as a variable and defined as follows:

```
T_intarray = array [0..3] of integer;

intarray : T_intarray;
```

intarray is an integer array. The values assigned to it depend on the type of parameter that the user wishes to operate on.

lengthread

The parameter must be passed as a variable and defined as follows:

lengthread : integer;

lengthread indicates how many characters have been read.

linebuffsts

The parameter must be passed as a variable and defined as follows:

T_linebuffsts = packed array [1...129] of booleans;

linebuffsts : T_linebuffsts;

It is a boolean array of 129 booleans. The first 113 bits inquires the status of the reception buffer of all the 113 lines.

lineconf

The parameter must be passed as a variable and defined as follows:

T_lineconf = array [0..16] of char;

lineconf : T_lineconf;

17-character array with which the user can define the programming characteristics of the line. (For the permitted values and their meaning, see Chapter 2 in this manual.)

lnid

The parameter must be passed as a value and defined as follows:

lnid : integer;

lnid is a number that identifies the line.

openmode

The parameter must be passed as a value and defined as follows:

openmode : integer;

Parameter reserved for future use. The present value must be 0.

readmode

The parameter must be passed as a value and defined as follows:

readmode : integer;

Parameter reserved for future use. The present value must be 0.

reply

The parameter must be defined as follows:

reply : integer;

Integer that contains the reply code for the function.

startindex

The parameter must be passed as a value and defined as follows:

startindex : integer;

The meaning of this parameter depends on the primitive in which it is used:

- For the "READLINE" primitive it contains the index of the user-buffer which indicates where to start loading the data that has been read.
- For the "WRITELINE" primitive it contains the index that indicates where the record containing the data to be transmitted starts in the user buffer.

status

The parameter must be defined as follows:

status : integer;

Integer indicating the reception buffer status.

timeout

The parameter must be passed as a value and defined as follows:

timeout : integer;

timeout specifies the amount of time that the user is prepared to wait to access the resource. The present value must be 0.

whatinfo

The parameter must be passed as a value and defined as follows:

whatinfo : integer;

The meaning of this parameter depends on the primitive in which it is used:

- For the "WRITELINE" function, this indicates the type of parameter that the user wishes to operate on.
- For the "READINFOLINE" function, it indicates the type of information that the user wishes to obtain.

writemode

The parameter must be passed as a value and defined as follows:

writemode : integer;

Parameter reserved for future use. The present value must be 0.

7. RS232/CL SERIAL INTERFACE CALLS

This chapter contains a detailed description of all the functions provided by the RS232/CL interface software. They are given in alphabetical order, according to their global names.

The description covers:

- the activity carried out by the function
- the calling syntax from COBOL, Compiled BASIC, FORTRAN e PASCAL+
- the following aspects for each parameter:
  - its name
  - if it is an input or output parameter
  - a brief description
- the values that the reply code can assume
- any characteristics.

This function is used to attach a line to a line pool.

COBOL CALL

```
CALL "RS_RSATTLN" USING lnid, reply.
```

COMPILED BASIC CALL

```
CALL RS_RSATTLN (lnid, reply)
```

FORTRAN CALL

```
reply = ATTACH (lnid )
```

PASCAL CALL

```
reply := ATTACHLN (lnid);
```

The parameters used in this function are described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
input parameter:	
lnid	Identifies the line. The value must be in the range of 1 to 113 inclusive. For a full explanation of this parameter see "IDENTIFICATION OF THE LINE" in Chapter 2.
output parameter:	
reply	Reply code.

The primitive returns one of the following replies:

- 0 Operation terminated correctly.
- 1 An invalid identifier has been given.
- 8 Illegal operation requested.

Ends data exchange on a line.

COBOL CALL

```
CALL "RS_RSCLOSELN" USING lnid, reply.
```

COMPILED BASIC CALL

```
CALL RS_RSCLOSELN (lnid, reply)
```

FORTRAN CALL

```
reply = CLOSLN (lnid)
```

PASCAL CALL

```
reply := CLOSELN (lnid);
```

The parameters for this function are described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
INPUT	
lnid	Identifies the line. The value must be in the range of 1 to 113 inclusive. for a full explanation of this parameter see "IDENTIFICATION OF THE LINE" in Chapter 2.
OUTPUT	
reply	Reply code.

The primitive returns one of the following replies:

- 0 Operation terminated correctly.
- 1 An invalid identifier has been given.
- 8 Illegal operation requested.

Characteristic

All the lines opened by a FORTRAN program are automatically closed in case of abort of the program.

This function is used to detach a line from a line pool.

COBOL CALL

```
CALL "RS_RSDETLN" USING lnid, reply.
```

COMPILED BASIC CALL

```
CALL RS_RSDETLN (lnid, reply)
```

FORTRAN CALL

```
reply = DETACH (lnid)
```

PASCAL CALL

```
reply := DETACHLN (lnid);
```

The parameter used in this function are described in the following table (see Chapters 3, 4, 5, and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
INPUT	
lnid	Identifies the line. The value must be in the range of 1 to 113 inclusive. For a full explanation of this parameter see "IDENTIFICATION OF THE LINE" in Chapter 2.
OUTPUT	
reply	Reply code.

The primitive returns one of the following replies:

- 0 Operation terminated correctly.
- 1 An invalid identifier has been given.
- 8 Illegal operation requested.

This function is used to execute data exchange.

COBOL CALL

```
CALL "RS_RSOPENLN" USING lnid, lineconf, timeout, openmode, reply.
```

COMPILED BASIC CALL

```
CALL RS_RSOPENLN (lnid, lineconf$, timeout, openmode, reply)
```

FORTRAN CALL

```
reply = OPENLN (lnid, lineconf, timeout, openmode)
```

PASCAL CALL

```
reply := OPENLN (lnid, lineconf, timeout, openmode);
```

The parameters used in this function are described in the following table (for their structure see Chapters 3, 4, 5 and 6 in this manual):

PARAMETER NAME	DESCRIPTION
INPUT	
lnid	Identifies the line. The value must be in the range of 1 to 113 inclusive. For a full explanation of this parameter see "IDENTIFICATION OF THE LINE" in Chapter 2.
lineconf	Parameter allowing the user to program the line controller (see Chapter 2).
timeout	The present value for this parameter must be 0.
openmode	The present value for this parameter must be 0.
OUTPUT	
reply	Reply code.

The primitive returns one of the following replies :

- 0 Operation terminated correctly.
- 1 An invalid identifier has been given.
- 2 One or more invalid parameters have been given.
- 4 Line busy.

Characteristics

1. If the protocol handles the break, it is not considered to be an error during reception.

   The protocol that handles the xon and xoff characters and the protocol that handles the break can be used both for checking how full the reception buffer is (protocol used during transmission) and for stopping and restarting a writing session (protocol used in reception).

   The protocol that handles the busy signal can be used for starting and restarting a writing session. In the present line control this protocol is implemented using the CTS signal.

2. The "OPENLINE" function supplies the line controller, the output signals and the driver with their initial value, as they are indicated in the driver configuration record, in the "lineconf" parameter or by assuming them by default.

   If the character ? (hex 3F) is specified as the initial value for the "lineconf" parameter, the line programming characteristics assumed are those indicated in the configuration record and those assumed by default.

3. A line opened (by means of the "OPENLINE" function) can only be used by one user at a time.
   This user must release the line by means of the "CLOSELINE" function.

4. When the "OPENLINE" function is called, the reception buffer is emptied.

5. All the functions must always be called during a session that is started by an opening operation and terminated with a closing operation (except for the functions that return or update the line state). A function that is called outside a session emits error code 8 (an illegal operation has been requested).

This function is used to read the state of the reception buffer of the lines 1 to 15.

This function is mantained only for compatibility with earlier releases. From release 5.0 on the user should use the READSYSLINE function which inquires the status of the reception buffer of up to 113 lines.

COBOL CALL

```
CALL "RS_RSRASTLN" USING status$.
```

COMPILED BASIC CALL

```
CALL RS_RSRASTLN (status)
```

FORTRAN CALL

```
status = RDSTLN
```

PASCAL CALL

```
status := READALLSTLN;
```

The parameters used in this function are described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
OUTPUT	
status	Reception buffer state.

Characteristic

This function returns a word: each bit corresponds to a line and indicates if there is at least one character that has not been read (bit value = 1) in the reception buffer, or if the buffer is empty (bit value = 0). The meaning of the bits is as follows:

BIT POSITION	MEANING	CHANNEL
0	reserved	
1	line 1 : connection via CPU	CPU
2	line 2 : connection via Twin	Twin 1 channel A
3	line 3 : " " "	Twin 1 channel B
4	line 4 : " " "	Twin 2 channel A
5	line 5 : " " "	Twin 2 channel B
6	line 6 : " " "	Twin 3 channel A
7	line 7 : " " "	Twin 3 channel B
8	line 8 : " " "	Twin 4 channel A
9	line 9 : " " "	Twin 4 channel B
10	line 10 : " " "	Twin 5 channel A
11	line 11 : " " "	Twin 5 channel B
12	line 12 : " " "	Twin 6 channel A
13	line 13 : " " "	Twin 6 channel B
14	line 14 : " " "	Twin 7 channel A
15	line 15 : " " "	Twin 7 channel B

Tab. 7-1 Correspondence between Lines and Channels

This function reads the states of the line.

COBOL CALL

```
CALL "RS_RSRINFOLN" USING lnid, whatinfo,intarray, chararray, reply.
```

COMPILED BASIC CALL

```
CALL RS_RSRINFOLN (lnid, whatinfo, intarray(), chararray$, reply)
```

FORTRAN CALL

```
reply = RDINFO (lnid, whatinfo, intarray, chararray)
```

PASCAL CALL

```
reply := READINFOLN (lnid, whatinfo, intarray, chararray);
```

The parameters that are used in this function are described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
INPUT	
lnid	Identifies the line. The value must be in the range of 1 to 113 inclusive. For a full explanation of this parameter see "IDENTIFICATION OF THE LINE" in Chapter 2,
whatinfo	This parameter indicates the type of information the user wants to receive. The accepted values are: 0 = rxMode (reception mode) 1 = txMode (transmission mode) 2 = lineStatus (state of the input/output signals, and number of characters in the reception buffer) 3 = lineConf (programming characteristics of the line) 4 = rxStatus (reception state)
OUTPUT	
chararray	The meaning of the values in the parameter depends on the type of information the user has requested in the "whatinfo" parameter. For detailed information see the following Characteristic.
intarray	The meaning of the values in the parameter depends on the type of information the user has requested in the "whatinfo" parameter. For a detailed information see the following Characteristic.
reply	Reply code.

The primitive returns one of the following replies:

- 0 Operation terminated correctly

- 1 An invalid identifier has been given

- 2 One or more invalid parameters have been given

Characteristic

The meaning of the information which can be obtained as follows:

rxMode : The intarray[0] and intarray[1] parameters specify how long the user will wait for the completion of any read.

intarray[0] specifies the time which the user is prepared to wait for the arrival of the first character when the reception buffer is empty. The time unit is 100 msec.

intarray[1] specifies the time which the user is prepared to wait to receive each character after the first one.

The possible values for these two parameters are those included in the interval from 0 to +32767. The value 32767 implies an infinite waiting time. When the timeout has expired, the characters that have been received are sent to the user buffer.

The intarray[2] parameter indicates whether a received character has been sent to the transmitter (echo) and which line has been used for this purpose.
The possible values are:

intarray[2]	meaning
0	The echo is not activated.
X	The echo is activated by using the 'X' line.

The intarray[3] parameter indicates whether the user is receiving in match mode or in transparent mode.
The possible values are:

intarray[3]	meaning
0	Match mode
32767	Transparent mode

The chararray array contains the current match table as indicated in the WRITEINFOLINE instruction.

txMode : The intarray[0] parameter indicates the timeout currently used in transmission, i.e. how long the user is prepared to wait for completion of each write operation. The possible values are from 0 to 32767. The time unit is equal to 100 msec. The value 32767 signifies an infinite time.

lineStatus: The intarray[0] parameter indicates the number of characters in the reception buffer which have not yet been read; intarray [1], intarray [2] and intarray [3] are not used.

The chararray array indicates the state of the input and output signals. The value for each of its parameters be 0 (hex 30) or 1 (hex 31) for signal off or signal on respectively.

The meanings of the parameters are as follows:

parameter	signal
chararray[0]	break in output
chararray[1]	RTS (Request To Send)
chararray[2]	DTR (Data Terminal Ready)
chararray[3]	output of the Peripheral Dependent
chararray[4]	break in input
chararray[5]	busy (in the current use this is the same as the CTS)
chararray[6]	DSR (Data Set Ready)
chararray[7]	CTS (Clear To Send)
chararray[8]	DCD (Data Carrier Detector)

lineConf: The intarray[0] parameter indicates the size of the reception buffer in numbers of characters.

The intarray[1] parameter indicates the size of the transmission buffer in numbers of characters.

The parameters from chararray[0] to chararray[16] indicate the values which have been assigned to the line parameters for line programming. Their meaning is described in the section "Programming Characteristics of the Line" in Chapter 2 of this manual.

rxStatus : The chararray array gives indications on the state of reception, and each of its parameters is either 0 (OK) or 1 (if an anomaly is found).

The meanings of the parameters are as follows:

parameter	anomaly found
chararray[0]	framing error
chararray[1]	parity error
chararray[2]	overrun
chararray[3]	break (only for a protocol which does not handle the break signal)
chararray[4]	line down
chararray[5]	the data received exceed the size of the reception buffer (overflow)

These types of anomaly prevent the correct execution of a "READLINE", which will give the error code 16 (line error, see "READLINE"). The anomalies mentioned above are cleared once they have been taken from the "READINFOLINE" function. If the detected anomaly was overflow (chararray[5]=1) the reception buffer is cleared.

Note: See also the "WRITEINFOLINE" function.

This function allows the data of a line to be read.

COBOL CALL

```
CALL "RS_RSREADLN" USING lnid, data$,startindex, datasize,
                         readmode, lengthread, reply.
```

COMPILED BASIC CALL

```
CALL RS_RSREADLN (lnid, data$, startindex, datasize,
                  readmode,lengthread, reply)
```

FORTRAN CALL

```
reply = READLN (lnid, data, startindex, datasize, readmode, lengthread)
```

PASCAL CALL

```
reply := READLN (lnid, data, startindex, datasize, readmode, lengthread);
```

The parameters used in this function are described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
INPUT	
lnid	Identifies the line. The value must be in the range of 1 to 113 inclusive. For a full explanation of this parameter see "IDENTIFICATION OF THE LINE" in Chapter 2.
startindex	Index from which the record containing the received data starts in the user buffer.
datasize	Value which indicates the maximum length of the data which the user intends to read.
readmode	At present the value for this parameter must be 0.
OUTPUT	
data	Data which are read from the line indicated by the "lnid" parameter.
lengthread	This parameter indicates the number of characters which have been read.
reply	Reply code.

The primitive returns one of the following replies:

- 0 Operation terminated correctly.

- 1 An invalid identifier has been given.

- 2 One or more invalid parameters have been given.

- 8 Illegal operation requested.

- 16 Line error which may be due to:

    - line down
    - reception buffer overflow
    - overrun
    - parity error
    - framing error
    - break in input.

- 32 Reading unsuccessful.

- 64 The time out has expired or all the data present on the line have been read.

Characteristics

By using the "rxmode" parameter of the "WRITEINFOLINE" function the user can specify the reading mode (timeout, echo, etc.) and choose whether to operate in transparent or match mode in particular (see "WRITEINFOLINE").

In transparent mode each "READLINE" function is considered to be complete when the same number of characters has been received as in the "datasize" parameter, unless reading is interrupted by error codes 16, 64 (see Table 2-2 in Chapter 2) or 32 (caused by a "CLOSELINE" operation).

If the user selects match mode, a "READLINE" operaton is considered to be complete when the same number of characters has been received, after a match character, as in the value corresponding to the character in the current match table (see "WRITEINFOLINE"). This is so unless the same number of characters is received, before the match character, as in the "datasize" parameter value, or reading is interrupted by error codes 16, 64 (see Tab. 2-2 in Chapter 2) or 32 (caused by a "CLOSELINE" operation).

The match characters that are received are stored in the user-buffer with the data.

The number of characters that is received (excluding the match ones) is indicated by the "lengthread" parameter; a check can also be made to see if any match characters have been received, by comparing the value of the "lengthread" character with that of the "datasize" parameter.

A difference between the two parameters can also occur if the "READLINE" operation is interrupted by either error code 64 (see Table 2-2 in Chapter 2) or 32 (caused by a "CLOSELINE" operation).

This function is used to read the state of the reception buffer of the lines 1 to 15.

This function is mantained only for compatibility with earlier releases. From release 5.0 on the user should use the READSYSLINE function which inquires the status of the reception buffer of up to 113 lines.

COBOL CALL

```
CALL "RS_RSRASTLN" USING status$.
```

COMPILED BASIC CALL

```
CALL RS_RSRASTLN (status)
```

FORTRAN CALL

```
status = RDSTLN
```

PASCAL CALL

```
status := READALLSTLN;
```

The parameters used in this function are described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
OUTPUT	
status	Reception buffer state.

Characteristic

This function returns a word: each bit corresponds to a line and indicates if there is at least one character that has not been read (bit value = 1) in the reception buffer, or if the buffer is empty (bit value = 0). The meaning of the bits is as follows:

BIT POSITION	MEANING	CHANNEL
0	reserved	
1	line 1 : connection via CPU	CPU
2	line 2 : connection via Twin	Twin 1 channel A
3	line 3 : " " "	Twin 1 channel B
4	line 4 : " " "	Twin 2 channel A
5	line 5 : " " "	Twin 2 channel B
6	line 6 : " " "	Twin 3 channel A
7	line 7 : " " "	Twin 3 channel B
8	line 8 : " " "	Twin 4 channel A
9	line 9 : " " "	Twin 4 channel B
10	line 10 : " " "	Twin 5 channel A
11	line 11 : " " "	Twin 5 channel B
12	line 12 : " " "	Twin 6 channel A
13	line 13 : " " "	Twin 6 channel B
14	line 14 : " " "	Twin 7 channel A
15	line 15 : " " "	Twin 7 channel B

Tab. 7-1 Correspondence between Lines and Channels

This function allows the user to wait to receive data from any one of the lines attached to the pool.

COBOL CALL

```
CALL  "RS_RSWONP" USING reply.
```

COMPILED BASIC CALL

```
CALL RS_RSWONP (reply)
```

FORTRAN CALL

```
reply = WAITPL
```

PASCAL CALL

```
reply := WAITONPOOL;
```

The parameter that is used in this function is described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
OUTPUT	
reply	Reply code.

This function can return the following reply:

- 0 There are no lines open.

Characteristics

This function returns a value that is the identifier for a line attached to a pool from which one or more characters have been received.

This function allows the programming line status and the driver modes to be set or updated.

COBOL CALL

```
CALL "RS_RSWINFOLN" USING lnid, whatinfo, intarray, chararray, reply.
```

COMPILED BASIC CALL

```
CALL RS_RSWINFOLN (lnid, whatinfo, intarray, chararray$, reply)
```

FORTRAN CALL

```
reply = WRINFO (lnid, whatinfo, intarray, chararray)
```

PASCAL CALL

```
reply := WRITEINFOLN (lnid, whatinfo, intarray, chararray);
```

The parameters used in this function are described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
INPUT	
lnid	Identifies the line. The value must be in the range of 1 to 113 inclusive. For a full explanation of this parameter see "IDENTIFICATION OF THE LINE" in Chapter 2.
whatinfo	This parameter indicates the type of line status the user wants to set or update. The accepted values are: 0 = rxMode (reception mode) 1 = txMode (transmission mode) 2 = lineStatus (state of the lines) 3 = rxFlush (set to zero) 4 = txFlush (transmission is set to zero) 5 = rxStart (starts the reception) 6 = rxStop (stops the reception) 7 = txStart (starts the transmission)
intarray	Integer vector. The meaning of the values assigned to this vector depends on the type of parameter on which the user intends operating. See Characteristic.
chararray	Character vector. The meaning of the values assigned to this vector depend on the type of parameter on which the user intends operating. See Characteristic.
OUTPUT	
reply	Reply code.

The function returns one of the following replies:

- 0 Operation terminated correctly
- 1 An invalid identifier has been given
- 2 One or more identifier parameters have been given
- 8 Illegal operation requested

Characteristics

The meanings of the parameters on which the user can intervene are as follows:

rxMode: The intarray[0] and intarray[1] parameters specify the time which the user is prepared to wait for completion of any read operation. The time unit is 100 ms.

intarray[0] specifies the time which the user is prepared to wait to received the first character, when the reception buffer is empty.

intarray[1] specifies the time which the user is prepared to wait to received each character after the first one.

The values accepted for these two parameters are those from 0 to +32767. The value 32767 is interpreted as an infinite time. A negative value leaves the existing timeout unchanged. When the timeout has expired, the characters already received are sent to the user buffer.

The intarray[2] parameter specifies if a character must be sent on to the transmitter (echo). This is only possible if the line is used in "full duplex". The accepted values are:

intarray[2]	meaning
< 0	The parameter value does not change.
0	Echo disabled.
X	Echo enabled using the "X" line.

If the user wants to activate the 'echo' using the same line that he is using for reception, he must indicate the identifier for that line in the intarray[2] parameter.

The intarray[3] parameter specifies whether the user intends terminating reception when a match character has been received, or when the number of characters indicated in the datasize parameter of the "READLINE" primitive has been received (transparent mode).

The accepted values are:

intarray[3]	meaning
< 0	The parameter value remains unchanged.
0	Match mode is used.
32767	Transparent mode is used.

Values other than those indicated are reserved for future use and are therefore unacceptable at the present time.

The chararray array specifies the type of control to be carried out for each of the characters which may be received. The hexadecimal value of the character is used as an index for accessing the actual array, and the control to be carried out is specified in the corresponding position. This array allows the user to indicate which characters are not to be controlled, which are to be ignored in input, and which are to be considered to be 'match' characters. These characters put the driver in the 'end of input' state: in other words, the driver closes the "READLINE" after retrieving a specified number of characters (max. 15, as specified in the chararray array at the position of the match character) from the reception buffer. The accepted values are:

chararray array contents in the position corresponding to "char", that is chararray[ord(char)]	control executed
0 (hex = 30)	The corresponding character is not controlled.
1 (hex = 31)	The corresponding character is ignored in input.
from @ to 0 (from hex = 40 to hex = 4F inclusive)	The corresponding character is a 'match' character. Each "READLINE" is closed after the reception of a number of characters equal to the difference between (ord(chararray[ord(char)] and (ord(@)).

If the value '?' (hex = 3F) is specified in the chararray[0] parameter the current match table remains unchanged.

The policy of a "READLINE" in match mode is illustrated in the following figure.

The variable 'datasize' is initialized with the value of the 'datasize' parameter of the "READLINE" function. The variable 'matchCounter' counts the number of 'match' characters received and is appropriately updated by chararray[ord(char)] as soon as the first 'match' character has been retrieved from the reception buffer.

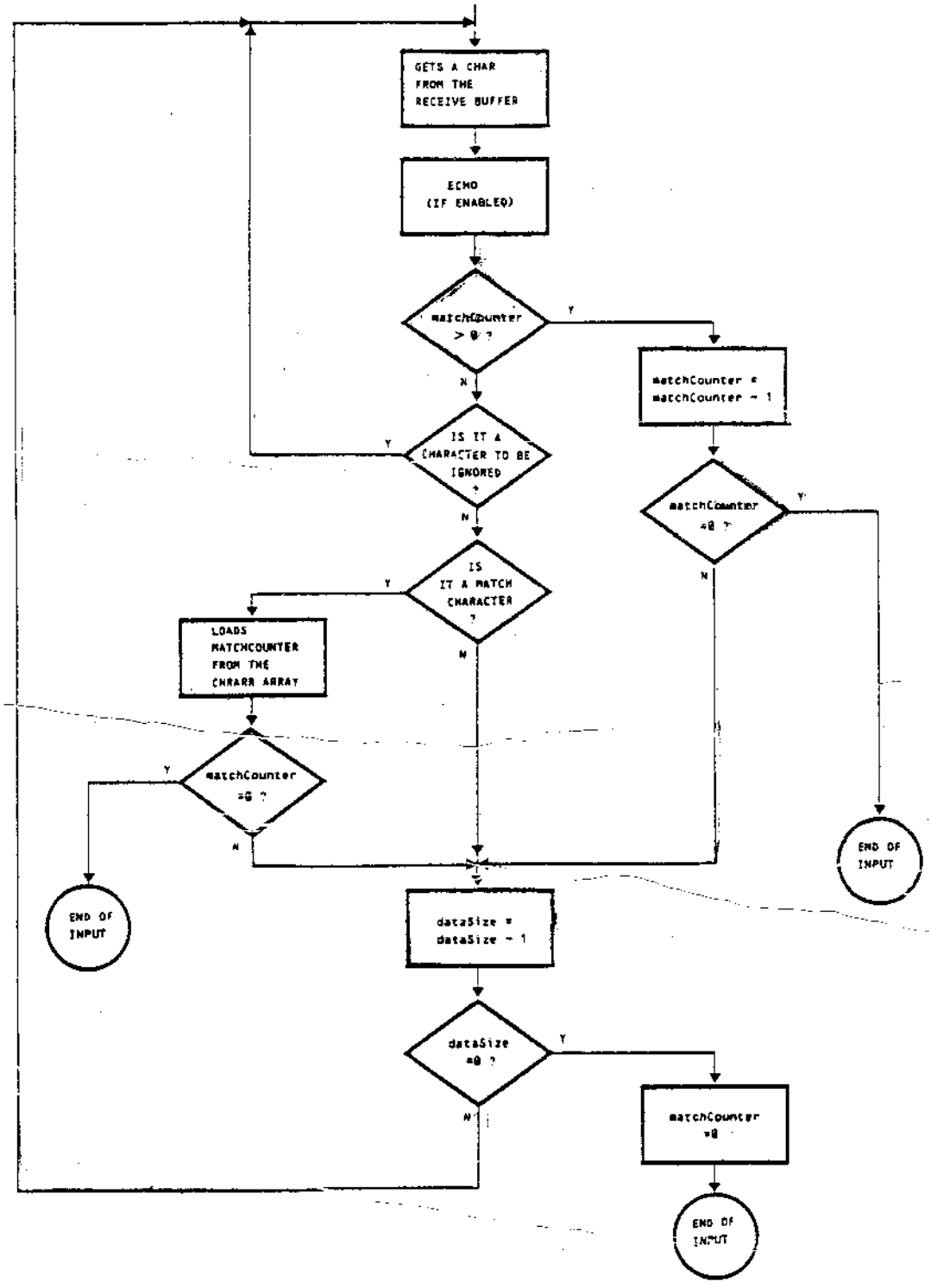


Fig. 7-2 Diagram of a READLINE Function in Match Mode

txMode: The intarray[0] parameter specifies how long the user is willing to wait for the completion of each successive write function. The accepted values are those from 0 to 32767. The time unit is equal to 100 msec. The value 32767 is interpreted as an infinite time.

lineStatus: The user can set up the output signals and the break in output using the chararray array. The accepted values are:

chararray[x]	meaning
?	The signal remains unchanged.
0	The signal value is set to 0.
1	The signal value is set to 1.

The parameters have the following meaning:

parameter	signal whose state is set up
chararray[0]	break in output
chararray[1]	RTS (Request To Send)
chararray[2]	DTR (Data Terminal Ready)
chararray[3]	output of the Peripheral Dependent

rxFlush: The reception buffer is cleared. The rest of the chararray array is not used and neither is intarray.

txFlush: The transmission buffer is set to zero. The status of transmission is mantained (stopped or started).

rxStart Starts the reception.

rxStop Stops the reception.

txStart Starts the transmission.

This function allows data to be sent on line.

COBOL CALL

```
CALL "RS_RSWRITELN" USING lnid, data$, startindex, datasize, writemode,
                          reply.
```

COMPILED BASIC CALL

```
CALL RS_RSWRITELN (lnid, data$, startindex, datasize, writemode, reply)
```

FORTRAN CALL

```
reply = WRTLN (lnid, data, startindex, datasize, writemode)
```

PASCAL CALL

```
reply := WRITELN (lnid, data, startindex, datasize, writemode);
```

The parameters used in this function are described in the following table (see Chapters 3, 4, 5 and 6 in this manual for their structure):

PARAMETER NAME	DESCRIPTION
input parameter:	
lnid	Identifies the line. The value must be in the range of 1 to 113 inclusive. For a full explanation of this parameter see "IDENTIFICATION OF THE LINE" in Chapter 2.
data	Data which are sent on the line indicated by the "lnid" parameter.
startindex	Index from which the record containing the data to be sent on line starts within the user buffer.
datasize	Value which indicates the length of the data record which the user intends to send on line.
writemode	At present the value for this parameter must be 0.
output parameter:	
reply	Reply code.

This function returns one of the following replies:

- 0 Operation terminated correctly.
- 1 An invalid identifier has been given.
- 2 One or more invalid parameters have been given.
- 8 Illegal operation requested.
- 16 Line down.
- 32 The write activity is unsuccessful.
- 64 Time out expired.

Characteristic

If the write time out (see "WRITEINFOLINE") expires before the write function has been completed, the error code 64 is given.

8. INTERPRETED BASIC INTERFACE (ONLY FOR L1 MOS SYSTEMS)

This chapter contains a detailed description of all the instructions offered by the Interpreted BASIC application environment for programming the RS232/CL interface.

They are listed in alphabetical order, and here briefly presented:

Open/Close Line

- OPEN
- CLOSE

Wait Data From Line

- ON SRQ GOSUB

Exchange Data

- INPUT
- RBYTE
- PRINT
- WBYTE

Read/Update the State of the Line

- TEST
- CMD

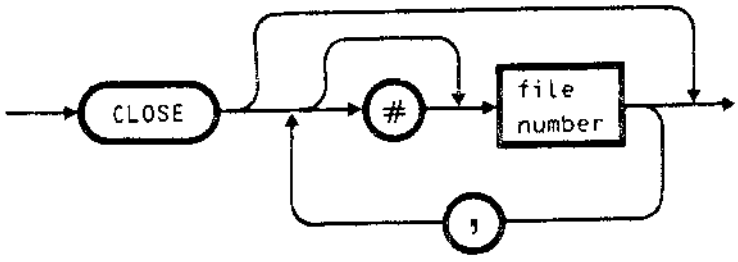
Each instruction is described in the following way:

- Function.
- Format: describes the way in which to call the instruction.
- Diagnostics: describes the replies given by the instruction.

- Characteristics: describes the working of the system during the execution of the instruction and gives the necessary indications for its correct use.

- Example: an example call for an instruction.

This instruction closes the I/O operations on a line.



where:

file number is a numeric expression, whose value represents the number associated to the line in the relative OPEN.

The CLOSE instruction, used without parameters, closes all the open files and lines.

Characteristics

1. When a CLOSE instruction is executed the association between the line and the number is cancelled. This line can be reopened (associating the same or another number) by using an OPEN instruction within the same or a different program.

2. All the lines opened by a program are automatically closed in case of abort of the program.

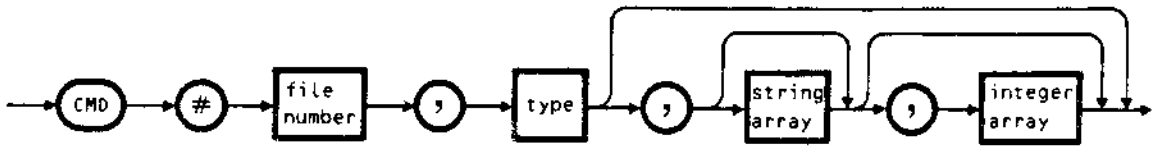
Note

When the END instruction or the NEW command are used all the lines and files which are open are automatically closed.

Example

```
100 CLOSE #2
.....
200 CLOSE #4
```

The CMD instruction (COMMAND) allows the state of the lines to be updated.



where:

file number is a numeric expression whose value represents the number associated to the line in the relative OPEN.

type is a numeric expression whose value indicates the type of parameters on which the user wants to intervene. The accepted values are:

```
0 = rxMode     (reception mode)
1 = txMode     (transmission mode)
2 = lineStatus (line states)
```

string array is a string vector for which the first element must be indicated. The meaning of the values assigned to this vector depend on the type of parameter on which the user has decided to intervene. If it is used, this vector must always be initialized. See Characteristic.

integer array is an integer vector of which the first element must be indicated. The meaning of the values assigned to this vector depend on the type of parameter on which the user has to intervene. If it is used, this vector must always be initialized. See Characteristic.

Diagnostics

The CMD instruction can give the following error message:

INVALID PARAMETERS - 110 = incorrect parameters have been given.

Characteristic

The parameters on which the user can intervene have the following meaning:

rxMode: The "string array" vector specifies the type of control which must be executed for each character received. The hexadecimal value of the character is used as an index for accessing the vector, and the control to be carried out is specified in the corresponding position. This vector allows the user to indicate which characters are not to be controlled, which are to be discarded, and which are to be considered match characters. These match characters place the driver in the 'end of input' state: in other words the driver closes the read function, after having taken a certain number of characters (max. 15) from the reception buffer. The accepted values are:

string array[0-255]	control executed
0 (hex = 30)	The corresponding character is not controlled
1 (hex = 31)	The corresponding character is ignored in output
from @ to 0 (from hex = 40 to hex = 4F included)	The corresponding character is a match character. Each read function is closed after a number of characters equal to the difference between (ord(stringarray[ord(char)])) and (ord(@)) has been received.

If the value ? (hex = 3F) is specified in the string array[0] parameter the table of controls to be executed remains unchanged.

The parameters "integer array[0]" and "integer array[1]" specify the time which the user is prepared to wait for the completion of a read function.

The parameter "integer array[0]" specifies the time which the user is prepared to wait for the arrival of the first character when the reception buffer is empty.

The parameter "integer array[1]" specifies the time which the user is prepared to wait for the arrival of each

successive character after the first.

The accepted values for these two parameters are those included in the interval from 0 to +32767. The time unit is equal to 100 msec. The value 32767 is interpreted as an infinite time. A negative value leaves the existing timeout unchanged.

The parameter "integer array[2]" specifies whether any character must be sent back to the transmitter (echo). This is only possible if the line is used in 'full duplex'. The accepted values are:

string array[X]	meaning
< 0	The value of the parameter remains unchanged
0	Disabled echo
X	Echo enabled using the 'X' line

If the user wants to enable the echo using the same line on which he is receiving, he must indicate the number of the PORT in the "integer array[2]" (which must be between 2 and 8) on which he is receiving (see the "OPEN" instruction, 'file name' parameter).

The parameter "integer array[3]" must always have the value 32767.

txMode: The parameter "integer array[0]" specifies how long the user is prepared to wait for the completion of each successive write function. The accepted values are included in the interval from 0 to 32767. The time unit is equal to 100 msec. The value 32767 is interpreted as an infinite time.

The parameter "string array" is not significant.

lineStatus: The user can pre-set the value of the output signals and the value of the break in output by intervening on the "string array" parameter. The accepted values are:

string array[X]	meaning
?	The value of the signal is left unchanged
0	The value of the signal is set to 0
1	The value of the signal is set set to 1

The parameters have the following meaning:

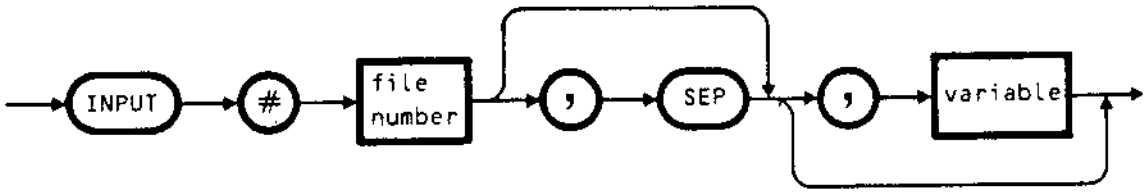
parameter	signal whose value is set up
string array[0]	break in output
string array[1]	RTS (Request To Send)
string array[2]	DTR (Data Terminal Ready)
string array[3]	output of the Peripheral Dependent

The parameter "integer array" is not significant.

Example

```
10 DIM Z$(255)
20 FOR I=1 TO 256
30 Z$(I)="0"
40 NEXT
50 DIM F%(4)
60 F%(1)=32767 : F%(2)=32767 : F%(3)=0 : F%(4)=32767
70 Z$(1)="?"
.......
100 OPEN "S",#1,"PORT1","000010133<<000111"
.......
150 CMD #1,0,Z$(1),F%(1)
.......
180 RBYTE #1,A%,1,B%,2,C%,2
```

This reads the data from the line and assigns them to the specified variables.



where:

file number is a numeric expression whose value represents the number associated to the line in the relative OPEN.

variable can be either a variable string or a variable numeric which must contain a datum read from the line.

If the INPUT instruction is called with the insertion of the SEP key word, the string is closed by the separator indicated using the type = 0 parameter of the CMD instruction. The separator is not placed in the variable, but can be found via the LTERM(0) function. If different separators have been used for more variables read with only one INPUT instruction, the LTERM(0) function only gives the last separator.

If the INPUT instruction is executed without the insertion of the key word SEP, the string is divided by the standard BASIC separators (comma, blank, etc.).

Diagnostics

The INPUT instruction can give one of the following error messages:

LINE ERROR - 112 = a line error has been found. The type of error can be discerned using the TEST instruction.

TIME OUT - 113 = the timeout has expired before the first character has been received or before the end of the instruction.

Notes

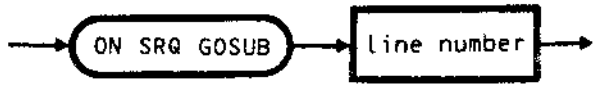
The LINE INPUT instruction can be used instead of the INPUT instruction.

See also the INPUT# instruction.

Example

```
70 INPUT #2,N1,N2,N3
80 PRINT "received = ";N1,N2,N3
.........
200 LINE INPUT #2,A$
.........
320 LINE INPUT #2,SEP,A$
```

This reads the state of the line reception buffers and, when the state changes from 'empty' to 'non empty', the control for execution of a program is passed to the line specified after GOSUB.



where:

line number is the number of the first line of the routine to be called.

Characteristics

1. The value of the LTERM(1) function is loaded, which is the result of the sum of the numbers of the PORTS whose buffers have changed from 'empty' to 'not empty'. The numbers of the PORTS are:

1	=	PORT1
2	=	PORT2
4	=	PORT3
8	=	PORT4
16	=	PORT5
32	=	PORT6
64	=	PORT7
128	=	PORT8

   For example, if the buffers in PORT7 and PORT8 have changed from 'empty' to 'not empty' the value of the LTERM(1) function is 192.

2. If the ON SRQ GOSUB 0 instruction is executed the control of the line reception buffers is disabled.

3. The execution of the routine called by GOSUB disables the reading of the state of the reception buffer.

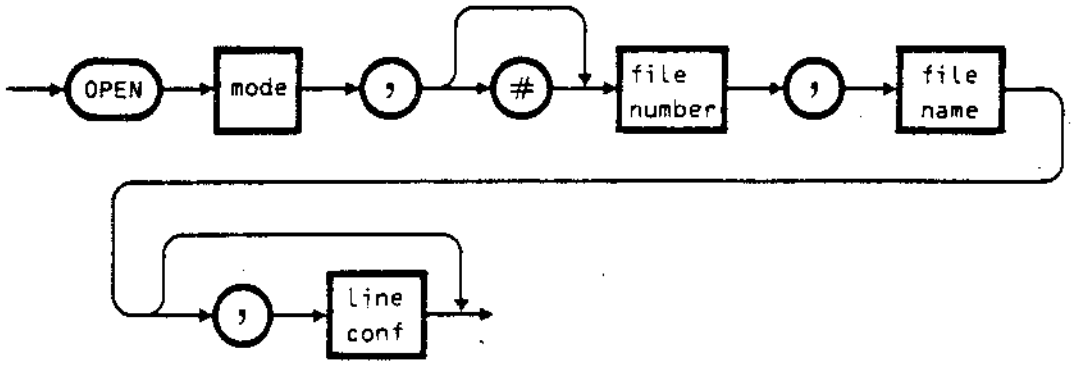
   The number of characters received from line can be read using TEST instruction. These characters are put in a buffer and the user can read them using the INPUT or RBYTE instructions.

   The RETURN instruction in the routine called by GOSUB re-enables the reading of the state of the reception buffer.

Example

```
50 OPEN "S",#3,"PORT2"
.......
100 ON SRQ GOSUB 250
.......
250 RBYTE #3,N$,20
260 PRINT "Values received are ",N$
```

This allows I/O operations to be carried out on a line.



where:

mode is a string expression whose first character must be "S".

file number is a numeric expression,the value must be between 1 and 15 inclusive. This number is associated to the line which is opened, and will be used to indicate that line in successive I/O operations.

file name is a string expression whose value indicates the line used. It must be between PORT1 and PORT113 inclusive. For the correspondence between lines and channels see the relative table in Chapter 2.

line conf is a string expression of 17 characters which allows the user to set the programming characteristics of a line. (For the accepted values and their meaning see the section "Programming Characteristics of the Line" in Chapter 2.)

Diagnostics

The OPEN instruction gives one of the following error messages:

INVALID PARAMETERS - 110 = incorrect parameters have been given.

LINE BUSY - 111 = the line which was to be opened is occupied.

Characteristics

1. If the protocol handles the break, a break during the reception phase is not considered to be an error.

   The protocol which handles the xon and xff characters and the protocol which handles the break can be used either for controlling the filling level of the reception buffer (protocol used in transmission), or for stopping and restarting a writing session (protocol used in reception).

   The protocol which handles the busy signal can be used for stopping and restarting a writing session. In the current line this protocol is implemented using the CTS signal.

2. The OPEN instruction gives the line controller, the output signals and the driver their initial values, as they are indicated in the configuration record of the driver, in the 'line conf' parameter, or by default.

   If the 'line conf' parameter is not specified, the line programming characteristics are assumed as indicated in the configuration record and by default.

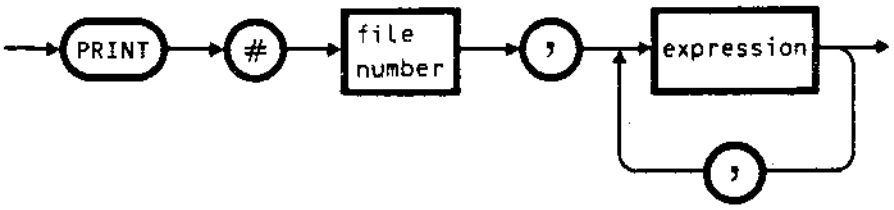
3. A line which has been opened by the OPEN instruction can only be used by one user at a time.

   He must then release the line with the CLOSE instruction.

Example

```
10 OPEN "S",2,"PORT1"
.............
50 OPEN "S",#5,"PORT5"
.............
90 OPEN "S",#3,"PORT2","001010033<<100111"
```

This sends data on line.



where:

file number is a numeric expression whose value represents the number associated to the line in the relative OPEN.

expression can be a constant numeric or string, or a variable numeric or string.

Diagnostics

The PRINT instruction gives one of the following error messages:

LINE ERROR - 112 = a line error has been found. The type of error can be discerned via the TEST instruction.

TIME OUT - 113 = the timeout has expired before the first character has been received or before the end of the instruction.

Notes

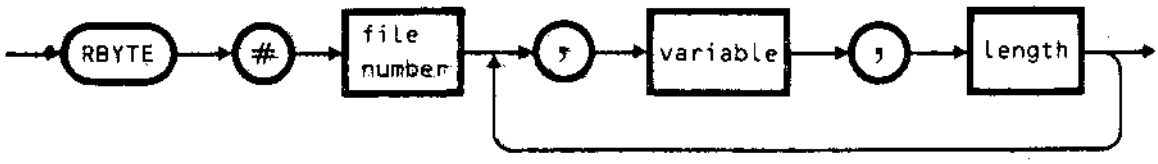
The PRINT USING or WRITE instructions can be used in place of the PRINT instruction.

See also the PRINT# instruction.

Example

```
70 PRINT #5,12,24,"COMPUTER"
........
100 WRITE #5,A$,B$
```

This reads the data from a line and assigns them to the specified variables.



where:

file number is a numeric expression whose value represents the number associated to the line in the relative OPEN.

variable is a variable which must contain the string or the numeric value read from line. The maximum number of bytes which can be read is 256.

length is a numeric expression which indicates the number of characters which must be read from line and stored in the variables which precede it. If numeric values are read, length must have a value of 1 or 2. In the latter case, for each pair of numeric values read from line, the first is considered the least significant, and its most significant bit is covered by the least significant bit of the second numeric value read.

Diagnostics

The RBYTE instruction can give one of the following error messages:

LINE ERROR - 112 = a line error has been found. The type of error can be discerned via the TEST instruction.

TIME OUT - 113 = the timeout has expired before the first character has been received or before the end of the instruction.

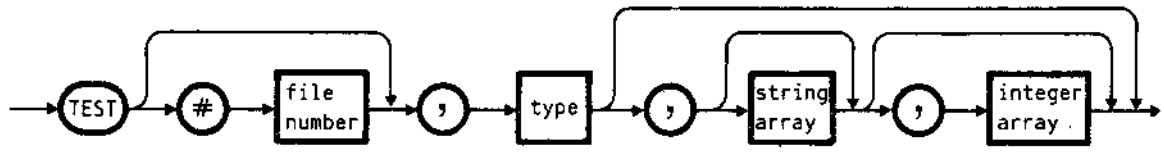
Note

The INPUT$ instruction can be used in place of the RBYTE instruction. INPUT$.

Example

```
70 RBYTE #2,N$,25,M$,20
80 PRINT "received = ";N$,M$
.........
100 A$ INPUT$(5,#2)
```

This function reads the state of the line.



where:

file number is a numeric expression whose value represents the number associated to the line in the relative OPEN. When "type 5" is used the "file number" must be ignored.

type is a numeric expression whose value indicates which state the user wants to read. The accepted values are:

0	=	rxMode	(reception mode)
1	=	txMode	(transmission mode)
2	=	lineStatus	(state of the 1/0 signals and the number of characters present in the reception buffer)
3	=	line conf	(programming characteristics of the line)
4	=	rxStatus	(reception state)
5	=	rxbuffsts	(status of the reception buffer)

string array is a string vector whose first element must be identified. The meaning of the values present in this string depends on which state the user has decided to read. If it is used, this vector must always be initialized. See Characteristic.

integer array is an integer vector whose first element must be identified. The meaning of the values present in this vector depends on which state the user has decided to read. If it is used, this vector must always be initialized. See Characteristic.

Diagnostics

The TEST instruction can give the following error message:

INVALID PARAMETERS - 110 = incorrect parameters have been given.

Characteristic

The meaning of the values present in the "string array" and "integer array" parameters is the following:

rxMode: The "integer array[0]" and "integer array[1]" parameters specify the time which the user is prepared to wait for the completion of a read function.

"integer array[0]" specifies the time which the user is prepared to wait for the arrival of the first character when the reception buffer is empty.

"integer array[1]" specifies the time which the user is prepared to wait for the arrival of each successive character after the first.

The possible values for these two parameters are those included in the interval from 0 to +32767. The time unit is equal to 100 msec. The value 32767 is interpreted as an infinite time.

The "integer arrary[2]" parameter indicates whether a received character has been sent back to the transmitter (enabled echo) or not, and which line has been used for this purpose. The possible values are:

intarr[2]	meaning
0	Disabled echo
X	Echo enabled using the 'X' line

The "integer array[3]" parameter has a value of 32767.

The "string array[0..255]" vector contains the current table of match characters, as indicated in the CMD instruction.

txMode: The "integer array[0]" parameter indicates the actual timeout used in transmission, that is the time the user is prepared to wait for the completion of each write function. The possible values are included in the interval from 0 to 32767. The time unit is equal to 100 msec. The value 32767 signifies an infinite time.

lineStatus: The "integer array[0]" parameter indicates the number of characters whose presence has been detected in the reception buffer.

The "string array" vector indicates the state of the I/O signals, each of which may have a value of 0 or 1. The meaning of the parameters is the following:

parameter	signal whose state is indicated
string array[0]	break in output
string array[1]	RTS (Request to Send)
string array[2]	DTR (Data Terminal Ready)
string array[3]	output of the Peripheral Dependent
string array[4]	break in input
string array[5]	busy (in the current use it is the same as for the CTS)
string array[6]	DSR (Data Set Ready)
string array[7]	CTS (Clear To Send)
string array[8]	DCD (Data Carrier Detector)

line conf: The parameters from "string array[0]" to "string array[16]" indicate the values which have been assigned to the line parameters. Their meaning is described in the section "Programming Characteristics of the Line" in Chapter 2.

The "integer array[0]" parameter indicates the size of the reception buffer in numbers of characters.

The "integer array[1]" parameter indicates the size of the transmission buffer in numbers of characters.

rxStatus: The "string array" vector gives indications on the state of the reception, and each of its parameters can have a value of 0 (OK) or 1 (if an anomaly has been found).

The parameters have the following meaning:

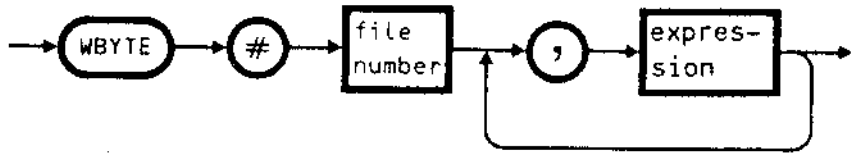
parameter	anomaly found (if = 1)
string array[0]	framing error
string array[1]	parity error
string array[2]	overrun
string array[3]	interruption (only in the case of protocol which does not handle the break)
string array[4]	line down
string array[5]	the data received exceed the size of the reception buffer

This type of anomaly prevents the correct execution of a read function, which will give the error code LINE ERROR.

The anomalies listed above are cleared when they have been detected by the TEST instruction.

rxbuffsts — the array "string array" can assume values between 1 and 129 inclusive. The first 113 bits inquires the status of the reception buffer of all the lines. Each character of "string array" indicates if the relative reception buffer is empty (hex = 00) or not (hex = 01).

This sends the integer values or the string expressions on line.



where:

file number is a numeric expression whose value represents the number associated to the line in the relative OPEN.

str variable is a variable, a constant integer, or a string expression which is to be sent on line.

Diagnostics

The WBYTE instruction can give one of the following error messages:

LINE ERROR - 112 = a line error has been found. The type of error can be discerned via the TEST instruction.

TIME OUT - 113 = the timeout has expired before the first character has been received or before the end of the instruction.

Example

```
10 DEFINT N
.......
60 N1=32
70 N2=281
80 WBYTE #2,N1,N2,"OLIVETTI"
```

A. INTERPRETED BASIC ERROR CODES

This appendix gives the error codes for the Interpreted BASIC instructions that handle the RS232/CL, followed by their meanings.

INVALID PARAMETERS code = 110

This is emitted by the OPEN, CMD, and TEST instructions when one or more invalid parameters have been given.

LINE BUSY code = 111

This is emitted by the OPEN instruction if the line which is to be opened is already busy.

LINE ERROR code 112

This is emitted by the WBYTE and PRINT instructions when there is a line down, or by the RBYTE and INPUT instructions when one of the following errors is found:

- line down
- the data received exceed the size of the reception buffer
- framing error
- parity error
- overrun
- break (only significant for a protocol that does not handle the break)

The type of error found can be read using the TEST instruction.

TIME OUT code = 113

This is emitted by the WBYTE and PRINT instructions when the transmission timeout expires before the whole of the instruction has been executed, or by the RBYTE and INPUT instructions when the reception timeout expires before the first character has been received, or before the instruction has been executed.

B. GLOSSARY

This appendix defines some of the terms which have been used in this manual and which are typical of the topic being discussed, illustrating their meaning in that context.

Match (match character) - A character is considered to be a match character when the user has defined it as such, using the "WRITEINFOLINE" function or the "CMD" instruction. Through the corresponding match table, it indicates the maximum number of characters which can be read after it has been received.

Match (match mode) - The read activity is carried out through the interpretation of the match characters.

Protocol - A set of syntactical and semantic rules which control communication. The following protocols can be used:

- The protocol which handles the xon-xoff characters.
- The protocol which handles the break.
- The protocol which handles the busy signal.

Session - A set of operations which begins with the opening of a line and finishes with its closure.

Timeout - An interval of time which the user is prepared to wait for an event. It prevents infinite waiting periods.

Transparent (transparent mode) - The read activity is carried out without the interpretation of eventual match characters, and is ended after a certain number of characters has been received (decided by the user) or when the timeout expires.

C. STANDARDS PUBLICATIONS

EIA RS232 STANDARD

This standard defines the conventions regarding an interface between Data Terminal Equipment (DTE) and Data Circuit-terminating Equipment (DCE) which uses a serial exchange of binary data (August 1969).

Published by : ELECTRONIC INDUSTRIES ASSOCIATION
Engineering Department
2001 Eye Street, NW
Washington, DC 20006

CCITT V.24

This standard is entitled:

Recommendation V.24
LIST OF DEFINITIONS FOR EXCHANGE CIRCUITS BETWEEN DATA TERMINAL EQUIPMENT AND DATA CIRCUIT-TERMINATING EQUIPMENT

and forms part of Volume Viii.1 (i.e. Part 1 of Volume VIII) of:

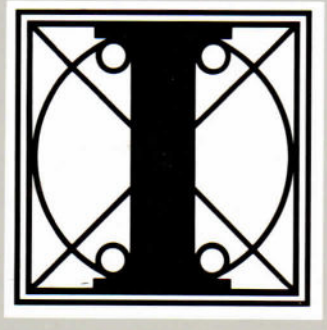
CCITT SIXTH PLENARY ASSEMBLY
Geneva, 27 September - 8 October 1976
VOLUME VIII.1

DATA TRANSMISSION VIA THE TELEPHONIC NETWORK, in 1977

Published by :

INTERNATIONAL TELECOMMUNICATIONS UNION
GENEVA, Switzerland

Code 4004450 H (3)
Printed in Italy



olivetti