

RCSL No: 991 10631

Edition: September 1986

Author: Carl Henrik Dreyer
Ole Thomasen

Title:

RC8000/IMC Port Handler
Reference Manual



Keywords:

RC8000, IMC, Local Area Network.

Abstract:

Reference Manual for the RC8000/IMC Port Handler process. The manual describes the user interface and the associated functions. It covers the specific aspects of using the RC8000/IMC Port Handler Process, and is meant to be used by RC8000 users operating within an IMC local area network.

(36 printed pages)

TABLE OF CONTENTS	PAGE
1. INTRODUCTION	1
2. THE PORT DEVICE	2
2.1 Synchronization of Messages and IMC Functions	2
2.2 Buffer Management	3
3. GENERAL MESSAGE INTERFACE	5
3.1 Message Format	5
3.2 Answer Format	5
4. OPERATIONS	9
4.1 Control Operations	9
4.1.1 Sense	9
4.1.2 Close port	10
4.1.3 Open port	10
4.1.4 Getconnection	12
4.1.5 Connect	12
4.1.6 Disconnect	13
4.1.7 Get credit	14
4.1.8 Reset	15
4.1.9 Get reset	16
4.1.10 Sense port	16
4.1.11 Initialize buffers	17
4.1.12 Define timer	18
4.2 Data Transfer Operations	19
4.2.1 Receive	19
4.2.2 Receiveall	20
4.2.3 Send	21
4.2.4 Receiveletter	23
4.2.5 Sendletter	24

TABLE OF CONTENTSPAGEAPPENDIX:

A. REFERENCES	26
B. MESSAGE/ANSWER FORMAT SURVEY	27

1. INTRODUCTION.

The external RC8000 process PORT implements a general, high-level access method to an IMC (ref.1) local area network.

The RC8000 is connected to the local area network by an Attached Device Processor (ADP), where the mapping between RC8000 operations and IMC network operations take place. The implementation of the external RC8000 process PORT is done within the ADP.

All external processes in RC8000 represent a device of some kind. The term device merely covers a logical unit with a well-described and (from a RC8000 point of view) a convenient behaviour, and need in general not cover a physical device.

The PORT device implements the characteristics of an IMC PORT, thus giving a 'direct network access' (or at least direct IMC-access) from RC8000 applications to the local area network.

As IMC is a common base for communication on the network (and independant of hardware and operating systems), the PORT device gives an excellent facility for enabling the RC8000 to communicate with other units in a local area network.

2. THE PORT DEVICE.

The basic concepts of IMC may be studied in Distributed Systems Architecture (DSA, ref.2). These include the port and connection concepts.

The PORT device can handle (at most) one IMC-PORT. All IMC functions may be called by sending messages to PORT.

The IMC system is implemented as an 'event-driven' system. A function call never stops/delays the calling process, but (in general) a buffer is handed over to IMC. After execution, the buffer is returned as an event (indicating the result/success of the function call).

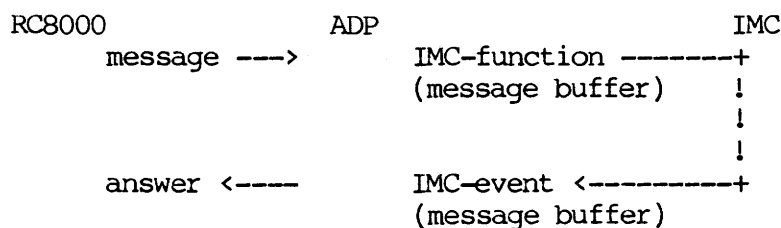
This scheme nicely conforms with the RC8000 message communication. A message (from an internal process) invokes an IMC-function call, and when the buffer is returned as an event, the message is answered.

Though the 'event-driven' communication scheme is usable, this is not the way standard RC8000 applications (e.g. utility programs) communicate. They use a 'synchronized' scheme, by sending one message and awaiting the answer.

The PORT device supports this scheme, and there is not a strict correlation between IMC-event-buffers and message/answers. The details are described in section 2.1.

2.1 Synchronization of Messages and IMC-Functions

The 'event-driven' scheme can be illustrated as followed:



This scheme is used strictly for the operations receive, receiveall, send, receiveletter and sendletter. The scheme is well suited for normal data transfer in a synchronized scheme, provided the other party of the communication agrees on the direction of transfer.

Operations concerning the IMC-PORT use a different scheme. The open port operation is answered without delay, even though an event buffer is supplied for IMC. This buffer is not returned until the port is closed (either deliberately or because of some incident). When this event occurs, PORT changes the state to closed. If the closeport operation was called (deliberately close), this message will pend, and not answered until the closed-event occurs, even though the IMC-function closeport does not itself require an event buffer.

A similar scheme is used for connection management. The IMC-functions connect and getconnection use two buffers. The first is returned when the connection is established, and PORT will use this for answering the corresponding message. The second buffer is returned when the connection has been removed, whether it is done deliberately (calling disconnect) or because of some incident or because the opposite connection end point has disconnected. In any case, the event will cause PORT to update the state of the connection. If a disconnect operation was the cause, this message will pend, and not be answered until the disconnect-event occurs.

2.2 Buffer Management

When the PORT device is created (at link creation from RC8000), it has no buffers available. After creation the user must specify the number and size of the buffers by sending a 'initialize buffer' message. The message may be sent several times and in that way enable the user to allocate more buffers if needed.

The number of buffers supplied defines the number of simultaneous messages, that PORT is capable of handling. This may have a side-effect, which one should be aware of. Suppose PORT has 5 buffers. If the application sends 5 input messages (for example receive operations), these will be pending

until some data arrives. Until then, PORT cannot handle any more messages (not even a closeport message), and eventual output will not penetrate to PORT. As a consequence of this, the number of buffers should be coordinated with the application.

The buffers have a fixed size (fixed for each port device). This gives a limit to the data blocks sent on IMC. Input messages (for example receive operation) may supply a larger buffer area (in RC8000), but will never receive more than the buffer size in PORT. On the other hand, output messages may specify a large data block, as the block will automatically be split into the needed buffers (when such an output is received in some distant end, each buffer will match one input message, so the buffers are not re-collected).

The PORT device supports a non-IMC facility. In connection based data transfer, (i.e. operations send, receive, receiveall), a single header-byte may be placed in front of the data block sent from RC8000. This byte is placed in the message, and not part of the data block from RC8000. PORT will put it as head of the buffer sent on IMC. Conversely, the header byte may be stripped off when received, and returned as part of the answer (to the input message). The purpose of this facility is to ease implementation of protocols (on top of IMC), using a single-byte header, as for example Server Access Protocol (ref.3).

3. GENERAL MESSAGE INTERFACE.

The PORT device receives messages from internal processes in RC8000, each message expressing an operation to be executed. The formats follow the normal conventions of RC8000.

Messages are accepted from any internal process which has initialized or reserved the external process representing the device.

3.1 Message Format

The detailed message formats are described in chapter 4 (where only the relevant fields are mentioned). As an example, the message for receive is:

```
+ 0 : 3<12 + mode
+ 2 : first RC8000 storage address
+ 4 : last RC8000 storage address
+ 6 : 0<8 + connection-index
+ 8 : irrelevant
+10 : irrelevant
+12 : irrelevant
+14 : irrelevant
```

Messages are grouped as control messages and I/O messages.

Connection index is always placed in the same field, if used. It is a positive integer in the range 1-255, but is also limited by the port parameter max-connections. Below, connection index is referred to as index.

The last four words are in some messages used for a name (of an IMC-PORT).

3.2 Answer Format

Each message is answered, as described for each operation in chapter 4 (where only the relevant fields are mentioned). An answer to a receive message has the format:

- + 0 : status word
- + 2 : halfwords (12 bit entities) transferred
- + 4 : bytes (8 bit entities) transferred
- + 6 : connection index
- + 8 : (start byte received, if mode 2 used)
- +10 : irrelevant
- +12 : connection state
- +14 : disconnect reason

The status word (24 bits) is a bit mask, indicating the success of the operation. All bits are zero if no incident is reported. The following status bits are possible from PORT (numbering bits 0-23, bit 0 being MSB):

bit 4: Blocklength. The data block received was bigger than the buffer area pointed out by the input message, or: The event from IMC indicated 'overflow', or both. As much as possible is transferred from the data block.

bit 5: End document. The connection index used is free (or: not connected). Used in I/O operations.

bit 9: Mode error. The state of the port or connection does not permit this call.

The connection index (or index) is the parameter from the message.

The connection state is the state at answer time, i.e. after the event triggering the answer. The coding is:

- 0 = free
- 1 = remote_accept (after getconnection)
- 2 = connecting (after connect)
- 3 = connected
- 4 = disconnecting (intermediate, after disconnect)

The disconnect reason is the value delivered with the disconnected event from IMC. It states the reason for the disconnecting. It is cleared at the subsequent connect/getconnection operation. The coding is:

0 = no reason (not after disconnect)
 1 = ok
 2 = name
 3 = resource
 4 = closed (i.e. port closed)
 5 = network

In some answers, the last two words contain port state and close reason, cf. section 4.1.1.

In case the buffer corresponding to the message is returned by IMC as a DUMMY-event, the message is answered with:

status word = 0
 halfwords = 0
 bytes = 0

This conforms with the 'message cleanup' described in section 2.2.

Result

Together with any answer, a result is delivered, stating the 'executability' of the operation. Only if the result indicates a normal answer, the answer buffer (above) is relevant, otherwise the answer is dummy.

Possible result values:

- 1: normal (answer relevant).
- 2: message rejected, as the (message-) sending process has not reserved or initialized the external process.
- 3: message unintelligible, as the contents does not conform with the formats described in chapter 4.
- 4: malfunction: the device is during reset, creation or removal, or a serious error has occurred.
- 5: the receiving (external) process does not exist.

The results indicating dummy answer may be returned on any operation, but the descriptions in chapter 4 assume a normal answer.

The answer to a message may be delayed, as described for each operation. In chapter 4, the following terms are used:

- the answer is not delayed: no event is awaited, as the answer is returned as soon as the operation (IMC-procedure) is executed.
- the answer is delayed: the answer awaits some event, which is supposed to happen after a short time, for example after a cleanup.
- the answer is pending: the answer awaits some event, which is triggered by some other entity in the network, so it may be pending infinitely.

4. OPERATIONS.

The IMC operations are described in two groups:

- control operations (section 4.1)
- input/output operations (section 4.2).

4.1 Control Operations

These operations are used for port control (openport, closeport, sense port and initialize buffers) and connection control (connect, getconnection, disconnect, getcredit, reset, get-reset and define timer).

4.1.1 Sense

The operation has the following format:

Message format

- + 0: 0<12 + 0
- + 6: 0<8 + index (or 0)

Answer format

- + 0: 0
- + 6: = (message + 6)
- +12: connection state/port state
- +14: disconnect reason/close reason

The message is answered immediately, and does not change any state. If the index is = 0, port state and reason is delivered in the answer. Otherwise connection state and reason is delivered.

The port state is encoded as follows:

0 = closed

1 = open

2 = closing (intermediate, after closeport)

A close reason is delivered with the portclosed-event from IMC. It states the reason for the closing. It is cleared at the subsequent openport operation. The coding is:

0 = no reason (not after closeport)

1 = ok

2 = name

3 = resource

4.1.2 Closeport

The operation has the following format:

Message format

+ 0: 2<12 + 0

Answer format

+ 0: 0

+12: port state

+14: close reason

If port is closed already, the answer is delivered immediately. Otherwise, the IMC procedure closeport is called, and the answer is delayed until the closed-event occurs.

4.1.3 Open port

The operation has the following format:

Message format

+ 0: 4<12 + 0
 + 2: 0<8 + scope
 + 4: 0<8 + rec_all
 + 6: 0<8 + max_connections
 + 8: portname (1)
 +10: portname (2)
 +12: portname (3)
 +14: portname (4)

Answer format

+ 0: status
 +12: port state
 +14: close reason

Possible status bits

bit 9: Port not closed, or no buffers available

Parameters

All parameters correspond to the parameters of the IMC procedure openport:

scope : 0 = anonymous
 1 = local
 2 = regional
 3 = global

rec_all : If <> 0, the receiveall operation is allowed, otherwise not.

max_connections: The number of connection end-points requested. Value 1-255. Later use of connection index is restricted by this maximum.

portname : Text of max 12 characters. If less, it must be terminated by a NULL character. The port is opened with this name.

The IMC-procedure openport is called, with the parameters as indicated in the message.

4.1.4 Getconnection

The operation has the following format:

Message format

+ 0: $6 < 12 + 0$
+ 6: $0 < 8 + \text{index}$

Answer format

+ 0: status
+ 6: index
+12: connection state
+14: disconnect reason

Possible status bits

bit 9: Connection index not free, port not open or index out of range.

The IMC procedure getconnection is called. When the remote-connect event occurs, the answer is returned.

4.1.5 Connect

The operation has the following format:

Message format

+ 0: $8 < 12 + 0$
+ 2: $0 < 8 + \text{service}$
+ 6: $0 < 8 + \text{index}$

+ 8: portname (1)
 +10: portname (2)
 +12: portname (3)
 +14: portname (4)

Answer format

+ 0: status
 + 6: index
 +12: connection state
 +14: disconnect reason

Possible status bits

bit 9: Connection index not free, port not open or index out of range.

Parameters

All parameters correspond to the parameters of the IMC procedure connect.

Service: Indicates the requested priority of operations on this connection.

0 = normal service.

1 = high priority

Portname: Text of max. 12 characters. If less, it must be terminated by a NULL character. The connection is established to a remote port of this name.

The IMC procedure connect is called, with parameters from the message. The answer is delayed until the local-connect event occurs.

4.1.6 Disconnect

The operation has the following format:

Message format

+ 0: 10<12 + 0
 + 6: 0<8 + index

Answer format

+ 0: status
 + 6: index
 +12: connection state
 +14: disconnect reason

Possible status bits

bit 9: Index out of range, port not open or previous disconnect not terminated

If the connection state is free, the answer is returned immediately, otherwise the IMC procedure disconnect is called, and the answer is delayed until the disconnected event occurs.

4.1.7 Get credit

The operation has the following format:

Message format

+ 0: 12<12 + 0
 + 6: 0<8 + index

Answer format

+ 0: status
 + 6: index
 + 8: credit count
 +12: connection state

+14: disconnect reason

Possible status bits

bit 9: Index out of range, port not open or not connected

The IMC procedure get-credit is called. The request is delayed as long as no credit exist. When one or more credit exist, the answer is returned with the pending number of credits.

4.1.8 Reset

The operation has the following format:

Message format

+ 0: 14<12 + 0
+ 6: 0<8 + index

Answer format

+ 0: status
+ 6: index
+12: connection state
+14: disconnect reason

Possible status bits

bit 9: Index out of range, port not open or not connected

The connection is reset, i.e. pending buffers at the local and remote end of the connection are released. When the resetcompletion event arrives, the answer is returned.

4.1.9 Get reset

The operation has the following format:

Message format

+ 0: 16<12 + 0
+ 6: 0<8 + index

Answer format

+ 0: status
+ 6: index
+ 8: credit count
+12: connection state
+14: disconnect reason

Possible status bits

bit 9: Index out of range, port not open or not connected

The IMC procedure get-reset is called. When the remote resetindication event arrives, the answer is returned with the pending number of credits.

4.1.10 Sense Port

The operation has the following format:

Message format

+ 0: 18<12 + 0

Answer format

+ 0: status
+ 6: maxconnections (IMC)

+ 8: maxlettersize (IMC)
 +12: port state
 +14: close reason

Possible status bits

none.

The IMC procedure maxconnections and maxlettersize are called, and the answer is returned immediately.

4.1.11 Initialize buffers

The operation has the following format:

Message format

+ 0: $20 < 12 + 0$
 + 6: $0 < 16 + \text{number of buffers}$
 + 8: $0 < 16 + \text{size of buffers}$

Answer format

+ 0: status
 + 6: $0 < 16 + \text{number of buffers}$
 + 8: $0 < 16 + \text{size of buffers}$

Possible status bits

none.

Buffers with the specified size will be allocated for the port.

If a buffer exists, additional buffers are created if the size of the requested buffers is the same as the already created ones. The buffers created will be the requested number minus the existing number of buffers.

In the answer the total number of buffers, and the size, is returned.

Before a port can be opened buffers must be allocated.

4.1.12 Define timer

The operation has the following format:

Message format

+ 0: 22<12 + 0
+ 6: timeout value in seconds

Answer format

+ 0: status

Possible status bits

none.

The timer is initialized by the operation. If an output operation is not completed within the stated number of seconds, the connection is reset and the output messages are returned with bytes transferred=0 and status=0.

The maximum value for the timer is 1200 seconds, and if the requested value is above 1200 seconds, the value 1200 will be used.

4.2 Data Transfer Operations

These operations are used for data transfer, and may be connection based (receive, receiveall, send) and connection-less (receiveletter, sendletter).

4.2.1 Receive

The operation has the following format:

Message format

- + 0: 3<12 + leading<7 + mode
- + 2: first address
- + 4: last address
- + 6: 0<8 + index

Answer format

- + 0: status
- + 2: halfwords transferred
- + 4: bytes transferred
- + 6: index
- + 8: header byte received (if mode = 2)
- +12: connection state
- +14: disconnect reason

Parameters

leading: Byte offset in the RC8000 buffer where the received data is loaded. Possible values are 0,1,2,3,4,5. If leading > 5 the value 5 will be used.

mode: Two modes are possible. Mode = 0 is transparent, i.e. the data block received is delivered. If mode = 2, the first byte of the data block received is delivered as start byte in the answer, and the rest is delivered in the buffer area.

Possible status bits

- bit 9: Index out of range, port closed
- bit 5: Connection state is not 'connected'
- bit 4: The data block received could not be contained in the buffer area pointed out by the message or the data block is returned from IMC as data-overflow event.

The IMC procedure receive is called, and the answer is pending until the corresponding buffer is returned (the event being data-arrived, data-overflow, or dummy-arrived). If mode is 2, the header byte is returned in the answer (and not in the buffer area).

The bytes transferred will contain the number of bytes written to the buffer area, including the leading bytes.

4.2.2 Receiveall

The operation has the following format:

Message format

- + 0: 3<12 + 1<2 + mode
- + 2: first address
- + 4: last address

Answer format

- + 0: status
- + 2: halfwords transferred
- + 4: bytes transferred
- + 6: index
- + 8: header byte received (if mode = 2)
- +12: connection state
- +14: disconnect reason

Parameters

mode: Same interpretation as in receive (cf. Section 4.2.1), i.e. values 0 or 2 possible.

Possible status bits

bit 9: Port closed, receiveall not allowed.

bit 4: The data block received could not be contained in the buffer area pointed out by the message, or the data block is returned from IMC as data-overflow event.

If the receiveall feature was not requested in the openport operation, the answer is returned with status indication. Otherwise, the IMC procedure receiveall is called, and the answer is pending until the corresponding buffer is returned (the event being data-arrived, data-overflow or dummy-arrived).

If mode is 2, the header byte is returned in the answer (and not in the buffer area).

Bytes transferred will contain the exact number of bytes written to the buffer area.

4.2.3 Send

The operation has the following format:

Message format

- + 0: 5<12 + leading<7 + trail<4 + mode
- + 2: first address
- + 4: last address
- + 6: 0<8 + index
- + 8: header byte to send (if mode = 2)

Answer format

+ 0: status
 + 2: halfwords transferred
 + 4: bytes transferred
 + 6: index
 +12: connection state
 +14: disconnect reason

Parameters

leading: A number of bytes in the beginning of the buffer that is not transferred. Possible values are 0,1,2,3,4,5. If leading > 5 the value 5 will be used.

trail: A number of bytes in the end of the buffer that is not transferred. Possible values are 0,1,2,3,4,5. If trail > 5 the value 5 will be used.

mode: Three modes are possible.

Mode = 0: Transparent mode i.e. the data block contained in the buffer area is sent.

Mode = 2: The header byte (in message) is placed in front of the data block, before sending.

Mode = 4: The header byte (in message) is modified by PORT before it is placed in front of the data block. That is in order to enable the remote end to assemble a number of received blocks originally sent as one block. In an output message where data is sent in several blocks PORT will clear bit 7 (MSB) in the header of all blocks sent, except the last one, where bit 7 will be set to one.

header byte: See mode above.

Possible status bits

bit 9: Index out of range, port closed.

bit 5: Connection state is not 'connected'.

If mode is 2 or 4, the header byte (in message) is placed in front of the data block, pointed out by the message.

The IMC procedure send is called, and the answer is pending until the corresponding buffer is returned (the event being data-sent or dummy-sent).

If the data block pointed out by the message exceeds the size of the buffers used by the PORT device, it will be split in the necessary data blocks, each one sent by the send procedure (possibly extending each one by the header byte).

Regardless of leading, trail and mode, halfwords and bytes transferred will contain the size of the buffer area (if ok status).

4.2.4 Receiveletter

The operation has the following format:

Message format

+ 0: 11<12 + 0
+ 2: first address
+ 4: last address

Answer format

+ 0: status
+ 2: halfwords transferred
+ 4: bytes transferred
+12: port state
+14: close reason

Possible status bits

bit 9: Port closed.

bit 4: The data block received could not be contained in the buffer area pointed out by the message.

The IMC procedure receiveletter is called, and the answer is pending until the corresponding buffer is returned (the event being letter-arrived or dummy-letter).

4.2.5 Sendletter

The operation has the following format:

Message format

+ 0: 13<12 + 0
 + 2: first address
 + 4: last address
 + 8: portname (1)
 +10: portname (2)
 +12: portname (3)
 +14: portname (4)

Answer format

+ 0: 0
 + 2: halfwords transferred
 + 4: bytes transferred
 +12: port state
 +14: disconnect reason

Parameters

Port name : Text of max. 12 characters. If less, it must be terminated by a NULL character. The letter is addressed to a port of this name.

This operation does not require that the port state is open. The port state in the answer is purely informative. The IMC procedure sendletter is called and the answer is delayed until the buffer is returned. Note that there is no delivery guarantee on letters.

A. REFERENCES

1. RCSL 42-i1983
DSA Inter Module Communication
Functional Description
2. RCSL 42-i1982
Distributed Systems Architecture
Report
3. RCSL 42-i2504
An IMC-based Server Access Protocol (SAP)
Report

<u>Operation</u>	<u>Message</u>	<u>Answer</u>
Sense	0<12+0 irrelevant irrelevant 0<8+index	0 irrelevant irrelevant 0<8+index irrelevant irrelevant conn/port state disc/close reason
Closeport	2<12+0	0 irrelevant irrelevant irrelevant irrelevant irrelevant port state close reason
Openport	4<12+0 0<8+scope 0<8+rec_all 0<8+max_conn portname (1) portname (2) portname (3) portname (4)	status irrelevant irrelevant irrelevant irrelevant irrelevant port state close reason
Getconnection	6<12+0 irrelevant irrelevant 0<8+index	status irrelevant irrelevant index irrelevant irrelevant conn state disc reason
Connect	8<12+0 0<8+service irrelevant 0<8+index portname (1) portname (2) portname (3) portname (4)	status irrelevant irrelevant index irrelevant irrelevant conn state disc reason

<u>Operation</u>	<u>Message</u>	<u>Answer</u>
Disconnect	10<12+0 irrelevant irrelevant 0<8+index	status irrelevant irrelevant index irrelevant irrelevant conn state disc reason
Get Credit	12<12+0 irrelevant irrelevant 0<8+index	status irrelevant irrelevant index credit count irrelevant conn state disc reason
Reset	14<12+0 irrelevant irrelevant 0<8+index	status irrelevant irrelevant index irrelevant irrelevant conn state disc reason
Get reset	16<12+0 irrelevant irrelevant 0<8+index	status irrelevant irrelevant index irrelevant irrelevant conn state disc reason
Sense port	18<12+0	status irrelevant irrelevant maxconnections (IMC) maxlettersize (IMC) irrelevant port state close reason
Initialize buffers	20<12+0 irrelevant irrelevant number of buffers size of buffers	status irrelevant irrelevant number of buffers size of buffers

<u>Operation</u>	<u>Message</u>	<u>Answer</u>
Define timer	22<12 + 0 irrelevant irrelevant timeout value in seconds	status
Receive	3<12+leading<7+mode first address last address 0<8+index	status halfwords bytes index headerbyte irrelevant conn state disc reason
Receiveall	3<12+1<2+mode first address last address	status halfwords bytes index headerbyte irrelevant conn state disc reason
Send	5<12+leading<7+trail<4+mode first address last address 0<8+index headerbyte	status halfwords bytes index irrelevant irrelevant conn state disc reason
Receiveletter	11<12+0 first address last address	status halfwords bytes irrelevant irrelevant irrelevant port state close reason
Sendletter	13>12+0 first address last address irrelevant portname (1) portname (2) portname (3) portname (4)	0 halfwords bytes irrelevant irrelevant port state close reason

RETURN LETTER

Title: RC8000/IMC Port.Handler

RCSL No.: 991 10631

A/S Regnecentralen af 1979/RC Computer A/S maintains a continual effort to improve the quality and usefulness of its publications. To do this effectively we need user feedback, your critical evaluation of this manual.

Please comment on this manual's completeness, accuracy, organization, usability, and readability:

Do you find errors in this manual? If so, specify by page.

How can this manual be improved?

Other comments?

Name: _____ Title: _____

Company: _____

Address: _____

Date: _____

Thank you

RCSL Nr. 46-F 0093

..... Fold here

..... Do not tear - Fold here and staple

Affix
postage
here

E REGNECENTRALEN
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark