

Tall med Carl Henrik Drey 87.03.30
Finnis 11

John Jensen
R-parkering gruppe

RCSL No: 42-11983

Edition: June 1982

Author: Jens Hammerum
Ejvind Lynning
Tom Skovgaard
Ole Thomasen

Title:

DSA Inter Module Communication
Functional Description



Keywords:

Distributed Systems, Local Networks, Inter Module Communication.

Abstract:

DSA/IMC is the standardized Inter Module Communication (IMC) method developed by RC for use in distributed systems. The IMC service constitutes a key element of the RC Distributed System Architecture (DSA) described in ref. [1].

This manual describes in detail the interface functions and the service provided by the DSA/IMC as it appears independent of actual implementation environments.

This manual is intended as the basis for system design and also as an introduction to DSA/IMC.

(50 printed pages)

Copyright © 1982, A/S Regnecentralen af 1979

RC Computer A/S

Printed by A/S Regnecentralen af 1979, Copenhagen

Users of this manual are cautioned that the specifications contained herein are subject to change by RC at any time without prior notice. RC is not responsible for typographical or arithmetic errors which may appear in this manual and shall not be responsible for any damages caused by reliance on any of the materials presented.

TABLE OF CONTENTS

PAGE

1. INTRODUCTION	1
2. BASIC IMC CONCEPTS	3
2.1 Connection Based Communication	5
2.2 Flow Control Between Residents	7
3. IMC FUNCTIONS	12
3.1 OPENPORT	14
3.2 CLOSEPORT	16
3.3 GETCONNECTION	17
3.4 CONNECT	18
3.5 DISCONNECT	19
3.6 SENDLETTER	20
3.7 RECEIVELETTER	22
3.8 SEND	22
3.9 RECEIVE	23
3.10 RECEIVEALL	24
3.11 GETCREDIT	25
3.12 RESET	25
3.13 GETRESET	27
4. IMC EVENTS	28
4.1 Remote Connect	28
4.2 Local Connect	28
4.3 Disconnected	29
4.4 Letter Arrived	30
4.5 Letter Sent	30
4.6 Data Arrived	31
4.7 Data Overrun	31
4.8 Data Sent	32
4.9 Credit	32
4.10 Reset Indication	33
4.11 Reset Completion	33
4.12 Port Closed	34
4.13 Dummy Event	34

TABLE OF CONTENTS (continued)PAGE

5. SPECIAL IMPLEMENTATION FEATURES	36
5.1 Local IMC Services	36
5.2 IMC Functional Subsets	36
5.3 Different Media	37

APPENDICES:

A. REFERENCES	39
B. IMC FUNCTION SURVEY	40
C. IMC EVENT SURVEY	42
D. CORRELATION OF IMC FUNCTIONS TO EVENTS	43

1. INTRODUCTION

1.

This manual contains a detailed description of the Inter Module Communication (IMC) service which is the backbone of distributed systems developed by RC based upon local area networks. The general architectural concepts of distributed systems (DSA) are discussed in ref. [1].

Since the term module is a heavily used term with many interpretations, the alternative term 'resident module' or simply 'resident' is used throughout this manual. This term refers to the fact that a module (in the sense of DSA) resides at one PU.

The communication between resident modules in a distributed system (ref. [1]) is based upon a set of standard functions constituting the IMC service. By means of these functions, the residents in the distributed system communicate in a uniform way even though they may be implemented in different programming languages and may run in different runtime system environments.

To the individual resident, the IMC service is represented as a set of functions implemented as system calls or predefined language routines. By calling these functions - and only in that way - the resident is able to communicate with other residents in a distributed system.

This manual presents the IMC functions independent of implementation languages and runtime systems. The exact description of the IMC functions as they appear in a given language/runtime environment will be found in a manual relating to the environment in question, cf. ref. [2] and ref. [3].

Taken as a whole, all the hardware, including the interconnection media, and software which in a distributed system supports the IMC functionality is called the IMC network, or just the IMC.

The present manual is relevant in a project design phase, when the resident module interfaces and intercommunication are being specified, and when questions of implementation languages etc. are irrelevant.

Further, with respect to the documentation of the inter module communication in a given system, this manual may be useful as a basis or as a reference.

Finally, the present manual may serve as an introduction to the concepts of the RC standardized IAC service.

2. BASIC IMC CONCEPTS

2.

In this chapter the basic functional concepts of the Inter Module Communication are introduced.

The IMC is intended for use in distributed systems based upon local networks. A general (but not concise) description of the characteristics of the interconnection structure includes:

- high transfer rates
- low error rates
- a large number of stations (processors)
- a limited geographical area

Obviously this description is only a vague indication of the target systems. However, low error rates should be stressed as an important characteristic, since (local) networks based upon media not having this quality normally have a totally different kind of user interface from what IMC offers, oriented towards global network services.

The IMC service is defined to the user as a set of functions or primitives. This set of functions provides basically two types of communication service, namely datagram communication and virtual circuit communication. In the following the latter will be referred to as connection based communication.

A resident gains access to IMC services via a port. Thus ports represent application entities attached to the IMC network. Port names, in turn, establish the identification of ports towards residents and within the IMC network.

Ports may be opened and closed by residents at any time by means of the IMC primitives OPENPORT and CLOSEPORT. A resident may open several ports.

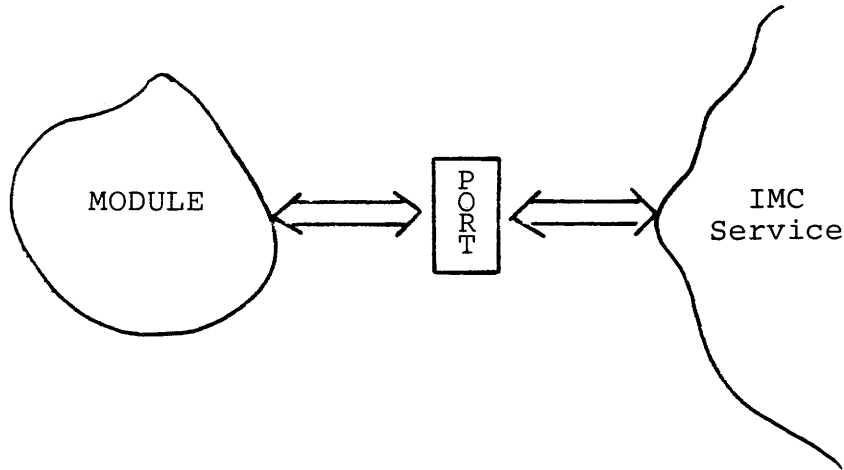


Figure 1: A port.

Even without a port a resident may send datagrams addressed to other residents' ports by means of the primitive `SENDLETTER`. At a port received datagrams are delivered to the owner resident in response to a call of `RECEIVELETTER`.

A resident may establish connections from an open port to other ports. Thus a port may be the end-point of a number of connections to other ports (fig. 5). Each connection has precisely two end-points.

A connection serves as a full duplex virtual circuit between the two residents owning the ports which are the end-points for the connection. On a connection the IMC secures correct sequencing and flow control.

A connection may be established by either of the two involved residents by a call of `CONNECT` and removed again (by either of the two modules) by a call of `DISCONNECT`. Each new call of `CONNECT` from a resident will establish a new connection. Establishment of a connection is possible only if enabled by the addressed resident by a call of `GETCONNECTION`, allowing an incoming connect request to be accepted.

Between two ports a number of (parallel) connections may exist.

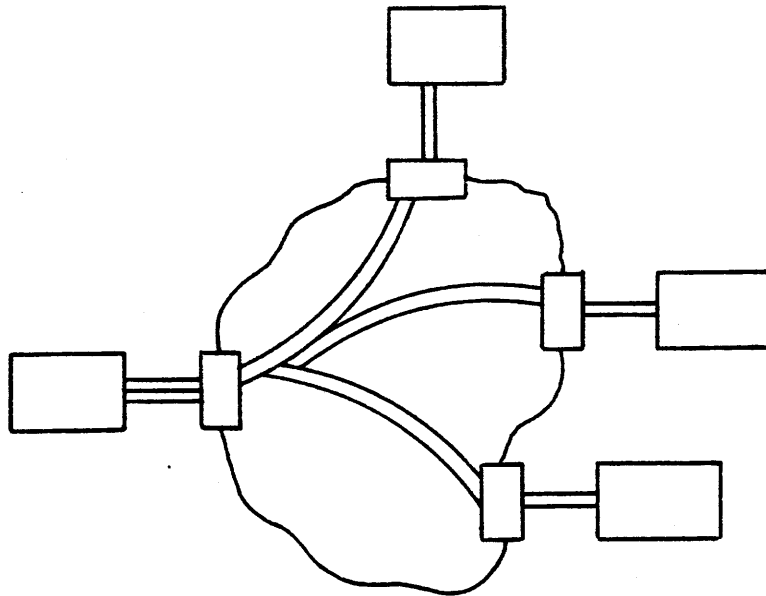


Figure 2: Modules, ports and connections.

On a connection the two involved residents communicate by sending and receiving data. This is accomplished by means of the IMC primitives SEND and RECEIVE or RECEIVEALL.

Flow control between residents communicating on a connection is possible due to the IMC primitive GETCREDIT, and outstanding data buffers on a connection may be retrieved by means of RESET.

By using an appropriate subset of the IMC primitives, the resident modules define their own communication scheme according to the specific requirements of the application.

2.1 Connection Based Communication

2.1

In most cases the required type of inter module communication is connection based communication.

On a connection, data transfers takes place in an end-to-end synchronized way.

The IMC network uses flow-control internally (on connections), so a data block entered at one end of a connection (by a call of SEND), will not be transferred to the other end of the connection before a receive resource has been made available there (by a call of RECEIVE or RECEIVEALL).

Thus it is impossible to create a congestion situation in the IMC network or at the receiver end (possibly both ends) of a connection, since no data will be transmitted before the receive resource is ready.

On a connection, an end-to-end synchronization between the involved modules is possible.

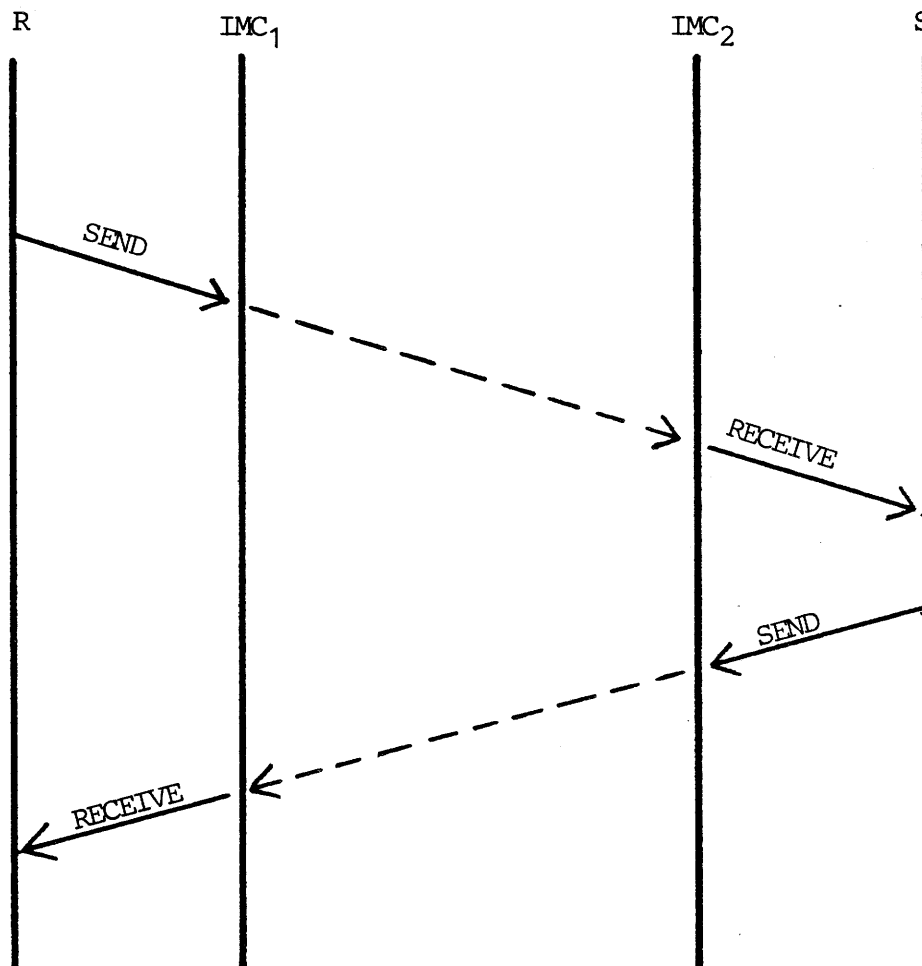


Figure 3: Synchronization on a connection.

Often some user protocol will be implemented on top of the IMC connection. If for instance a server resident S and a requester resident R synchronize their communication (before the data transfer phase), it might look like fig. 3.

The communication on a connection may be of any kind. It may be one-way data from a producer to a consumer or it may be dialog-oriented or any type of session-oriented communication.

After establishment, the addressing on a connection is direct, i.e. no port-names are involved. In the call of CONNECT the destination port is addressed and the local connection end-point is identified by a connection-identifier supplied by the calling resident. Thereafter this connection-identifier is used in all actions involving the connection.

In the call of GETCONNECTION the calling resident, similarly, identifies the connection end-point, at which it wishes to receive an incoming call.

2.2 Flow Control Between Residents

2.2

By means of the IMC primitive GETCREDIT it is possible for residents communicating on a connection to obtain flow-control from resident to resident.

The GETCREDIT call enables a credit event to be generated as an answer to the calling resident, as soon as a receive resource has been delivered by the remote resident by a call of RECEIVE. This is shown in fig. 4. If a receive resource is available when the call is made the event occurs immediately.

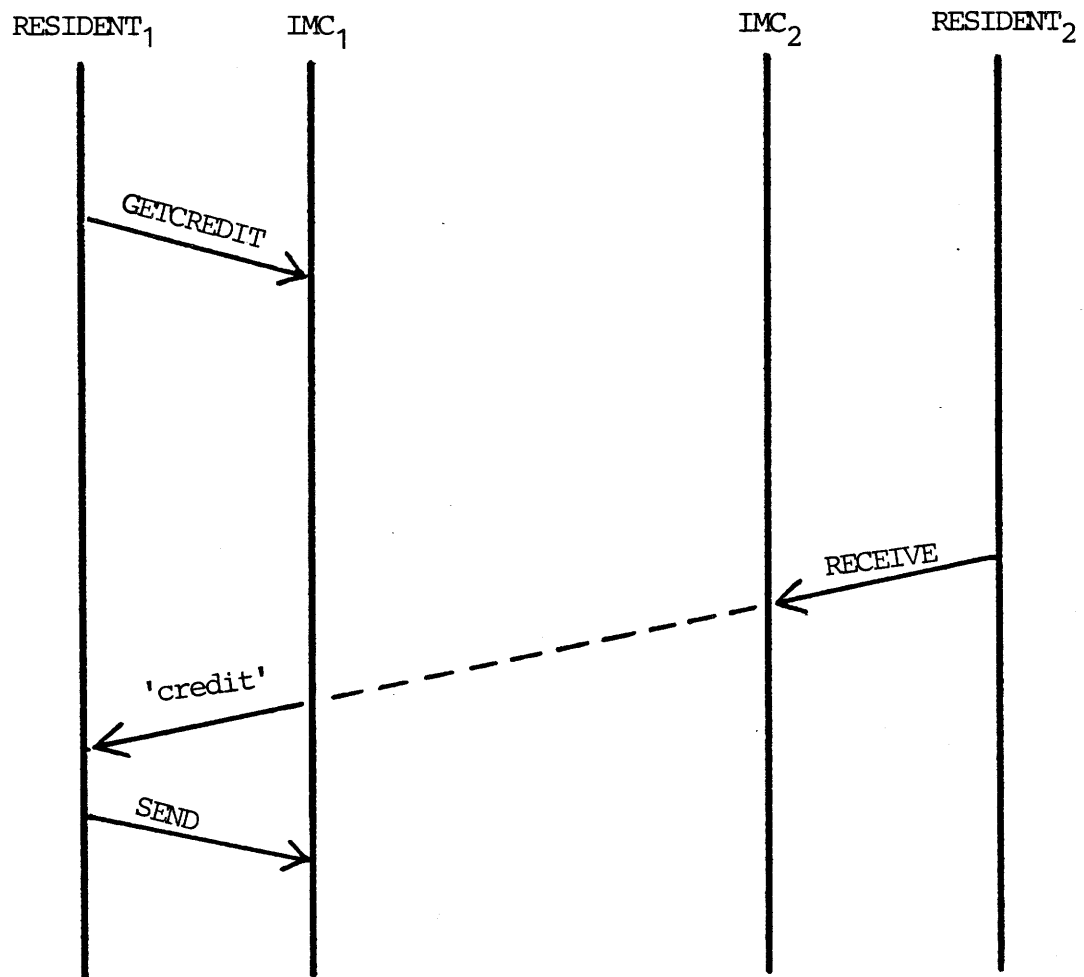


Figure 4: Credit event on a connection.

The communication scheme involving GETCREDIT is specifically useful in cases where the sender has a limited number of buffers and is able to control the production of data to be transmitted.

An example is a system with one-to-many communication, i.e. one sender resident producing data for a number of consumer residents (see fig. 5).

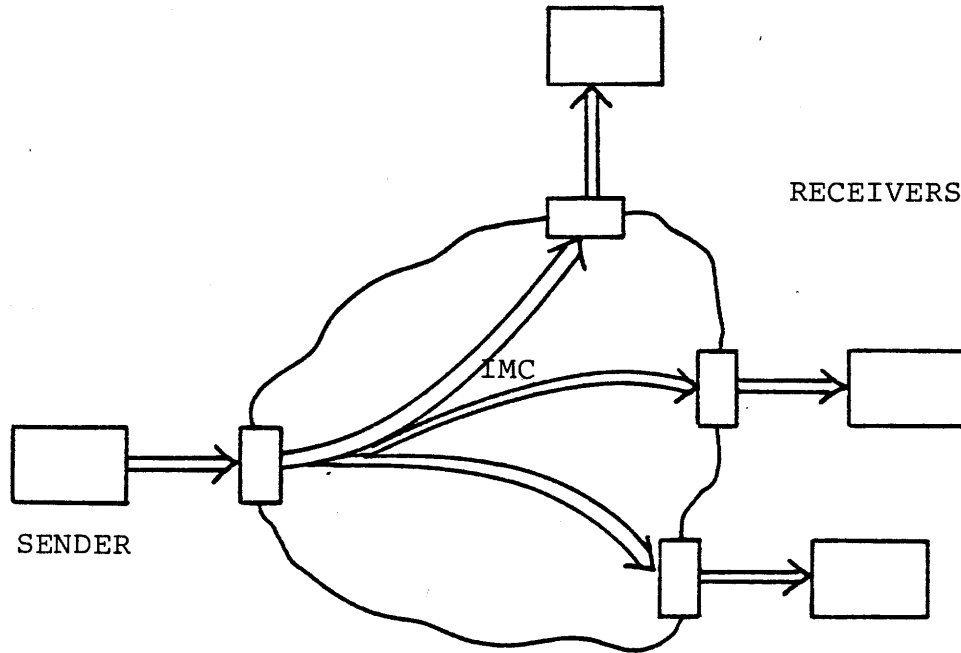


Figure 5: One-to-many communication.

In this case the sender resident might call GETCREDIT for each connection and only produce or allow for the production of a data block to be transmitted when a credit event occurs on the connection in question.

On the other hand, if the communication is many-to-one, i.e. a number of residents sending data to one consumer resident, the GETCREDIT primitive is of no use. (See fig. 6).

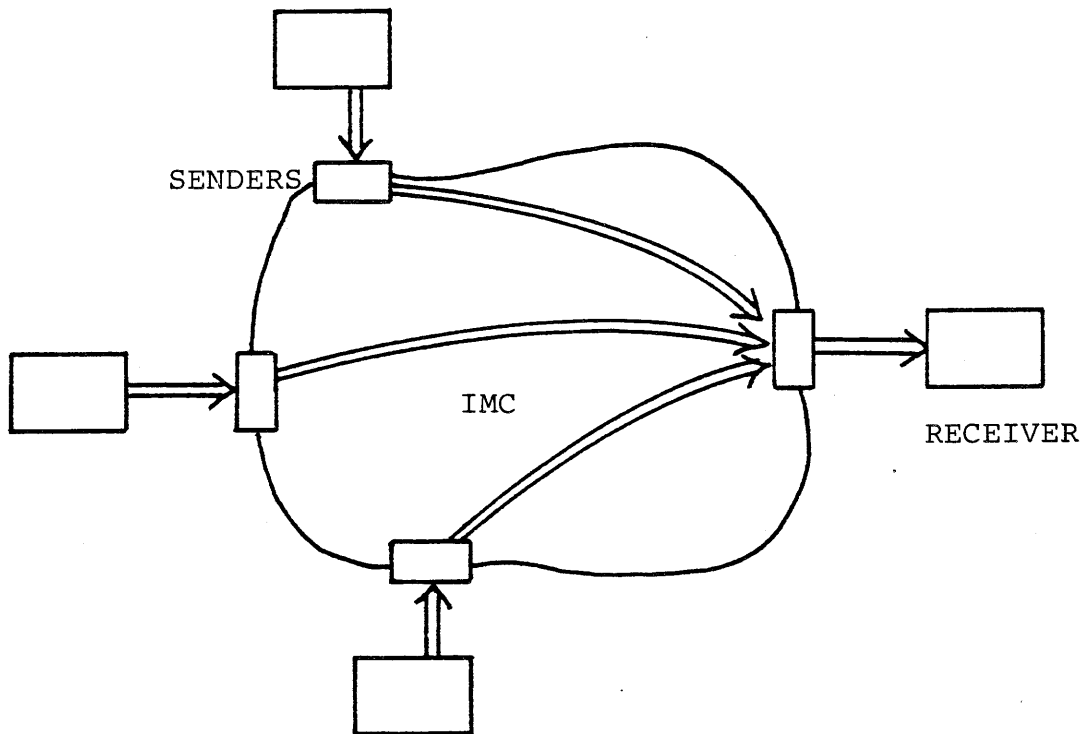


Figure 6: Many-to-one communication.

In this case the consumer will possibly not have a receive buffer for every connection. However, by calling `RECEIVEALL`, the consumer resident provides a receive resource ready to receive data on the first connection for which a call of `SEND` is issued remotely.

Thus, `RECEIVEALL` enables a data block to be received from any one of the connections in a specified port.

In the `RECEIVEALL` answer - including the data block - the IMC will specify which connection actually produced the data block.

As illustrated by these examples, it should be possible by means of the basic IMC primitives to implement a simple inter-resident flow control and end-to-end synchronization in cases where the communicating residents do not have a user protocol on top of the IMC communication which includes these features.

In cases where a higher-level protocol is actually present and used by the communicating residents, some of the IMC primitives may be redundant and may even be omitted from the configuration of the actual IMC service.

3. IMC FUNCTIONS

3.

The description of the communication primitives of the IMC service given in this chapter is the reference description of the IMC service independent of language and implementation environments. Thus the description is not concerned with syntax nor with the manner in which immediate results from calls of IMC primitives are passed.

The processing units (PU's) providing the IMC service in a distributed system will be referred to as IMC nodes.

Each IMC node has its own implementation of the IMC service, maintaining the communication to other IMC nodes in the system on the basis of the internal IMC protocol, as well as administering the local open ports, connections, and the interface to the residents.

Each IMC node may host a number of residents, each known by one or more port names.

An important feature of an IMC network is the ability to handle transit traffic. In the following description of the IMC primitives - especially the communication primitives involving a connection - the diagrams and the description will concern the end-point nodes, ignoring the fact that a number of intermediate IMC nodes may actually carry the information from PU to PU in order to establish the link from end-point to end-point. This fact will not be of any functional significance at the end-point nodes.

Some of the IMC primitives described have local significance only and some have global significance. The description will focus on the request initiating the IMC function in question and on the events which some requests may cause both at the local and at the remote site. This is shown in fig. 7.

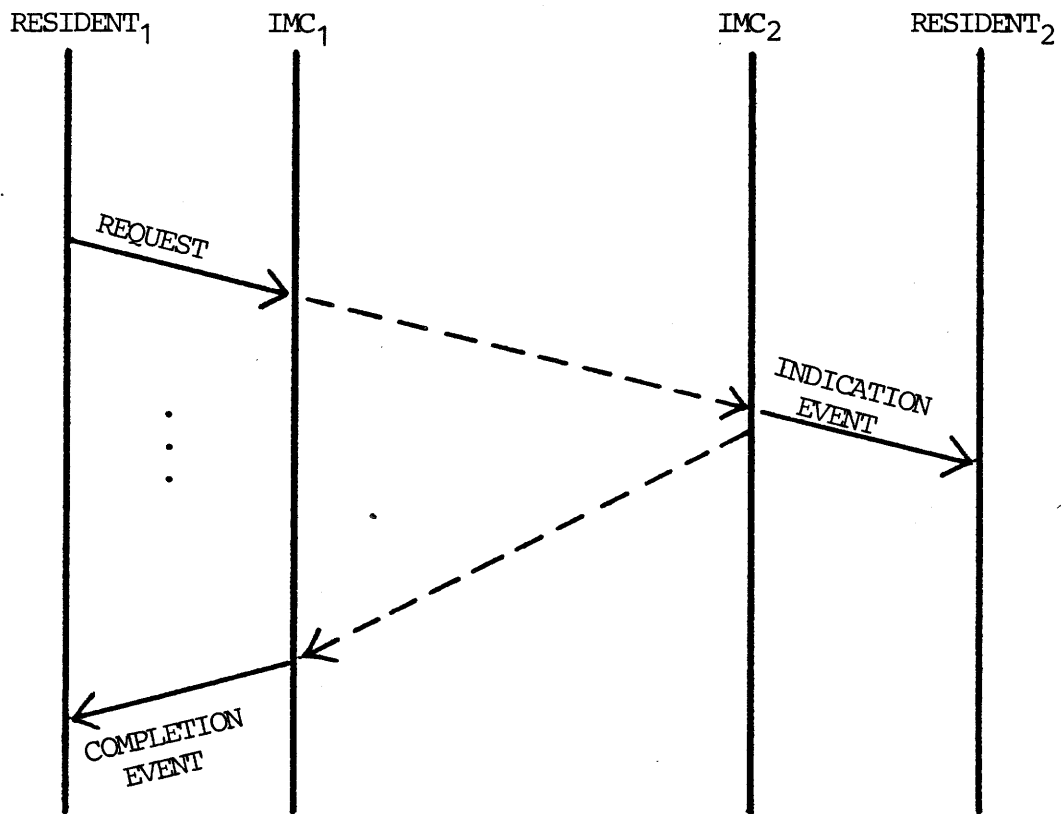


Figure 7: Request and events.

The actual implementation of the IMC service in a language/runtime system environment, will be of an asynchronous nature.

The resident calling an IMC function, i.e. issuing a request, will return immediately from the call, with a result code indicating whether the requested function is initiated or not, and if not why.

Messages and indications from the IMC network to a resident are delivered as events, enabled by the resident via IMC function calls.

When supported by the runtime environment, the delivery of events from the IMC to a requesting resident will be based upon standard inter process (task) communication facilities.

Thus, a resident enabling an IMC event in a call of an IMC function, will pass an inter-process-communication resource to the IMC service, which will return this resource to the resident when the event occurs.

The same strategy applies to data buffers. A further description of this interaction may be found in the appropriate implementation manual.

3.1 OPENPORT

3.1

A port is opened by a call of the IMC function OPENPORT.

OPENPORT

Parameters:

port	port object
name	port name
scope	scope of the port
no. of connections	
'port closed' resource	

Indication event:

none

Completion event:

'port closed' if a name conflict is detected or if a network resource problem occurs.

Hereby the calling resident identifies itself to the IMC system by a (logical) name.

If the parameters are accepted the new name is broadcast in the local network. However, during the broadcasting of a name a name conflict may be discovered, i.e. two ports have the same name. In this case the new port will be closed. A port may also be closed because of lack of port resources in the IMC network. In both cases the resident is notified by a 'port closed' event without having requested this event by a call of CLOSEPORT.

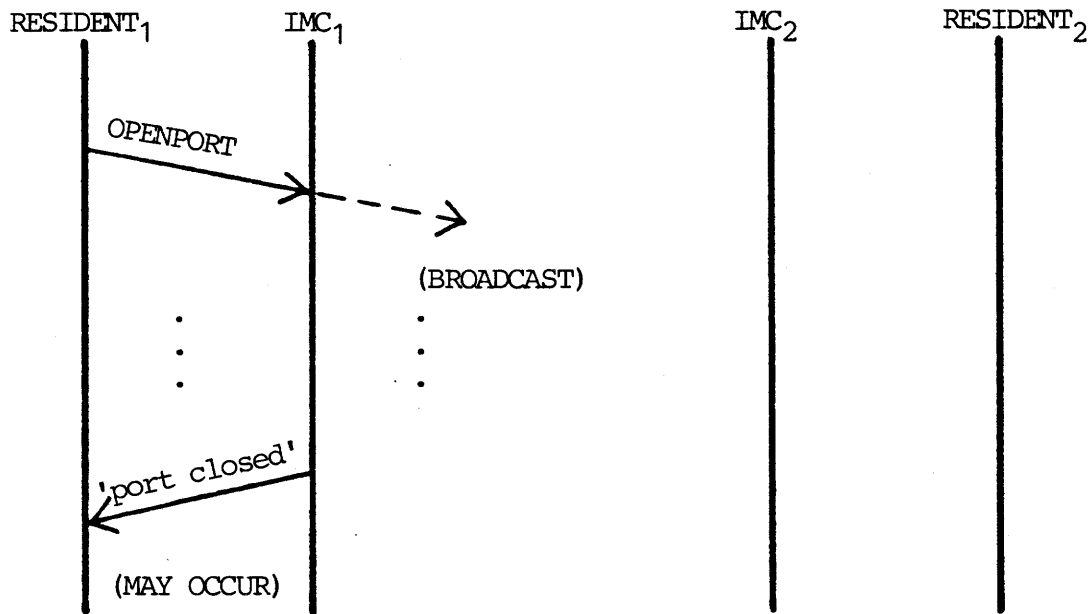


Figure 8: OPENPORT.

Concerning the naming strategy, the basic approach is that names must be unique. However, in order to reduce configuration information and avoid the administration of a global (centralized) name file, scope rules are introduced for port naming.

The scope conventions introduce a sort of hierarchical port naming, allowing port name broadcasting to be limited to a region or even a smaller subnet (see ref. [1]). The naming levels are:

- global,
- regional,
- local,
- anonymous.

According to the scope rules a name conflict occurs only if two ports are made known to an IMC node with the same name and the same scope. As an example consider a printing service, which may be installed at a number of cluster controllers throughout the network. The port names of the resulting server residents may be chosen alike by applying regional naming.

Regions are defined at IMC network configuration time.

Local names are known only in the subnet of the IMC node where the port resides.

Finally a port may be anonymous, having no name. This feature is relevant to (requesting) residents using the service of other residents, but not offering a service themselves. Anonymous ports can not receive lettergrams or connects (calls) from other residents.

Apart from the regional and local scope conventions, port names are purely logical, and no conventions have (yet) been established for the construction of port names.

The parameter no. of connections must hold the maximum number of connections required at the port.

3.2 CLOSEPORT

3.2

A port is closed by a call of the IMC function CLOSEPORT.

CLOSEPORT

Parameters:

port port object

Indication event:

none

Completion event:

'port closed'

The call causes the IMC network to remove existing connections within the port and return all pending resources attached to the port and the connections. The name of the port (and the knowledge of its existence) will be deleted from the entire IMC network.

In fig. 12 is shown a CLOSEPORT, where the port has one active connection. Resources of active connections are returned as the completion event 'dummy event', and the connection removal indicated by 'disconnected' events.

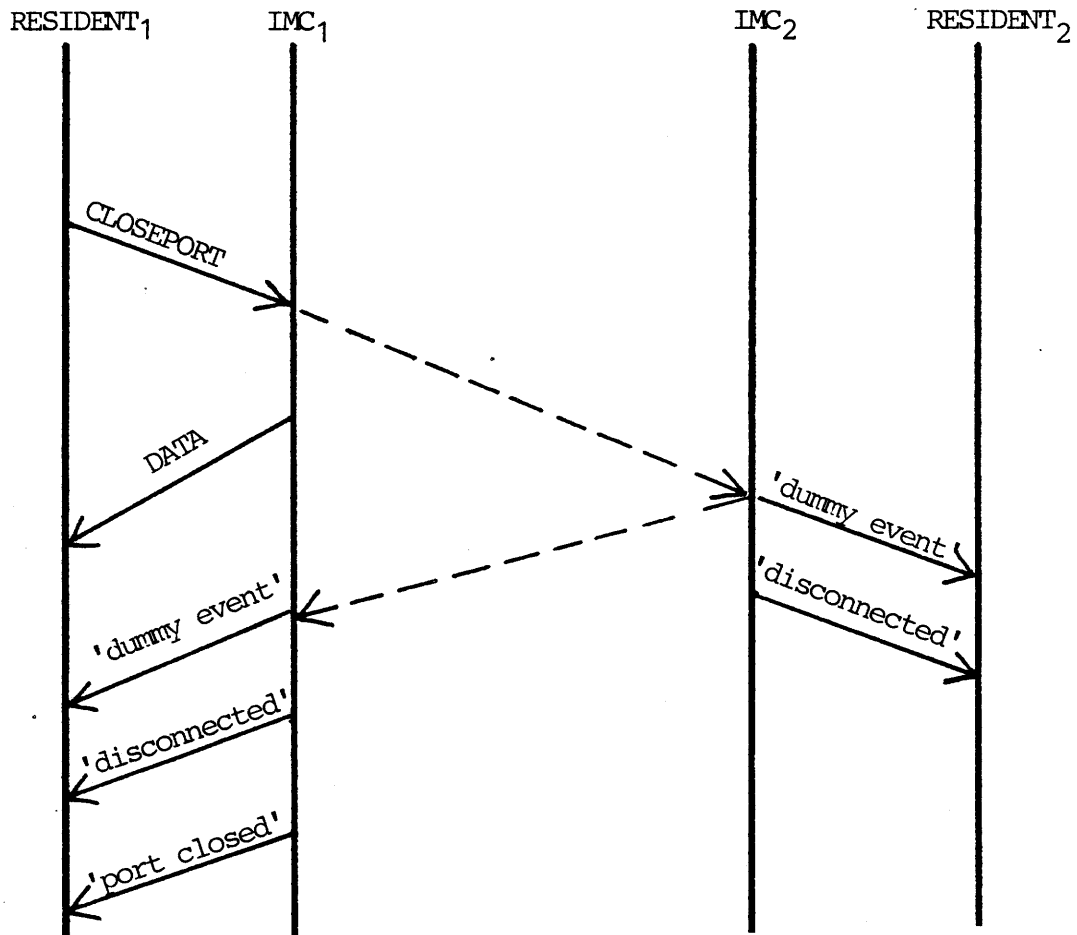


Figure 9: A port with one active connection is closed.

3.3 GETCONNECTION

3.3

A call of the IMC function GETCONNECTION enables a connection to be established to this port by a remote call of CONNECT.

GETCONNECTION

Parameters:

port	port object
connection	local connection identification (chosen by the resident)
'remote connect' resource	enables completion event
'disconnected' resource	

Completion event:

'dummy event'

'remote connect'

the connection is established

'disconnected'

contains a failure code

A call of GETCONNECTION allows a connection to be established from another port to this port.

The connection may later be disallowed by a call of DISCONNECT, in which case the 'remote connect' resource is returned as 'dummy event'.

In the section on CONNECT below, a successful establishment of a connection is shown.

3.4 CONNECT

3.4

A call of the IMC function CONNECT will initiate an attempt to establish a connection from the local port to another port.

CONNECT

Parameters:

port

port object

connection

local connection identification
(chosen by the resident)

name

destination port name

'local connect' resource

enables completion event

'disconnected' resource

Indication event:

'remote connect'

the connection is established

Completion event:

'local connect'

the connection is established
(optional)

'disconnected'

contains a failure code

A call of CONNECT will cause the local IMC node to transmit a connection request to the remote IMC node hosting the addressed destination port.

If resources for a new connection is present at both ends (and intermediate IMC nodes), a 'remote connect' indication event is generated at the remote end, and when the acknowledge arrives, a 'local connect' completion event is delivered to the initiating module.

The CONNECT request may be rejected locally or at other IMC nodes if there are no resources for a connection or if the destination port is unknown. These failures will cause the completion event 'disconnected' to be generated. A failure code will indicate whether the failure was caused by missing resources (somewhere) or the destination port being unknown.

Below is shown a successful CONNECT.

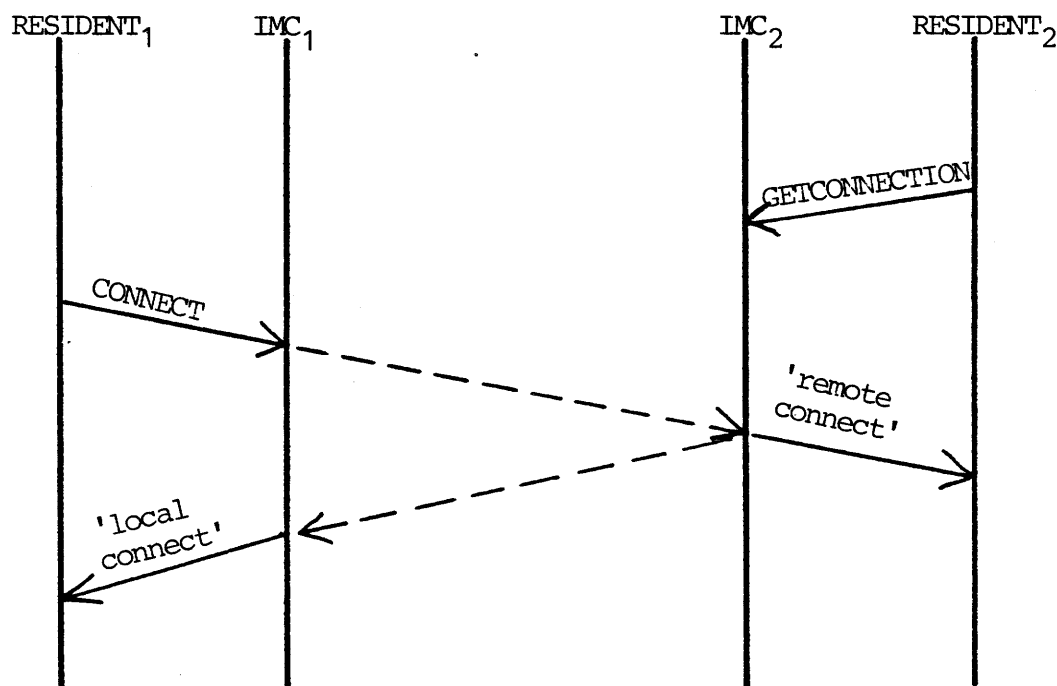


Figure 10: Successful CONNECT.

3.5 DISCONNECT

3.5

A call of the IMC function DISCONNECT will complete current data transfers and remove the specified connection.

DISCONNECT

Parameters:

port	port object
connection	local connection identification

Indication event:

'disconnected' the connection is removed

Completion event:

'disconnected' the connection is removed

The close down of a connection is graceful, in that all possible data transfers are carried out before the unused data resources are returned (as 'dummy event').

A 'disconnected' event is generated at both ends.

A DISCONNECT is shown below.

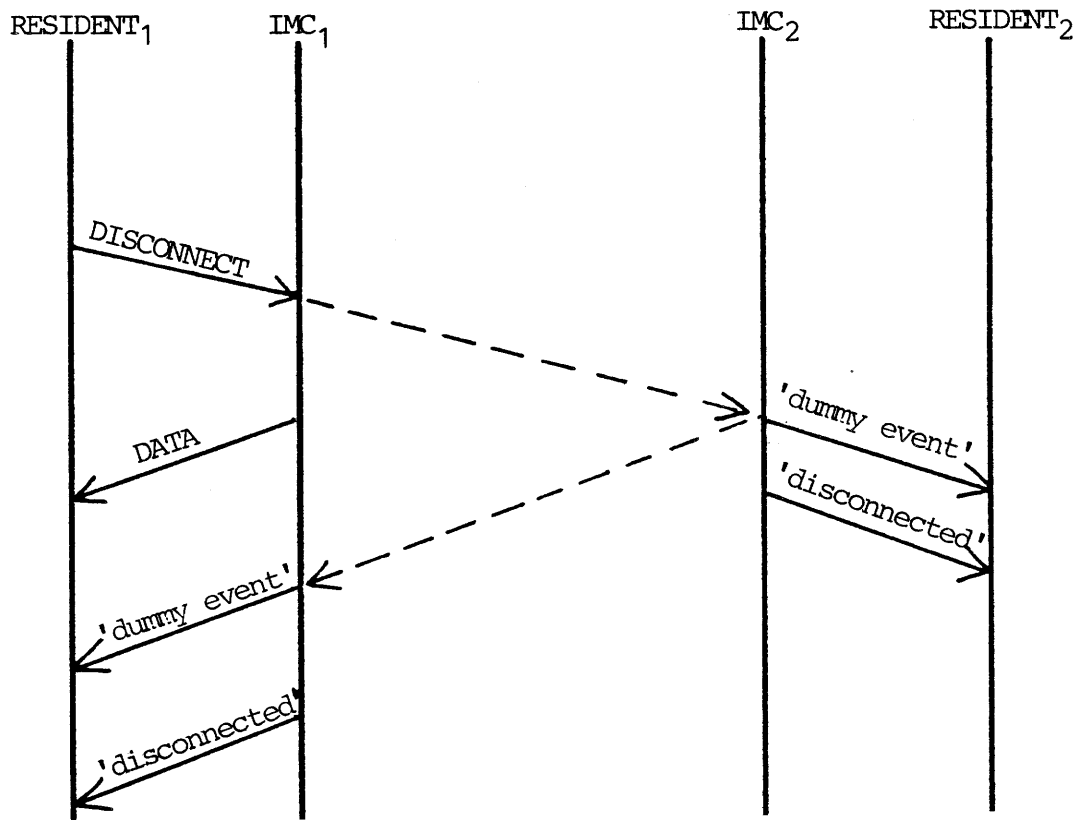


Figure 11: Disconnect a connection.

By a call of SENDLETTER a resident sends a data block to a port as a connectionless data transfer.

SENDLETTER

Parameters:

name destination port name
 data resource a data resource containing the data block

Indication event:

'letter arrived'

Completion event (with data resource):

'letter sent' data transport is done - implies no delivery guarantee.

After a call of SENDLETTER, the data transfer will happen immediately - not implying flow-control or end-to-end control. Thereafter a 'letter sent' event is delivered to the initiating module - implying no delivery guarantee.

If the receiving port has not enabled the receipt of the letter (by a call of RECEIVELETTER) it will be lost.

Below are shown two situations of letter transfer.

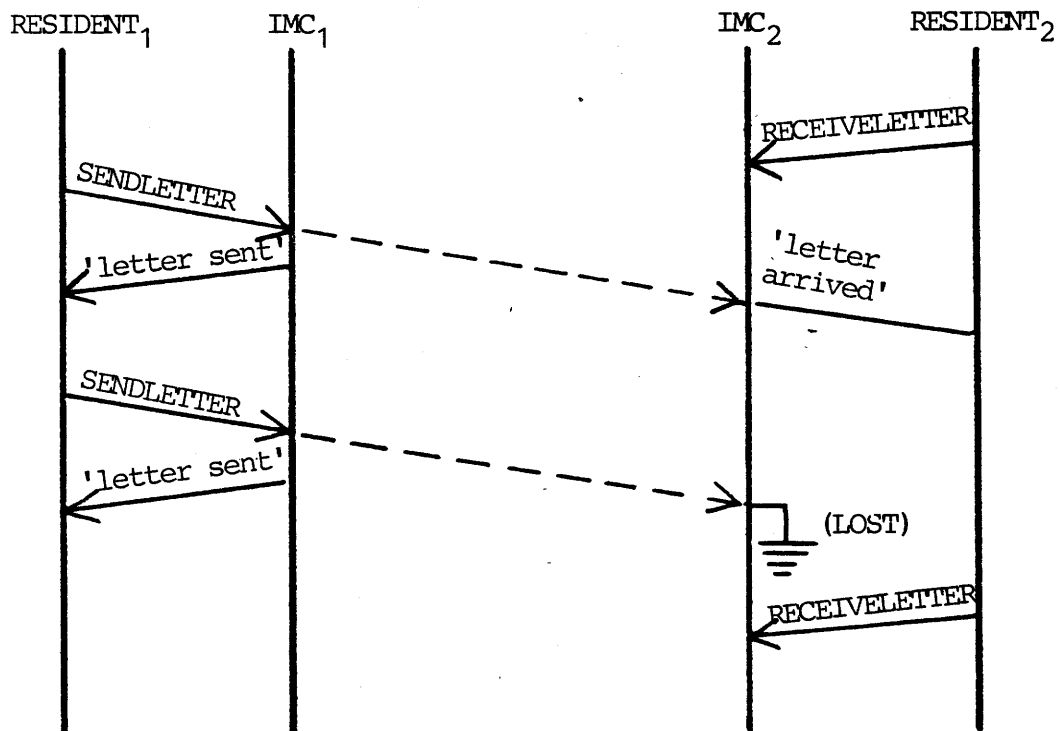


Figure 12: Letter transfer.

An implementation will give a maximum size for letters.

3.7 RECEIVELETTER

3.7

This IMC function has local significance only.

If a resident wants to receive datagrams (letters) sent to it from other residents, a data receive resource must be delivered to the IMC system by a call of RECEIVELETTER.

The receive resource will be returned to the resident only when a datagram arrives or when the port is closed.

RECEIVELETTER

Parameters:

port	port object
data resource	a data resource to hold a received letter

Indication event:

none

Completion event (in data resource):

'letter arrived'

'dummy event'

3.8 SEND

3.8

The IMC function SEND is the means for data transmission on a connection.

SEND

Call parameters:

port	port object
connection	local connection identification
data resource	a data resource containing the data block

Indication event:

'data arrived'

'data overrun'

Completion event (in data resource)

'data sent'

'dummy event'

The local IMC node maintains the necessary dialogue with the remote IMC node (on the implied connection) in such a way that the actual data transfer takes place only when the receiving resident has a receive resource ready for the data block.

If SEND is called and there are no receive resources ready at the remote end, the data block will be kept at the local IMC node until a receive resource has been given at the remote end.

Upon a local RESET, or DISCONNECT or CLOSEPORT at either end the data block will be released by the IMC service.

The fragmentation of large datablocks into small blocks, if necessary on a data link, and the corresponding reassembly will be handled by the IMC network.

Below is shown a SEND data transfer.

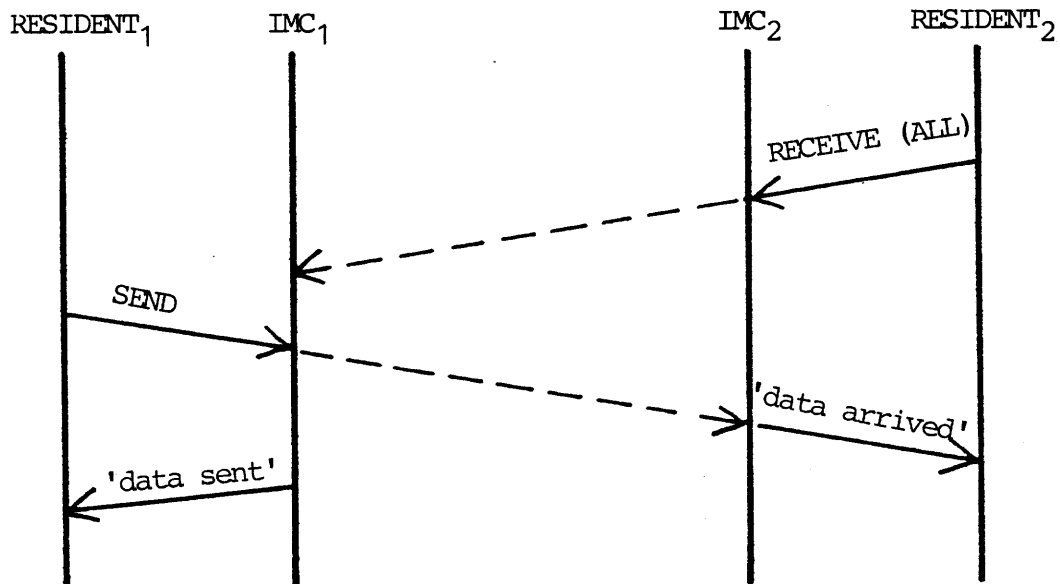


Figure 13: SEND and RECEIVE.

3.9 RECEIVE

3.9

By calling the IMC function RECEIVE, a resident delivers a free data resource for a data block received on the specified connection.

The data resource will be kept by IMC until a data block arrives and then delivered to the resident as a completion event.

RECEIVE

Parameters:

port	port object
connection	local connection identification
data resource	a data resource to hold a received data block

Indication event

'data sent'

'credit event'

Completion event

'data arrived'

'data overrun'

'dummy event'

The completion event 'dummy event' will occur upon a local RESET, or DISCONNECT or CLOSEPORT at either end.

A call of RECEIVE may cause a 'credit event' at the remote end. If a credit event resource is not present (remote), the current credit to send will be increased by one.

3.10 RECEIVEALL

3.10

A call of the IMC function RECEIVEALL enables the receipt of a data block from any of the connections on the port.

The completion event returns the data resource and identifies the connection from which the data block came.

RECEIVEALL

Parameters:

port	port object
data resource	a data resource to hold a received data block

Indication event:

'data sent'

Completion event:

'data arrived'

'data overrun'

'dummy event'

3.11 GETCREDIT

3.11

This IMC function has local significance only.

A call of GETCREDIT enables a 'credit event'. The event will occur when there is an outstanding receive resource at the remote end of the specified connection for which credit has not already been given, but only provided there is an outstanding 'reset indication' event resource at the local end-point.

GETCREDIT

Parameters:

port	port object
connection	local connection identification
'credit' resource	

Indication event:

none

Completion event:

'credit event'

'dummy event'

3.12 RESET

3.12

At a given point in time a resident may have a number of resources outstanding on a connection, resulting from calls of SEND or GETCREDIT and RECEIVE for which the completion events have not occurred. Such calls may be regretted and the associated resources forced back from the IMC by means of a call of RESET. The resources will be returned as 'dummy event'.

If RESET causes receive resources previously made available by calls of RECEIVE to be returned the remote end will lose credit.

In this case a 'reset indication' event will occur containing the number of lost credits.

At the local end-point RESET also causes the credit account to be cleared. Thus any previously received credit is no longer valid.

RESET

Parameters:

port port object
 connection local connection identification
 'reset completion' resource

Indication event:

'reset indication'

Completion event:

'reset completion'

'dummy event'

Below is shown a reset situation.

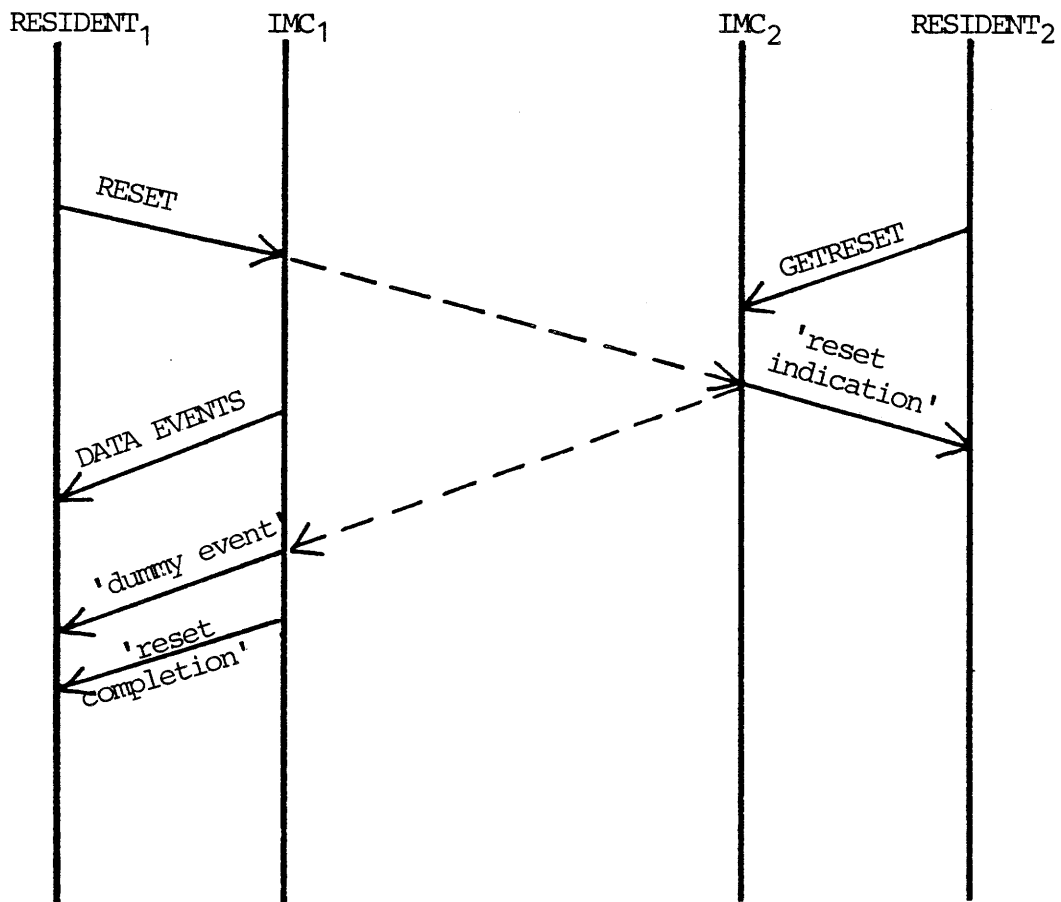


Figure 14: RESET on a connection.

This IMC function has local significance only. It enables a 'reset indication' event, cf. RESET.

GETRESET**Parameters:**

port	port object
connection	local connection identification
'reset indication' resource	

Indication event:

none

Completion event:

'reset indication'
'dummy event'

4. IMC EVENTS

4.

In this chapter the IMC events are described in detail. IMC events are caused by actions somewhere in the IMC network.

An event normally occurs as a result of a resident's call, local or remote, of an IMC function. In special circumstances events may occur because of failures in the IMC network.

The delivery of an event from the IMC to a resident takes place only when the resident has enabled that event. Events are enabled through IMC function calls by the transfer of a resource from the calling resident to the IMC. Later, the resource will be returned with relevant event information and possibly containing received data.

A diagram showing the relation between IMC functions and IMC events is found in appendix D.

4.1 Remote Connect

4.1

This event indicates the establishment of a new connection as a consequence of a remote call of CONNECT.

Indication of:

Remote call of CONNECT, successful

Enabled by:

GETCONNECTION

Parameter:

connection local connection identification

4.2 Local Connect

4.2

This event reports on the successful establishment of a connection, initiated by a local call of CONNECT.

Completion of:

A local call of CONNECT

Enabled by:

CONNECT

Parameter:

connection local connection identification

4.3. Disconnected

4.3

This event will occur at both ends of a connection, when it has been removed, or as a completion of an unsuccessful CONNECT.

Indication of:

Remote call of DISCONNECT, or connection removed because of a failure in the IMC network.

Completion of:

DISCONNECT or
CONNECT (unsuccessful)

Enabled by:

CONNECT
GETCONNECTION

Parameters:

connection local connection identification
reason diagnostics

Fig. 15 shows a disconnect situation caused by an internal IMC error.

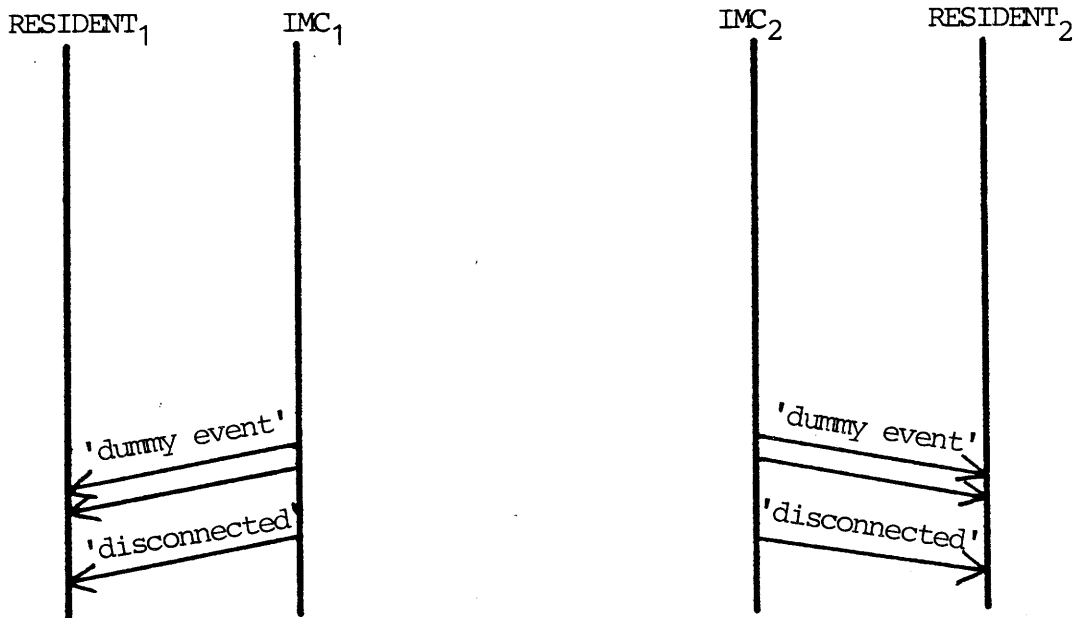


Figure 15: Disconnect caused by an internal IMC error.

4.4 Letter Arrived

4.4

This event occurs when a letter, sent from another resident, arrives and the receipt is enabled by a call of RECEIVELETTER.

Indication of:

SENDLETTER

Completion of and enabled by:

RECEIVELETTER

Parameter:

data data resource and data block description

4.5 Letter Sent

4.5

A completion of a call of SENDLETTER. Notice that this event implies no delivery guarantee.

Completion of and enabled by:

SENDLETTER

Parameter:

(data data resource returned)

4.6 Data Arrived

4.6

This event is a completion of a call of RECEIVE or RECEIVEALL.
The event delivers a data block transferred from the remote end
of an existing connection.

Data from a connection will be delivered in a "general resource"
(RECEIVEALL) only if no dedicated receive resource is waiting for
data at the connection.

Indication of:

SEND

Completion of and enabled by:

RECEIVE

RECEIVEALL

Parameters:

connection local connection identification

data data resource and data block description

4.7 Data Overrun

4.7

This event is delivered in the data resource of a call of RECEIVE
or RECEIVEALL.

A data block arrived on the connection in question, but had to be
truncated to fit into the data area of the data resource.

Indication of:

SEND

Completion of and enabled by:

RECEIVE

RECEIVEALL

Parameters:

connection	local connection identification
data	data resource and data block description

4.8 Data Sent

4.8

A completion and an end-to-end confirmation of a data transfer on a connection, initiated locally by a call of SEND.

(May be regarded as an implicit indication of RECEIVE or RECEIVEALL).

Completion of and enabled by:

SEND

Parameters:

connection	local connection identification
(data	data resource returned)

4.9 Credit

4.9

This event indicates that the remote end of a connection is ready to receive one or more data blocks.

The number of blocks for which credit has not been given previously is passed as a parameter.

Indication of:

RECEIVE

Completion of and enabled by:

GETCREDIT

Parameters:

connection	local connection identification
creditcount	allowed number of SEND transfers.

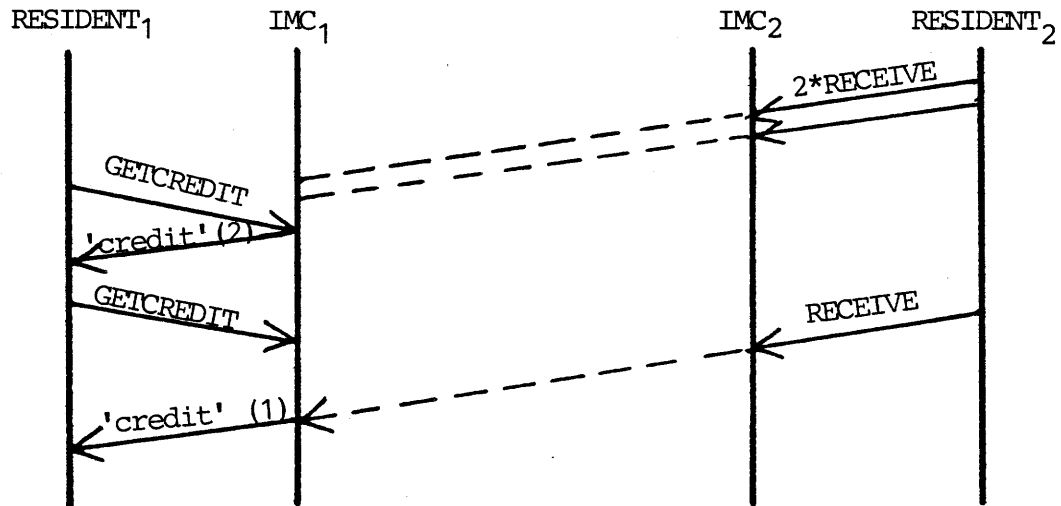


Figure 16: Credit event.

4.10 Reset Indication

4.10

Indicates that a call of RESET has been made at the remote end of the connection in question and has caused credit previously given to be taken back.

Indication of:

RESET

Enabled by:

GETRESET

Parameters:

connection	local connection identification
creditcount	number of data blocks for which credit has been taken back

4.11 Reset Completion

4.11

This event is the completion of a local call of RESET.

Completion of and enabled by:

RESET

Parameters:

connection local connection identification

4.12 Port Closed

4.12

This event is the completion of a call of CLOSEPORT. Alternatively the event may indicate, that a name conflict has arisen causing the conflicting ports to be excluded from the IMC system, or that the IMC network does not have resources to support the port.

Indication of:

Port closed by the IMC network

Completion of:

CLOSEPORT

Enabled by:

OPENPORT

Parameters:

reason diagnostics

4.13 Dummy Event

4.13

This event occurs, when the IMC for some reason has to return an event (or data) resource without delivering a proper event, i.e. when a port is closed, or a connection is removed or reset.

The event is a possible completion of most IMC function initiations, when the function is aborted for some reason. Refer to chapter 3 for a description of the individual cases of 'dummy event'.

Possible (failure)-completion of (and enabled by):

CONNECT	'local connect' resource
GETCONNECTION	'remote connect' resource
RECEIVELETTER	data resource
SEND	data resource
RECEIVE	data resource
RECEIVEALL	data resource
GETCREDIT	'credit' resource
RESET	'reset completion' resource
GETRESET	'reset indication' resource

Parameters:

(connection possibly, local connection identification)

5. SPECIAL IMPLEMENTATION FEATURES

5.

In an actual implementation of the IMC service on a specific type of PU it is possible to implement a special version or a subset of the IMC service - suited for the special requirements of the application(s) in question.

The purpose of implementing a reduced IMC service at a specified system type may be for instance:

- 1) to gain performance
- 2) to reduce the memory requirements of the IMC service
- 3) to simplify the implementation project, reducing total project costs.

As long as the internal IMC protocols and the implemented part of the basic IMC communication functions remain unchanged, an actual IMC implementation project may freely vary the actual characteristics of the user interface and the local IMC service functions.

5.1 Local IMC Services

5.1

In an implementation, a number of additional, local service functions may be developed in order to ease the use of the IMC service.

Such functions may for instance take care of parameter packing/unpacking or special control requirements such as sensing the number of active connections etc.

5.2 IMC Functional Subsets

5.2

Examples of subsets of the IMC service are:

- 1) IMC datagram communication only
- 2) IMC connection communication only

Such subsets of the total IMC service may be implemented locally without violating the 'open working' of the IMC.

Another example of a special IMC system is a preconfigured IMC system. In this case, the IMC functions OPENPORT, CLOSEPORT, GET-CONNECTION, CONNECT and DISCONNECT may be omitted, since all ports and connections have been pre-configured into the involved IMC nodes and applications.

In the future, the IMC service may be extended to offer communication classes, where a minimal transmission delay is required, thus offering REAL-TIME CONNECTIONS etc.

5.3 Different Media

5.3

By applying the concepts of layered network architecture to the problem of implementing an IMC network, it is possible to support the IMC service by means of a number of different media or low level protocols.

Thus, rather than having a universal IMC service appearing in all PUs and their interconnection media, the situation will look like figure 17 (below), where different IMC functional subsets are implemented in different basic runtime system (and language) environments, each being attachable to one or several interconnection media implying different low-level protocols.

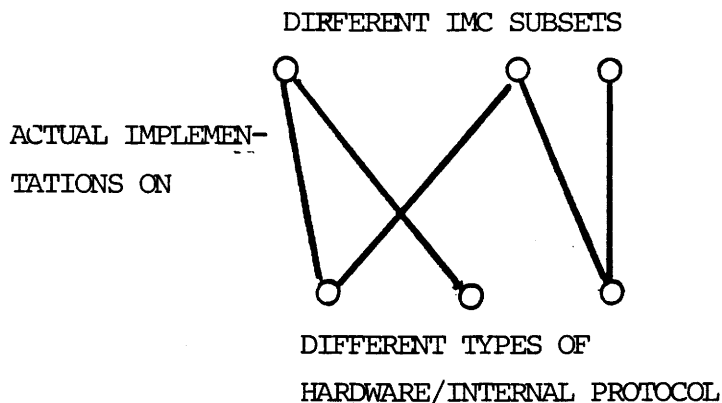


Figure 17: IMC implementations and systems.

A. REFERENCES

A.

- [1] RCSL No 42-i1982:
RC3900 Distributed System Architecture Report
April 1982
(the report is incomplete with respect to the topology of
IMC networks, and an improved revision is planned)

- [2] RCSL No 52-AA1078:
Reference Manual for the Programming Language Real-Time
Pascal

- [3] RCSL No 42-i2122:
RC3900 IMC, Reference Manual

- [4] ISO model of Open Systems Interconnection (OSI), ISO
TC97/SC16/DP7498.
E.g. found in COMPUTER COMMUNICATIONS REVIEW, vol. 11, no.
2, april 1981.

B. IMC FUNCTION SURVEY

B.

Name:	Parameters:
OPENPORT	port object name scope no. of connections 'port closed' resource
CLOSEPORT	port object
CONNECT	port object connection name 'local connect' resource (optional) 'disconnected' resource
DISCONNECT	port object connection
GETCONNECTION	port object connection 'remote connect' resource 'disconnected' resource
SENDLETTER	name data resource
RECEIVELETTER	port object data resource
SEND	port object connection data resource
RECEIVE	port object connection data resource

RECEIVEALL	port object data resource
GETCREDIT	port object connection 'credit' resource
RESET	port object connection 'reset completion' resource
GETRESET	port object connection 'reset indication' resource

C. IMC EVENT SURVEY

C.

Name:	Parameters:	(Ref. sect.:)
1. 'remote connect'	connection	(4.1)
2. 'local connect'	connection	(4.2)
3. 'disconnected'	connection reason	(4.3)
4. 'letter arrived'	data	(4.4)
5. 'letter sent'	(data)	(4.5)
6. 'data arrived'	connection data	(4.6)
7. 'data overrun'	connection data	(4.7)
8. 'data sent'	connection (data)	(4.8)
9. 'credit'	connection creditcount	(4.9)
10. 'reset indication'	connection creditcount	(4.10)
11. 'reset completion'	connection	(4.11)
12. 'port closed'	reason	(4.12)
13. 'dummy event'	(connection)	(4.13)

D. CORRELATION OF IMC FUNCTIONS TO EVENTS

D.

Below are shown the relationship between IMC function calls, completion events and indication events. i means the event is an indication event caused by the corresponding function and c indicates the event is a completion event of the function. e defines the function(s) that enables the EVENT.

IMC event		(numbers refer to appendix C)												
IMC		(RC	LC	DI	LA	LS	DA	DO	DS	CR	RI	RC	PC	DE)
function		1	2	3	4	5	6	7	8	9	10	11	12	13
OPENPORT													(c)	
'port closed'													e	
CLOSEPORT													c	
CONNECT	i	c	c											c
'local connect'		(e)												e
'disconnected'				e										
DISCONNECT				ic										
GETCONNECTION	c													c
'remote connect'	e													e
'disconnected'				e										
SENDLETTER					i	c								
data res.						e								
RECEIVELETTER					c									c
data res.					e									e
SEND							i	i	c					c
data res.									e					e
RECEIVE							c	c	(i)	i				c
data res.							e	e						e
RECEIVEALL							c	c	(i)					c
data res.							e	e						e
GETCREDIT										c				c
'credit'										e				e
RESET											i	c		c
'reset completion'												e		e
GETRESET											c			c
'reset indication'											e			e

(i = indication, c = completion, e = enable)

RETURN LETTER

Title: DSA Inter Module Communication
Functional Description

RCSL No.: 42-i1983

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

..... Fold here

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

Affix
postage
here

 **REGNECENTRALEN**
af 1979

Information Department
Lautrupbjerg 1
DK-2750 Ballerup
Denmark