Functional description of the Systems Communications Architecture

21-Nov-83

Version 1, Revision 5

1.0 Revision history

Revision 1:

1. Change user interface for packet sends to reflect data being copyed rather than mapped into monitor space.
2. Add set port counters function (.SSSPC)
3. Add configuration change PSI channel to .SSAIC JSYS function.

Note: There are no change bars for this revision. Change bars will begin with the next revision of the specification.

Revision 2:

Update path specification handling in .SSSDG and .SSSMG functions of SCS%.

Revision 3:

Make maintainance data send and receive work correctly.

Revision 4:

Fix additional problems with maintainance data send and receive.

Revision 5:

Add user level maintainance functions for reset/start a remote system. Also add "monotonic counter" to set/read port counter calls.

2.0 Table of Contents

3.0 Definitions

There are a number of terms and concepts vital to an understanding of SCA. They are:

1. Credit - Credit refers to buffers queued for message (only messages, there is no credit system for datagrams) reception or transmission. Receive credit is the number of buffers you have queued to get messages from a remote node. Send credit is the number of buffers the remote has queued to receive messages from you.

2. Callback - The callback is the mechanism through which SCA interrupts a SYSAP. For example, when a message or datagram has arrived for your connection, SCA will PUSHJ to the address given in the CONNECT, LISTEN, or set online notification address call, with T1-T4 setup to point to the new packet.

3. Port queues - This refers to the set of queues used by the monitor to communicate with the KLIPA. These queues include the message and datagram free queues, into which all inbound packets are placed.

4. Packet - Term used to denote any CI packet, I.E. a message or a datagram.

5. Circuit - Virtual communications path layered on top of the CI wire (I.E. hardware path) maintained by the port driver.

6. Connection - Virtual communications path layered on top of circuits and maintained by SCA.

4.0 Introduction

4.1 Related documents

The reader should at least be familiar with the following documents before proceeding with this specification. The corporate SCA spec is particularly important as all terminology used in this document follows standards set in the corporate spec.

1. Systems Communications Architecture, Rev 4, 20-July-82 (Strecker)

2. LCG CI Port Architecture Specification, 11-July-83 (Dossa/Keenan)

3. TOPS-20 KLIPA Driver Functional Specification, 19-Aug-83 (Grant/Miller)

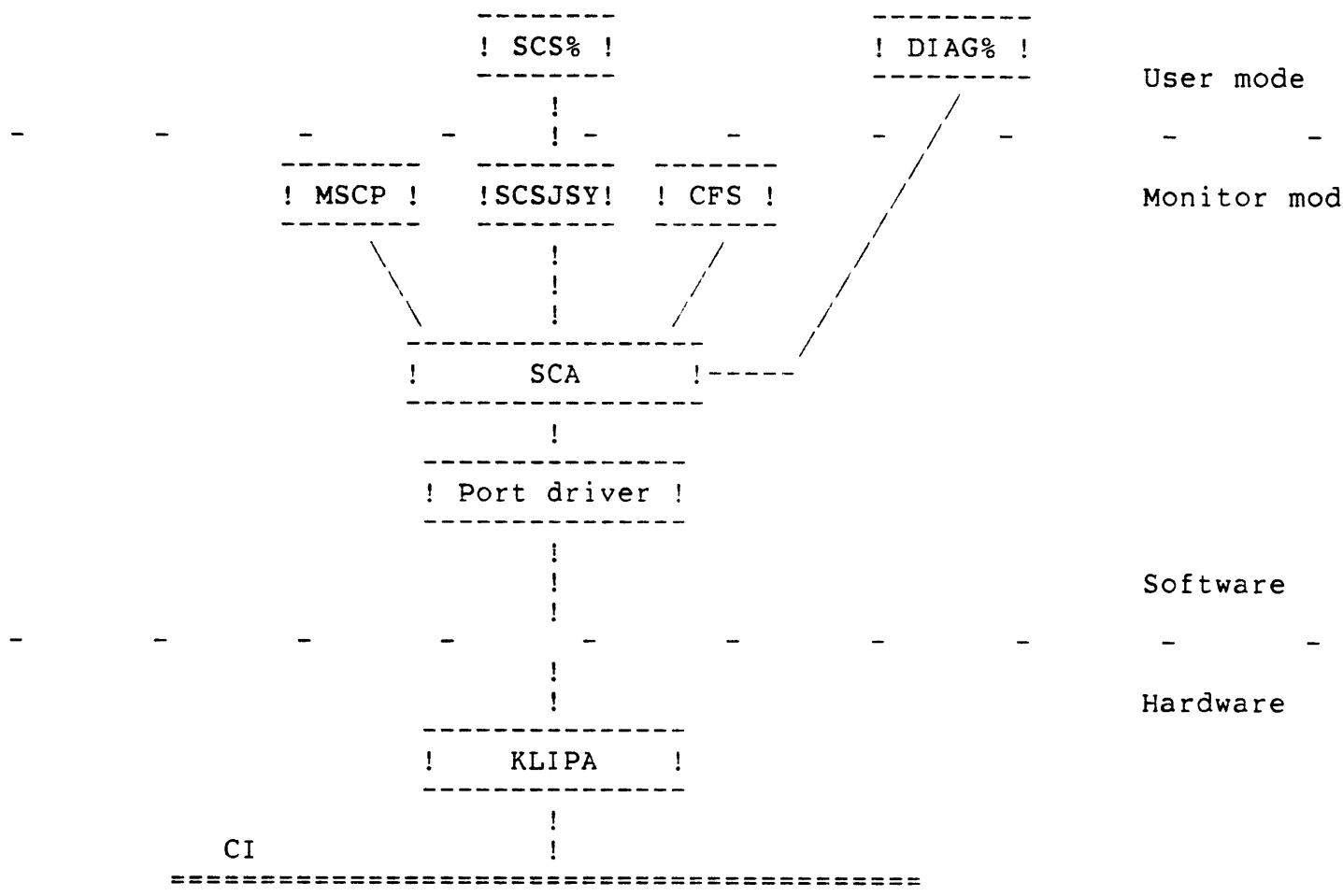
4. TOPS-20 Coding Standard, 23-Mar-83 (Murphy)

4.2 Scope

This document will describe the interfaces in and out of the Systems Communications Architecture (SCA). This includes the user interface through the SCS% JSYS. Some of the functions invisible to the rest of the monitor, yet important to the overall scheme of the CI software are also covered. Implementation details are left to the design specification however.

4.3 Module interactions

The following diagram illustrates the layout of the CI software and hardware.



Briefly, these are the functions performed by each of the software modules listed above.

The port driver is the hardware controller for the KLIPA. It communicates with the device through a queued protocol using KL main memory.

SCA is the CI gateway for the rest of the monitor. The only exception is the DIAG JSYS which has the capability of disabling the port driver and taking direct control of the hardware. In general however all monitor components access the CI through SCA.

System applications (SYSAP) are the end users of the CI. The Mass Storage Control Protocol (MSCP) is a CI disk controller. Common File System (CFS) is the global file lock manager for implementing distributed

access to CI disks and dual ported massbus disks.

4.4 SCA overview

The overall function of SCA can best be described as software multiplexer, demultiplexer for the CI. SCA is the module that creates the illusion of virtual connections over the circuits established by the port driver. A detailed description of how SCA performs this function can be found in [1]. What needs to be detailed here is exactly what has been done that is specific to TOPS-20 and hence could never be outlined in the corporate specification.

4.5 General differences

In general, the corporate specification is considered to be a model in which the seeds of the TOPS-20 implementation may be found. There are a number of global differences that need to be outlined.

1. The corporate specification uses polling to determine the completion of a request. The TOPS-20 implementation is interrupt driven.

2. TOPS-20 does not use the suggested format for connection and system blocks. Additional words were required and all support locations for path blocks were removed.

3. The most important difference is the lack of the "path" concept in TOPS-20. The corporate specification calls for the use of multiple paths to a single node. (Note that this is not the dual rail concept, I.E. Path A and Path B) The entire concept of multiple paths to a single node is dissallowed in TOPS-20. All nodes are directly connected to all other nodes by exactly one logical path. A logical path is a port onto a CI, even though the port may have more than one cable on it.

   TOPS-20 does not support paths because the KL-10 processor can currently support only one KLIPA. Since there is one KLIPA there can by definition only be one path to a node. When more than one KLIPA is supported the issue of paths will be addressed.

4.5.1 TOPS-20 only functionality

At present, there is only one function of SCA that is seen by the outside world and not covered by the corporate specification. This function is the periodic polling of nodes with open connections. The need for this service arises from the fact that the KLIPA answers REQID packets without software intervention. Since port level pollers for most machines use REQID packets to test the rails to the other nodes on the net, a system could suffer a complete software failure and no one would detect it until the port were reset during the BOOT process. There are SYSAPs that need to see remote software failures much sooner than this. Hence SCA will send credit requests for zero credit to nodes that have

not sent the local node a message for a certain time period. If t response comes back within a certain time then the node is up and SCA simply marks the fact the the credit request received a response. If the response does not come back within the defined time period, the node is declared offline, SCA requests that the port to port virtual circuit be broken, and all SYSAP's are notified of the remote node failure.

4.6 Service required of PHYKLP

Since PHYKLP maintains direct control over the hardware, there are services required of PHYKLP by SCA. These services are:

1. ULNKDG -- Remove a buffer from the datagram free queue.
2. ULNKMG -- Remove a buffer from the message free queue.
3. SNDDG -- Send a datagram
4. SNDMSG -- Send a message
5. LNKMFQ -- Link a buffer chain onto the message free queue
6. LNKDFQ -- Link a buffer chain onto the datagram free queue
7. CLOSVC -- Close a virtual circuit
8. MAPBUF -- Map a named buffer
9. UMAP -- Unmap a named buffer
10. SNDDAT -- Start a named buffer data send
11. REQDAT -- Start a named buffer data receive
12. LOCPRT -- Return the local node number
13. KLPOPN -- Open a circuit
14. PPDSRS -- Send a reset or start maintainance packet
15. PPDSMD -- Do a maintainance data send
16. PPDRMD -- Do a maintainace data read
17. PPDRPT -- Read port counters
18. PPDSPT -- Set port counters
19. STRPOL -- Start the poller
20. STPPOL -- Stop the poller
21. STRKLP -- Start the KLIPA
22. STPKLP -- Stop the KLIPA

4.7 PHYKLP interface

Direct control over the CI is done by the port driver (PHYKLP). Hence SCA must interface to the port driver to perform its function. The following is the set of SCA entry points understood by the port driver.

- SC.INT -

This routine is called when the port driver has received a packet bound for SCA.

Call

BLCAL. (SC.INT,<SS.PKA,SS.SBA,SS.FLG,SS.LEN>)

Where:
SS.PKA - Address of message/datagram/SCA buffer
SS.SBA - Address system block

SS.FLG - Flags
F.SPM - Packet data mode 0 - Industry compatable mod
1 - High density mode

F.RSP - Local or remote packet 0 - Remote packet
1 - Localy generated packet

SS.LEN - Length of packet in bytes (for industry compatible packets
- Length in words for high density packets

Return (+1)
T1/ Error code

Return (+2)
No data returned, all went well

- SC.MDC -

This routine is called by the port driver when a maintainance data transfer has completed.

Call
BLCAL. (SC.MDC,<SS.ADR,SS.NAM>)

Where:
SS.ADR -- Address passed to PHYKLP on SC.MDS/SC.MDR call
SS.NAM -- Local name for transfer which has completed

Return (+1)
T1/ Error code

Return (+2)
No data returned

- SC.DMA -

This routine handles the completion of a named buffer transfer.

```
Call
BLCAL. (SC.DMA,<SS.CID,SS.SBA,SS.NAM>)

Where:
SS.CID -- CID of connection we did this for
SS.SBA -- Address of system block
SS.NAM -- Buffer name for transfer that completed


Return (+1)
T1/     Error code

Return (+2)
No data returned
```

- SC.ERR -

This routine is called to notify SCA of a virtual circuit closure. This includes closures requested by SCA.

Call

BLCAL. (SC.ERR,<SS.NOD>)

Where:
SS.NOD --> Node number of the system who's VC just went away.

Return (+1)
T1/ Error code

Return (+2)
No data returned

- SC.ONL -

This routine is called when a node has come online.

Call
BLCAL. (SC.ONL,<SS.NOD>)

Where:
SS.NOD -- Node number of system that has come online

Return (+1)
T1/ Error code

Return (+2)
No data returned

- SC.OFL -

This routine is called when a system has gone offline. Note that SCA assumes the data for this node has not been cleaned up by the port driver.

Call
BLCAL. (SC.OFL,<SS.NOD>)

Where:
SS.NOD -- Node number of the system that has gone offline

Return (+1)
T1/ Error code

Return (+2)
No data returned

- SC.PRC -

This routine is called when the port detects that a READ-PORT-COUNTERS command has completed.

Call

```
BLCAL. (SC.PRC,<SS.CID,SS.ADR>)
```

Where:
SS.CID -- CID for whom the register read was done
SS.ADR -- Address of counter data packet

Return (+1)
T1/ Error code

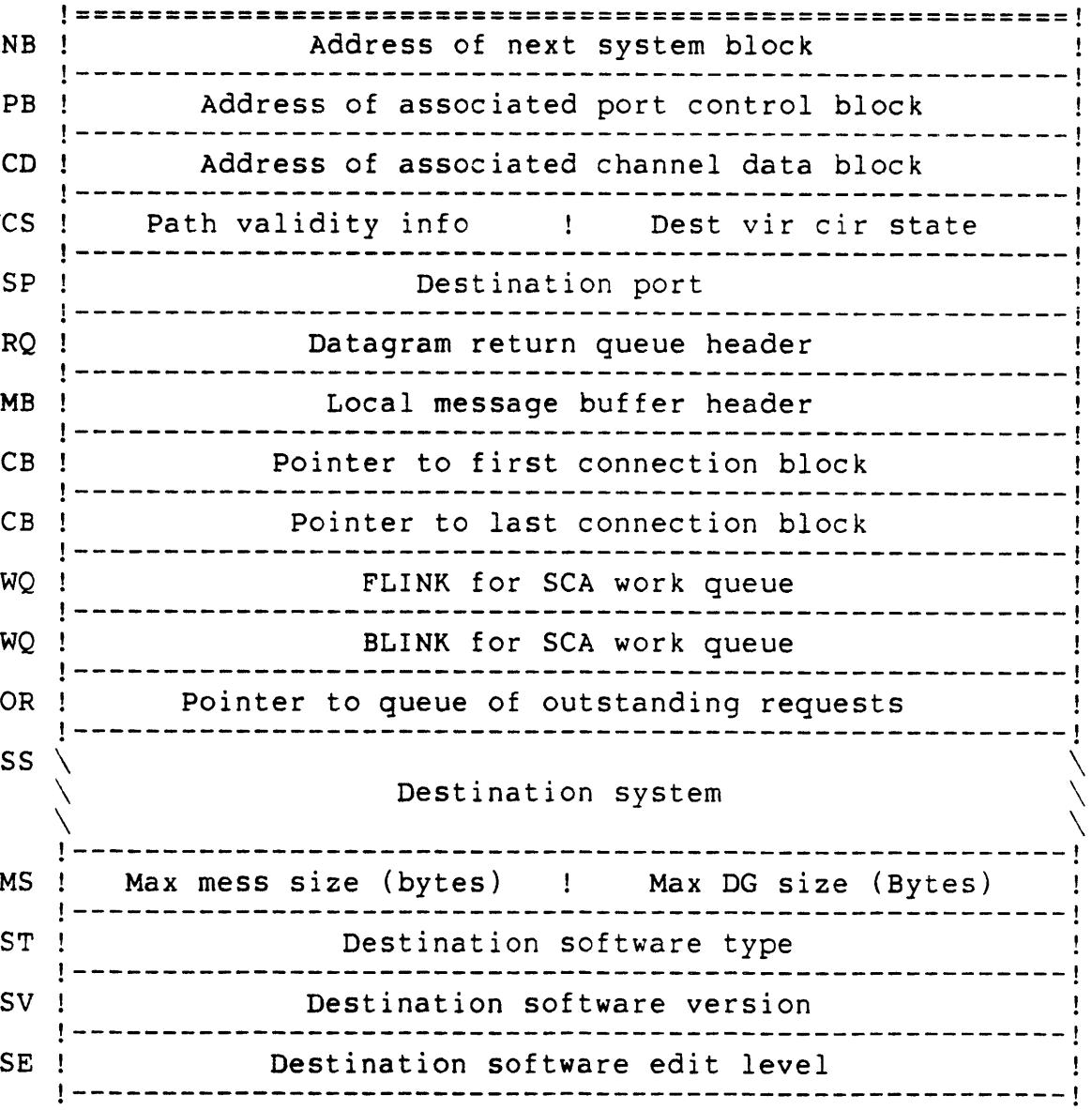
Return (+2)
No data returned

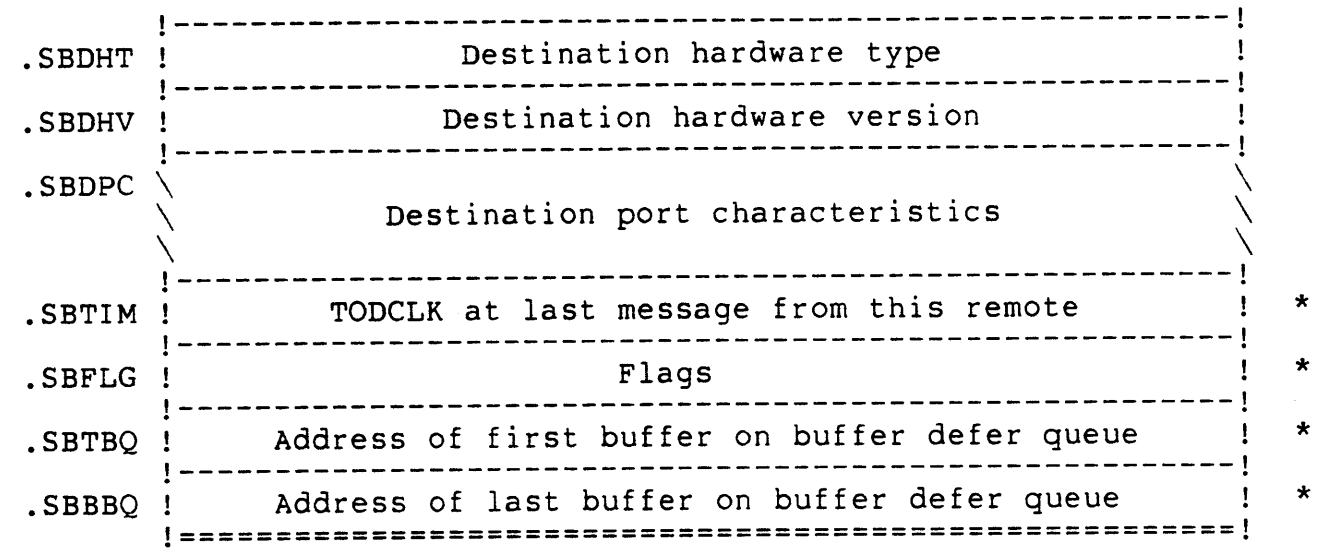
4.8 Data structures

To perform its function as multiplexer/demultiplexer SCA maintains a set of connections over the virtual circuits maintained by the port driver. There are two blocks basic to the maintenance of these connections, the system block and the connect block.

4.8.1 System block

The system block is a data structure shared with the port driver. In fact the port driver owns most of the block.





Most of the cells in this structure are maintained by the port driver. There are a few however that are the exclusive domain of SCA. These are described below.

.SBLMB
The local message buffer header is an interlock location for this remote. Since all SCA messages are flow controled, some method had to be arrived at for flow control of SCA overhead messages. This is done by always having just one receive credit available for each remote node. Hence we cannot send more than one overhead message at a time to a particular node. Hence this location is used to flag the fact that a request has been sent to the host. Note that when non-zero, this location contains the message type that is expected in response to the request we believe is outstanding.

.SBFCB/.SBLCB
The pointer to the first and last connect blocks are the FLINK and BLINK for one set of threads to the connect blocks. Each connect block is a member of two doubly linked lists. One is the list of connections to a system, the other is the list of connects owned by a fork. Note that unless the fork has done an SCS% JSYS, this second list is empty.

.SBTWQ/.SBBWQ
Something must be done with the requests that need to be sent to a system which already has a request outstanding. These requests are placed on the work queue for that system. The work queue FLINK and BLINK are stored in the system block.

.SBTIM
A function performed by SCA is the polling of remote systems that have connections open to them. This polling is done to detect a remote software stoppage. .SBTIM is TODCLK the last time the node talked to us. For further details on this process, see the section on SC.CLK.

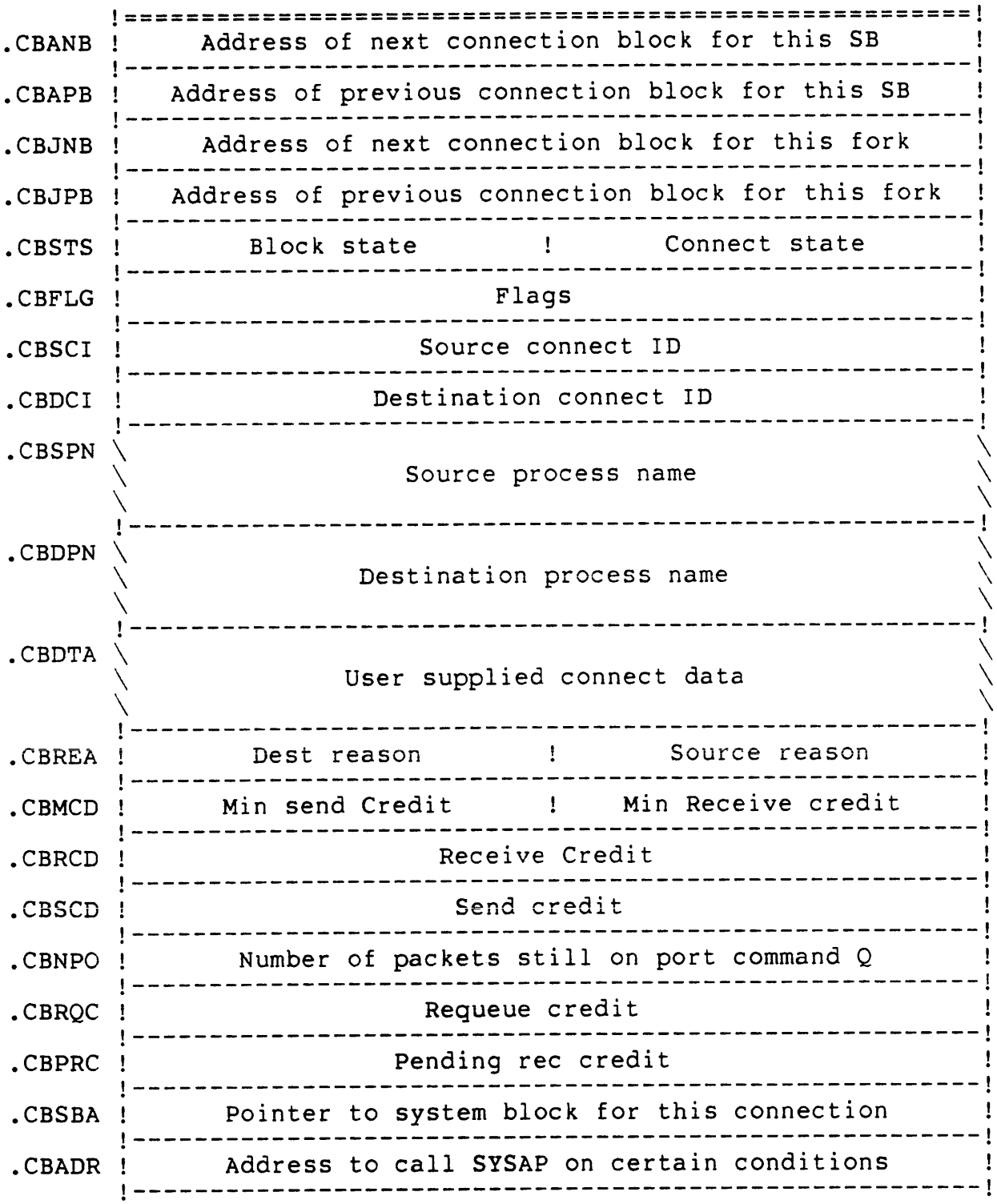
.SBFLG

Currently, the only defined flags are the timed message flag, and the circuit requires opening flag. When the timed message flag is lit in .SBFLG, a timed message is outstanding for this node. SCA uses this flag to help determine if a node should be declared dead. The circuit requires opening flag indicates that the circuit has been closed and just as soon as databases have been cleaned up, SCA should call the port driver to open the circuit again.

.SBTBQ/.SBBBQ
These words are the head and tail pointers for the queue of buffer defer requests. This queue is used by SCA to handle the allocation of more buffers than it has on hand.

4.8.2 Connect block

The other data structure used by SCA is the connect block. As the name implies this block is the complete set of information maintained about an SCA connection.



.CBCDD	Number of dropped datagrams	
.CBDGR	Number of real DG buffers queued	
.CBDGJ	Number of JSYS DG buffers queued	
.CBSBI	Destination node number	
.CBFRK	Job number of owner job	Fork number of owner fork
.CBTMQ	Pointer to top of message available queue (for JSYS)	
.CBBMQ	Pointer to bot of message available queue (for JSYS)	
.CBTDQ	Pointer to top of datagram available queue (for JSYS)	
.CBBDQ	Pointer to bot of datagram available queue (for JSYS)	
.CBTXQ	Pointer to top of the DMA xfer complete queue	
.CBBXQ	Pointer to bot of the DMA xfer complete queue	
.CBTEQ	Pointer to top of the event queue	
.CBBEQ	Pointer to bot of the event queue	
.CBTBQ	Pointer to first buffer descriptor block	
.CBBBQ	Pointer to last buffer descriptor block	
.CBPS0	PSI channel for messages	PSI channel for datagrams
.CBPS1	PSI channel for DMA	PSI channel for events

Of the above cells, a few are returned by SCA on certain routine calls. The following is a description of the cells that are of importance to a SYSAP.

.CBSTS
If non-zero, the block state indicates the connection is waiting for a response to a request. The block state also indicates what the response will be. These are the currently defined block states:

.BSCNP --> Connect pending
.BSACP --> Accept pending
.BSCRP --> Credit pending
.BSRPN --> Reject pending
.BSDPN --> Disconnect pending

As the name may indicate, the connection state indicates the current state of the connect. The following are the currently defined connection

states.

.CSDRE --> Disconnect received
.CSDSE --> Disconnect sent
.CSDAK --> Disconnect acknowledge
.CSDMC --> Disconnect match

.CSCLO --> Closed - by command
.CSCNM --> Closed - no match
.CSCRJ --> Closed - rejection
.CSCNR --> Closed - no resources
.CSCVC --> Closed - Port virtual circuit error

.CSLIS --> Listening

.CSCSE --> Connect sent
.CSCAK --> Connect acknowledge
.CSCRE --> Connect received

.CSOPN --> Connection open

.CBFLG
The flags word supports flags for both SCA and the SCS% JSYS. These are the currently defined flags:

CBFNNC --> Credit has come available since ntofication of credi lo
CBFJSY --> This is a JSYS initiated connect block
CBFABT --> CB has been aborted
CBFRAP --> CB is to be reaped
CBFDCL --> This was don't care listener
CBFKIL --> Owning fork has gone away (JSYS connect only)

.CBSCI
This is the source connect ID. This is the local name for this connection.

.CBDCI
This is the remote connect ID. This name is what the remote calls the connection.

.CBSPN/.CBDPN
These are the local and remote process names in 8 bit ASCII.

.CBDTA
This is the initial connection data sent during the handshake for opening the connection. It is always the data sent to us by the remote.

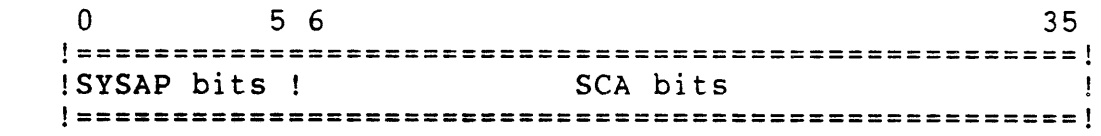
.CBREA
These are the remote and local disconnect reasons. When a disconnect is started, the local code is transmitted as the cause for the disconnect. When the disconnected is initiated remotely, the destination reason is stored from the disconnect request packet.

.CBRCD/.CBSCD

These cells contain the current credit levels for this connection. The receive credit reflects the number of buffer queued on this system for the reception of message from the remote. The send credit is the number of buffers the remote has queued for receiving localy generated messages.

4.8.3 Connect ID

The connect ID is the one word unique identifier for any connection. It is mentioned here because the SYSAP needs to understand the fprmat of the connect ID. The ID is made up of two fields, the SYSAP field and the SCA field. The SYSAP field contains bits defined by the SYSAP and passed to SCA when connection blocks area being created (the connect and listen calls). The SCA field is broken down into fields for use by SCA, but the SYSAP only needs to know that the entire field is for the use of SCA. The following is the format of the connect ID:



When the SYSAP field is passed to SCA it is passed in bits thirty (30) through thirty-five (35) of the BLCAL. argument. SCA will move the bits into the correct position for placement into the connect ID.

5.0 Writing a monitor SYSAP

5.1 General rules

To talk to other processes on remote systems, SYSAPs request services of SCA. There are some rules that must be followed if the conversation is to go smoothly.

5.1.1 Buffer allocation and management

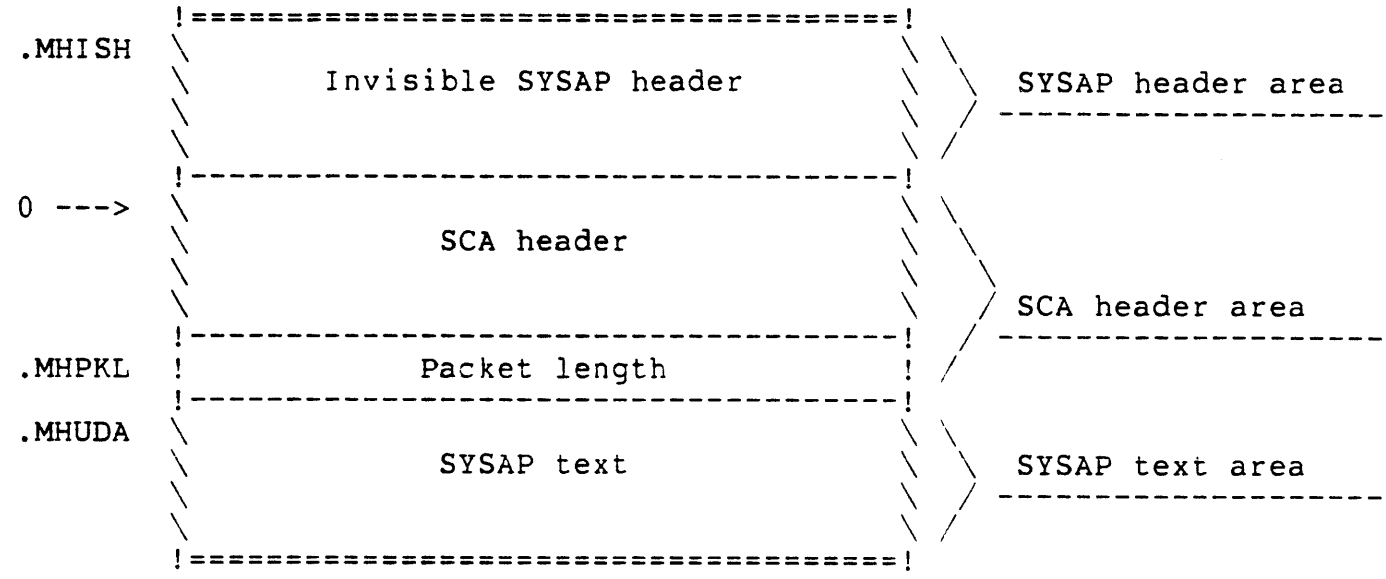
There are four buffer types that monitor SYSAPs may have to deal with, messages and datagrams, both of which can be inbound or outbound.

Rule one, ALL inbound buffers are allocated and controlled by SCA. When a SYSAP desires to queue buffers for message or datagram reception, call SCA (see calling sequence for SC.RMG and SC.RDG) and indicate how many you desire, or specify the address of a buffer already allocated by SCA. In the normal case, start up by asking for a number of buffers, and after they have been returned with text in them, return them to SCA as fresh buffers for the port queues. SYSAPs MUST NOT specify a buffer not allocated by SCA when queueing inbound buffers.

Rule two deals with outbound buffers. SYSAPs may ask SCA for a packet buffer (see calling sequence for SC.ABF and SC.ALD), fill it with data, and ask SCA to send it (see SC.SMG and SC.SDG). If the SYSAP uses SCA buffers for sending packets, the SYSAP may request the buffer go to the port free queue on packet send completion. Hence the SYSAP may do a packet send and queue a buffer for reception all at the same time. Note that only SCA buffers may end up on the port free queues. Hence SYSAPs may not allocate their own space and ask for these buffers to end up on the port free queue.

If the SYSAP desires to allocate its own send buffers, it may do so but the space allocated must meet rigid standards. First, the buffer must be contiguous in physical memory. Hence if a page boundry is crossed, the two virtual pages must be physically contiguous. Second, the start of the SYSAP text must be at least .MHUDA words into the page. This is used by SCA for header space. Third, when the packet send is done the SYSAP must ask for the buffer back. It MUST NOT end up on the port free queues.

SYSAPs must understand the format of SCA buffers if they are to be used correctly. The following is the general format of an SCA buffer:



.MHISH - This symbol defines the offset to the base of the invisible SYSAP header. It is important to note that this is a negative offset.

This header is for the exclusive use of the SYSAP. SCA will zero it when the SYSAP requests the buffer but otherwise will not touch it. This means the SYSAP can write data to this area and have it survive across port packet transmission.

0 - This is word zero of the buffer. It is the base address which is passed between SCA and te SYSAP when talking about this buffer. All offsets within the buffer are defined as offsets from this word.

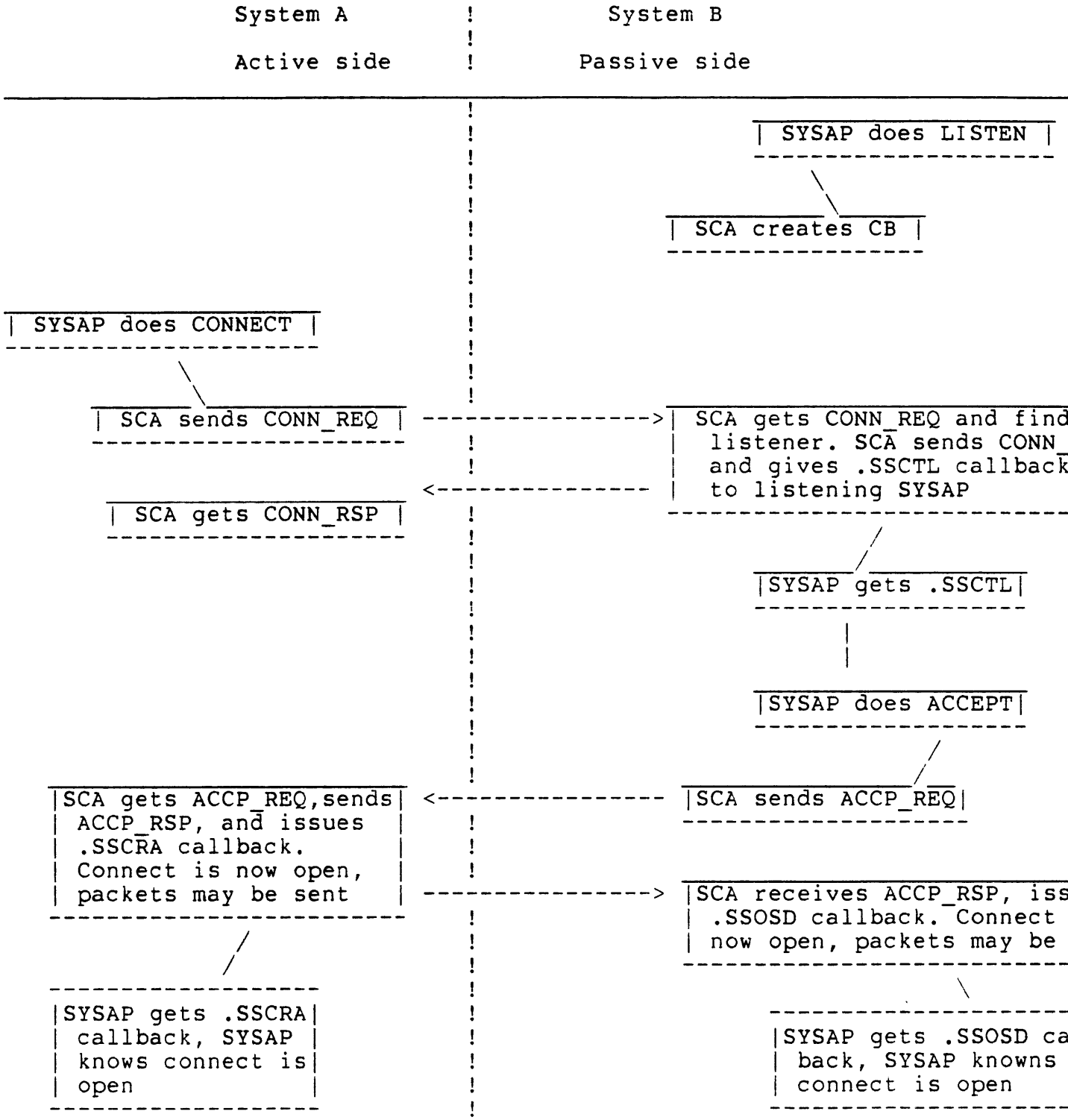
.MHPKL - This word is used by SCA to describe the length of an incoming packet. It is always filled in by SCA when it is called by the port driver for an incoming packet. This word is not filled in by the SYSAP on a message send.

This word is always the length of the SYSAP text, I.E. it never includes the length of any overhead data. If the packet was transmitted in high density mode, then the length is in words. If the transmission mode was industry compatable, then the length is in bytes.

.MHUDA - This offset defines the first word of the SYSAP text area. SCA will never touch this area except to zero it on buffer allocation.

5.1.2 Starting a connection

Before application packets can be exchanged between processes on two systems, SCA must mediate the opening of a connection. The following is a graphic description of this process.



The diagram above is a typical connection opening sequence. The important thing to watch is the set of callbacks from SCA. They are the direct indicator of what state the connect is in. Note that there are other options than the set indicated by the diagram. The detailed SCA/SYSAP interface details these options.

5.1.3 Global interlocks

There are times when a SYSAP desires to insure that no further CI traffic will be processed until it has completed some piece of code. Using PIOFF to accomplish this is undesirable since it does not allow for nesting and it turns off more channels than are necessary. Instead of PIOFF/PION, CIOFF/CION should be used. These macros call routines in SCAMPI which will do nesting and interrupt level checks. At present, there are 36 bits of nesting counter, and code executed at KLIPA interrupt level will not turn off the channel. Since these nesting checks are done, the SYSAP is guaranteed that the channel will stay off until it does a CION. Note that one side effect of going CIOFF is being NOSKED as well.

5.2 Messages

5.2.1 Receiving messages

To receive messages from a remote node a SYSAP must do the following:

1. Open a connection to the desired remote.
2. Queue message buffers.

The SYSAP will now receive a .SSMGR callback for each message that arrives for it.

5.2.2 Sending messages

To send messages the SYSAP must have the following:

1. An open connection with the desired remote.
2. Send credit.
3. A buffer which meets the requirements for placement on the port command queues.

If the connection has no send credits the remote node has not queued any buffers for receiving messages from you. Hence you cannot send any and will get the fail return from SC.SMG until the remote queues some buffers.

5.3 Datagrams

5.3.1 Receiving datagrams

Much like messages, datagrams require an open connection to the remote and buffers queued for datagram reception. The major difference is that when the remote does a send, if it fails because there are no buffers available for your connection, you will get the .SSDDG callback. This is mearly an indication that the software has detected a dropped datagram for your connection and you should queue some buffers for it. If the hardware drops the datagram because there are no buffers on the entire hardware queue, you will never hear about it. Hence the SYSAP cannot count on being told about dropped datagrams. Note that SCA never notifies the sender that the datagram was dropped.

5.3.2 Sending datagrams

Sending datagrams requires an open connection to the remote and a packet buffer that meets the requirements for placement on the port command queues. (See the section on buffers for more details)

5.4 Named buffers

The general procedure for the transfer of data with named buffers is as follows:

1. Both sides of an open connection do a map call to set up a buffer.
2. SYSAP A gives its name for the buffer to SYSAP B.
3. The SYSAP B does a request or send data function which starts the data in the desired direction.
4. The SYSAP B gets a callback indicating the completion of the named buffer transfer.
5. Lastly, and optionally, SYSAP B tells the SYSAP A that the transfer is complete.

This method implies that each SYSAP has message buffers queued. The data transfer functions require a buffer from the message free queue. The exchange of names should be done with messages or datagrams.

5.5 Monitor SYSAP interface

5.5.1 SYSAP to SCA interface

SCA provides many services to the SYSAP though the use of these calling sequences:

5.5.1.1 Connect (SC.CON)

- Connect (SC.CON) -

This routine requests a connect with a remote process.

```
        Call
BLCAL. (SC.CON,<SS.SPN,SS.DPN,SS.NOD,SS.MSC,SS.MRC,SS.ADR,^_
                SS.SID,SS.DTA,SS.BFR,SS.DGB>)
```

Where:

```
SS.SPN -- Byte pointer to source process name
SS.DPN -- Byte pointer to destination process name
SS.NOD -- Node number of destination system
SS.MSC -- Minimum send credit (Min for remote receive
                credit)
SS.MRC -- Minimum receive credit, point at which
                destination must give more
SS.ADR -- Address to call on condition changes for
                this connection
SS.SID -- Value to be placed in SYSAP field in CID (right justifie
SS.DTA -- Address of SYSAP connection data or zero
SS.BFR -- Number of buffers to queue for message reception
SS.DGB -- Number of datagram buffers to queue

Return (+1)
T1/     Error code

Return (+2)
T1/     Connect ID
```

The returned connect ID is the unique identifier for this connection. All further interaction with SCA will include this connect ID (CID).

5.5.1.2 Listen (SC.LIS)

- Listen (SC.LIS) -

This routine enables the process to listen for connections from remote processes.

Call

```
BLCAL. (SC.LIS,<SS.SPN,SS.DPN,SS.NOD,SS.ADR,SS.SID,SS.DTA,^_
                SS.MSC,SS.MRC>)
```

Where:

SS.SPN -- Byte pointer to source process name
SS.DPN -- Byte pointer to destination process name or 0
SS.NOD -- Node number or -1
SS.ADR -- Address to call on connection condition changes
SS.SID -- Value to be placed in SYSAP field of CID (Right justified
SS.MSC -- Minimum send credit
SS.MRC -- Minimum receive credit

Return (+1)
T1/ Error code

Return (+2)
T1/ Connect ID

When SCA receives a connect request from a remote node, the following algorithm is used when doing name matching with local listeners:

Figure 1

A successful return from this call means SCA has a held a connecti. for you waiting for connects from remote nodes. If a remote requests a connection with a process, that process will receive the .SSCTL callback.

5.5.1.3 Accept (SC.ACC)

- Accept (SC.ACC) -

This routine is used to accept a connect request from a remote node.

Call

```
BLCAL.  (SC.ACC,<SS.CID,SS.DTA,SS.BFR,SS.DGB>)
```

Where:

```
SS.CID -- Connect ID
SS.DTA -- Address of initial connection data or zero
SS.BFR -- Number of buffers to be queued for message reception
SS.DGB -- Number of buffers to queue for datagram reception

Return (+1)
T1/     Failure code

Return (+2)
T1/     Address of initial connect data
```

The returned address of connect data is a pointer to the connection data sent by the remote on the connect request. Note that the connection is not open until the .SSOSD callback is given.

5.5.1.4 Reject (SC.REJ)

- Reject (SC.REJ) -

This routine rejects the connect request of a remote process.

Call

BLCAL. (SC.REJ,<SS.CID,SS.RJR>)

Where:

SS.CID -- Connect ID
SS.RJR -- Reject reason code

Return (+1)
T1/ Failure code

Return (+2)
No data is returned by this routine

The reject reason may be any value the caller desires with the exception of those reserved to SCA (in the range /TBD/). The idea is that the SYSAP protocol should invent and use its own reject reasons. After the reject call has been issued, the connection goes back into the listen state.

5.5.1.5 Disconnect (SC.DIS)

- Disconnect (SC.DIS) -

This routine closes an open connection. Note that it may only be called if the connection is open. No other connection state is allowed for a connection that is to be disconnected.

Call

```
BLCAL.  (SC.DIS,<SS.CID,SS.DIR>)
```

Where:

SS.CID -- Connect ID
SS.DIR -- Disconnect reason

Return (+1)
T1/ Failure code

Return (+2)
No data is returned by this routine

The disconnect reason may be any value the caller desires with the exception of those reserved to SCA (in the range /TBD/). The idea is that the SYSAP protocol should invent and use its own disconnect reasons.

5.5.1.6 Abort (SC.ABT)

- Abort (SC.ABT) -

This routine is called to abort, or cancel, a connection. It differs from disconnect in that it may be issued for a connection in any state. Whereas disconnect may only be issued for an open connection.

Call

BLCAL. (SC.ABT,<SS.CID>)

Where:
SS.CID -- Connect ID of the connection to be aborted

Return (+1)
T1/ Error code

Return (+2)
No data returned, CID is no longer valid, and the connection has bee
aborted.

5.5.1.7 Send datagram (SC.SDG)

- Send datagram (SC.SDG) -

This routine is called to send a datagram over an open connection.

Call

BLCAL. (SC.SDG,<SS.CID,SS.FLG,SS.LDG,SS.ADG>)

Where:
SS.CID -- Connect ID
SS.FLG -- Flag word
SS.LDG -- Len of datagram (in bytes)
SS.ADG -- Address of datagram buffer

Return (+1)
T1/ Failure code

Return (+2)
No data is returned by this routine, datagram
placed on port queue

The flags word is used to indicate the transmission mode of the packet, and hence the units on the packet size. It also indicates wh should be done with the buffer after the message send is complete. See the section on buffer management for more details. These are the defined

flags for this word:

F.RTB -- 1 - return message send buffer to SCA
0 - return message send buffer to free Q

F.SPM -- 1 - Send mess/datagram in high den mode
0 - Send mess/datagram in industry compat mode

5.5.1.8 Queue datagram receive buffers (SC.RDG)

- Queue datagram receive buffers (SC.RDG) -

This routine puts buffers onto the port datagram free queue for you.

Call

BLCAL. (SC.RDG,<SS.CID,SS.NUM,SS.ADR>)

Where:

SS.CID -- Connect ID

If SS.ADR is zero then:
SS.NUM -- Number of buffers to queue

If SS.ADR is non-zero, then:
SS.NUM is ignored, and
SS.ADR -- Address of the buffer to queue. Note that this buffer must be a buffer procured from SCA. This means it has been used as a datagram buffer before. This is handy for the time you are done with an SCA datagram buffer and wish to requeue it. It eliminated the need for two calls to SCA.

Return (+1)
T1/ Error code

Return (+2)
Buffer(s) have been queued to receive datagrams.

5.5.1.9 Send a message (SC.SMG)

- Send a message (SC.SMG) -

This routine sends a message if there are sends credits available and the circuit is open.

Call

```
BLCAL.  (SC.SMG,<SS.CID,SS.FLG,SS.LMG,SS.AMG,SS.PRI,SS.TSH>)
```

Where:

SS.CID -- Connect ID
SS.FLG -- Flag word, see SC.SDG for a definition
SS.LMG -- Length of message in bytes
SS.AMG -- Address of message buffer
SS.PRI -- Priority to give message
SS.TSH -- Allow send only if more than this many send credits exis

Return (+1)
T1/ Error code

Return (+2)
No data returned

5.5.1.10 Queue message receive buffers (SC.RMG)

- Queue message receive buffers (SC.RMG) -

This routine places buffers onto the port message free queue and reflects these buffers as receive credit for the given connection.

Call

BLCAL. (SC.RMG,<SS.CID,SS.NRB,SS.AMB>)

Where:

SS.CID -- Connect ID

SS.NRB -- If SS.AMB is zero then this is the number of buffers to be queued for message reception.

SS.AMB -- Address of message buffer
If SS.AMB is zero then a buffer is taken from the SCA message buffer free queue. If it is non-zero, then it is assumed that no problems will arise if the buffer is returned to the SCA pool. Since it may end up in the SCA pool it must be the length of all other buffers in the pool, must have come from the general free pool,and must be resident.

Return (+1)
T1/ Error code

Return (+2)
Buffer has been queued.

5.5.1.11 Cancel datagram receive buffer (SC.CRD)

- Cancel datagram receive buffer (SC.CRD) -

This routine will take buffers off the port datagram free queue. It will not allow the caller to take more buffers than were queued for the given connection. The buffers are returned to the SCA free pool and not to the caller.

Call

BLCAL. (SC.CRD,<SS.CID,SS.NBF>)

Where:
SS.CID -- Connect ID
SS.NBF -- Number of buffers to dequeue

Return (+1)
T1/ Error code

Return (+2)
No data returned, buffer dequeued.

5.5.1.12 Cancel message receive buffers (SC.CRM)

- Cancel message receive buffers (SC.CRM) -

This routine will dequeue buffers from the message free queue. Note that it will not allow the caller to dequeue more buffers than were queued for the given connection. The buffers are returned to the SCA free pool and not to the caller.

Call

BLCAL. (SC.CRM,<SS.CID>)

Where:
SS.CID - The connection ID of connection who's buffer we are cance
SS.NBD - The number of buffers to dequeue

Return (+1)
T1/ Error code

Return (+2)
No data returned, buffer(s) have been returned to the free pool

5.5.1.13 Map a named buffer (SC.MAP)

- Map a named buffer (SC.MAP) -

This routine associates memory with a named buffer. Provided a descriptor block (shown below) it will get the port driver to give the buffer to the port.

Call

BLCAL. (SC.MAP,<SS.BLK>)

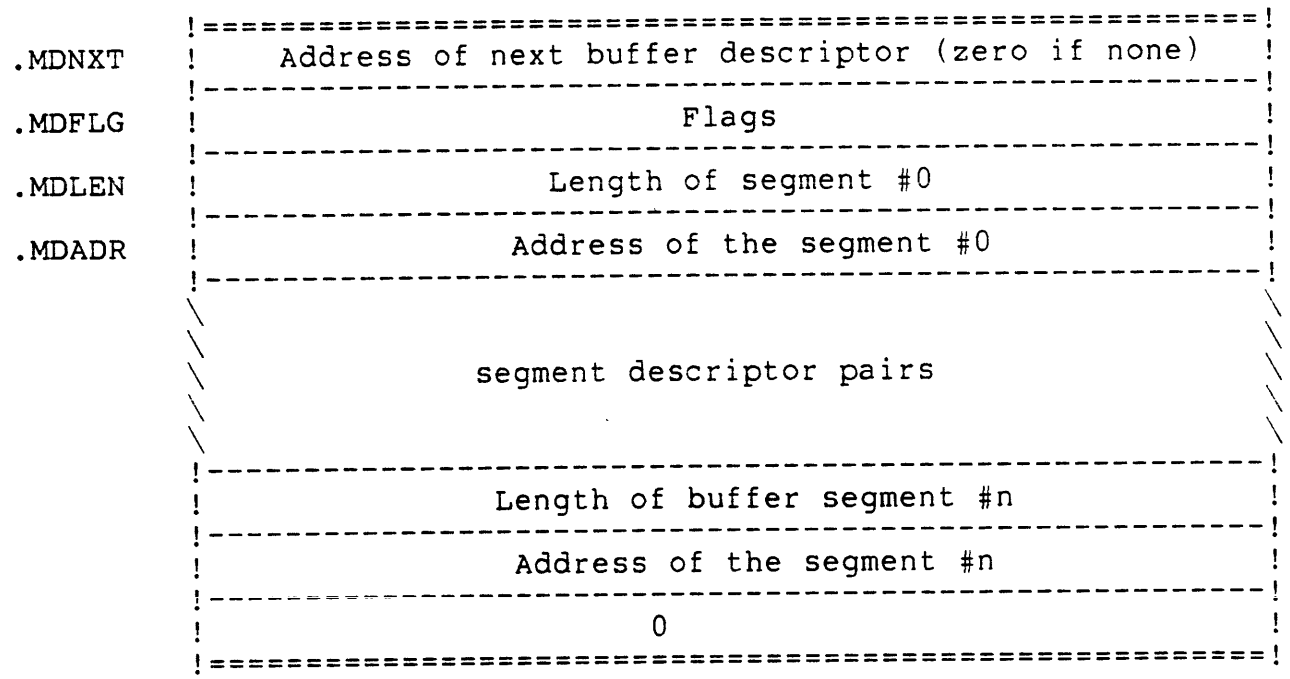
Where:

SS.BLK -- Address of the first entry in a chain of buffer descriptor blocks.

Return (+1)
T1/ Error code

Return (+2)
T1/ Buffer name

Buffer descriptor blocks have the following format:



The flags word has the following set of flags defined:

1. MD%DMD -- Two bit field indicating buffer mode. The following settings are defined:
   MD%DIC -- Industry compatable
   MD%DCD -- Core dump
   MD%DHD -- High density

All other values are illegal.

Note that .MDNXT and .MDFLG are offsets from the start of the block to the address of the next entry in the chain, but .MDLEN and .MDADR are offsets within the words describing a single buffer segment.

5.5.1.14 Unmap a buffer (SC.UMP)

- Unmap a buffer (SC.UMP) -

This routine is called to unmap a buffer. Note that this routine must be called before the buffer will go away. Hence if the routine is not called after all named buffer transfers are completed the buffer will stay around forever. This enables the SYSAP to do multiple operations with the same buffer, without having to map the buffer every time.

Call

BLCAL. (SS.UMP,<SS.NAM>)

Where:

SS.NAM -- Name of buffer to be unmapped

Return (+1)
T1/ Error code

Return (+2)
No data, buffer name is no longer valid

5.5.1.15 Send named buffer data (SC.SND)

- Send named buffer data (SC.SND) -

This routine sends block data from a named buffer.

Note:
One receive credit is required to do any named buffer transfer.

```
Call
BLCAL. (SC.SND,<SS.CID,SS.SNM,SS.RNM,SS.SOF,SS.ROF>)
```

Where:

```
SS.CID -- Connection on which this transfer is being done
SS.SNM -- Name of the buffer to get data from
SS.RNM -- Name of buffer to put data into
SS.SOF -- Byte offset into transmission buffer
SS.ROF -- Byte offset into reception buffer

Return (+1)
T1/     Error code

Return (+2)
No data returned.
```

5.5.1.16 Request remote named buffer transfer (SC.REQ)

- Request remote named buffer transfer (SC.REQ) -

This routine requests the transfer of data from a remote data buffer to a local one. Note that do perform this function the SYSAP must have at least one receive credit. This is due to the ports use of a message buffer from the message free queue on a named buffer operation.

Call

```
BLCAL. (SS.REQ,<SS.CID,SS.NOD,SS.SNM,SS.RNM,SS.SOF,SS.ROF>)
```

Where:

```
SS.CID -- Connection on which this transfer is being done
SS.NOD -- Node number of remote host
SS.SNM -- Name of remote buffer to get data from
SS.RNM -- Name of local buffer to put data into
SS.SOF -- Byte offset into transmission buffer
SS.ROF -- Byte offset into reception buffer
```

```
Return (+1)
T1/     Error code

Return (+2)
No data returned.
```

5.5.1.17 Connect state poll (SC.CSP)

- Connect state poll (SC.CSP) -

This routine returns information about the state of a connection.

```
Call
BLCAL.  (SC.CSP,<SS.CID,SS.FRE>)

Where:
SS.CID -- Connect ID of target connect
SS.FRE -- Address of a free space block for returned info

Return (+1)
T1/     Failure code

Return (+2)
T1/     Address of returned data
```

Note that it is the SYSAP that provides the free space for the return of data. This free space may be from an extended section or section zero. The format of the retutned data is as follows:

```
        !=====================================================!
.CDCST  !                    Connection state                 !
        !-----------------------------------------------------!
.CDDCI  !                 Destination Connect ID              !
        !-----------------------------------------------------!
.CDDPN  !        Byte pointer to destination process name     !
        !-----------------------------------------------------!
.CDSBI  !                      Node number                    !
        !-----------------------------------------------------!
.CDACB  !              Address of connection block            !
        !-----------------------------------------------------!
.CDREA  ! Source disconnect reasons !  Dest disconnect reasons !
        !=====================================================!
```

The byte pointer to destination process name points to a string in an SCA database. This is important to realize since the SYSAP must not change the indicated string.

5.5.1.18 Return destination connect ID (SC.DCI)

- Return destination connect ID (SC.DCI) -

This routine returns the remotes identifier for the given connection.

```
Call
BLCAL. (SC.DCI,<SS.CID,SS.FRE>)

Where:
SS.CID -- Connect ID for target connection
SS.FRE -- Address of free space into which data is to be returned

Return (+1)
T1/     Error code

Return (+2)
T1/     Destination connect ID for given connection
```

5.5.1.19 Return system configuration data (SC.RCD)

- Return system configuration data (SC.RCD) -

This routine returns information about a system. To discover who is available on the net simply call this routine for each possible node number. Node numbers are guarenteed to start at zero and and at

```
Call
BLCAL.  (SC.RCD,<SS.NOD>)
```

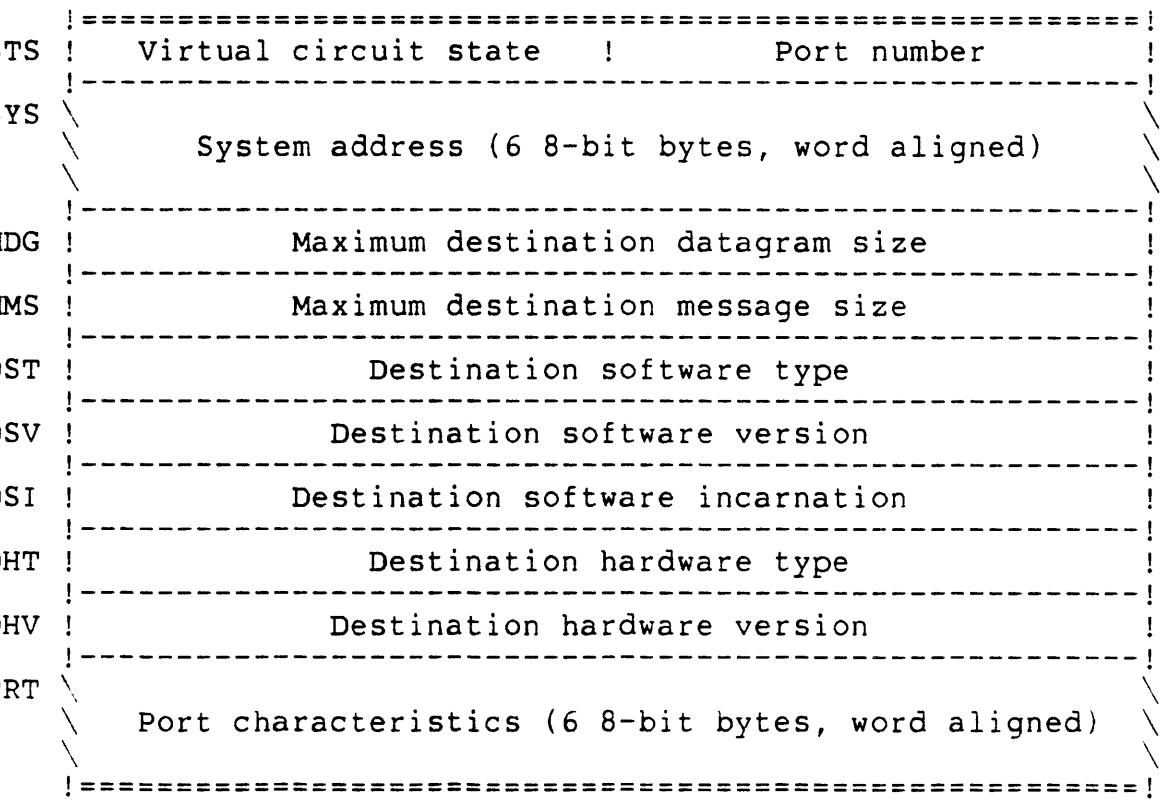
Where:

```
SS.NOD -- Target node number

Return (+1)
T1/     Failure code

Return (+2)
T1/     Address of configuration data block
```

The free space into which data is retuedn may be in an extended section or section zero, and has the following format:



The format of the system name and port characteristics fields are /TBD/.

5.5.1.20 Reset a remote system (SC.RST)

- Reset a remote system (SC.RST) -

This routine sends a maintenance reset to a remote node. No checking is done as to the node type. If the remote does not honor such packets then the remote will simply ignore the packet. There is no direct indication to the local SYSAP as to whether the packet was taken or ignored.

Call

```
BLCAL. (SC.RST,<SS.NOD,SS.FRB>)
```

Where:

SS.NOD -- Node number of node to be reset
SS.FRB -- Force reset bit, if one, reset is forced, if zero, reset will only be done if we are the last node to do a reset for this remote

Return (+1)
T1/ Error code

Return (+2)
No data returned, reset packet sent

5.5.1.21 Start a node (SC.STA)

- Start a node (SC.STA) -

This routine sends a maintainance start a remote node. No check of remote node type is done. If the remote node does not honor this packet type then the remote will simply ignore the packet. No direct indication of success or failure of the packet at the remote is provided.

Call

BLCAL. (SC.STA,<SS.NOD,SS.DSA,SS.STA>)

Where:

SS.NOD -- Node number of target remote
SS.DSA -- Flag for use of SS.STA, if zero, SS.STA is ignored and the default start address is used. If non-zero, SS is used as the start address of the remote.
SS.STA -- Start address for the remote node. Only used if SS.DSA i non-zero.

Return (+1)
T1/ Error code

Return (+2)
No data returned

5.5.1.22 Set port counters (SC.SPC)

- Set port counters (SC.SPC) -

This function sets the port counters for a desired function.

Call
BLCAL. (SC.SPC,<SS.NOD,SS.CTL>)

Where:
SS.NOD -- Node number or 377 for all nodes
SS.CTL -- Control word for port counters, bits as follows:

B0	If on, count ACKs received on Path A.
B1	If on, clear the counter.
B2	If on, count NAKs received on Path A.
B3	If on, clear the counter.
B4	If on, count NO_RSPs received on Path A.
B5	If on, clear the counter.
B6	If on, count ACKs received on Path B.
B7	If on, clear the counter.
B8	If on, count NAKs received on Path B.
B9	If on, clear the counter.
B10	If on, count NO_RSPs received on Path B.
B11	If on, clear the counter.
B12	The count of discarded datagrams because of no DGFree Queue entries.
B13	If on, clear the counter.
B14	Count the packets transmitted to the designated port.
B15	If on, clear the counter.
B16	Count the packets received from the designated port.
B17	If on, clear the counter.

Return (+1)
T1/ Error code

Return (+2)
No data retutned, port counters set

5.5.1.23 Read port counters (SC.RPC)

- Read port counters (SC.RPC) -

This routine returns the values of the port counters. It is an asynchronous event. The routine will return immediatly but the answer will occur as the .SSRPC callback.

```
Call
BLCAL. (SC.RPC,<SS.ADR>)

Where:
SS.ADR -- Address of SCA allocated buffer into
                  which data is to be returned

Return (+1)
T1/     Error code

Return (+2)
No data returned, .SSRPC callback will happen
                           when data is ready
```

5.5.1.24 Maintainance data send (SC.MDS)

- Maintainance data send (SC.MDS) -

This routine requests a block tranfser write, in maintainance mode. No SCA connection or port circuit are required for this function. The buffer must have been mapped with the SC.MAP call however.

```
Call
BLCAL. (SC.MDS,<SS.NOD,SS.SNM,SS.RNM,SS.XOF,SS.ROF,SS.ADR>)

Where:
SS.NOD -- Node number of target system
SS.SNM -- Local buffer name for transfer
SS.RNM -- Remote buffer name for transfer
SS.XOF -- Transmit offset (in words)
SS.ROF -- Receive offset (in words)
SS.ADR -- Address to call when transfer is compelte

Return (+1)
T1/     Error code

Return (+2)
No data returned, notification by .SSMNT callback when completed
```

5.5.1.25 Maintainance data read (SC.MDR)

- Maintainance data read (SC.MDR) -

This routine requests a block tranfser read, in maintainance mode. No SCA connection or port circuit are required for this function. The buffer must have been mapped with the SC.MAP call however.

```
Call
BLCAL. (SC.MDR,<SS.NOD,SS.SNM,SS.RNM,SS.XOF,SS.ROF,SS.ADR>)

Where:
SS.NOD -- Node number of target node
SS.SNM -- Remote buffer name for transfer
SS.RNM -- Local buffer name for transfer
SS.XOF -- Transmit offset (in words)
SS.ROF -- Receiver offset (in words)
SS.ADR -- Address to call when transfer is complete

Return (+1)
T1/     Error code

Return (+2)
No data returned, .SSMNT callback when complete
```

5.5.1.26 Set online address (SC.SOA)

- Set online address (SC.SOA) -

Notification of a node coming online happens on a per SYSAP basis rather than on a per connection basis. Hence SCA must have an address for each SYSAP that wishes to be called when a new VC has been opened. This routine sets the online address for the SYSAP. Note that this routine does not know who calls it and will simply add this address to the list of notification addresses. It does not try to delete old entries if called twice by the same SYSAP.

Note that this call will only allow .SSNCO callbacks to occur. All other callbacks are done through the address specified at connect or listen time.

Call

BLCAL. (SC.SOA,<SS.ADR>)

Return (+1)
T1/ Error code

Return (+2)
No data, address saved

5.5.1.27 Swap bytes from 11 to 10 format (SC.ISW)

- Swap bytes from 11 to 10 format (SC.ISW) -

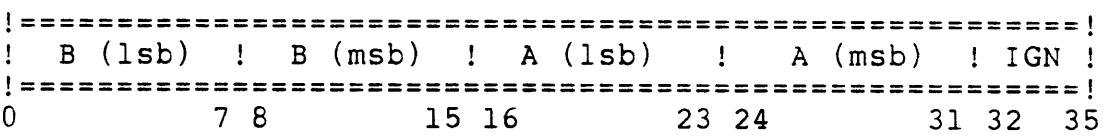
This support routine swaps a single PDP-11 format word into a PDP-10 format word. Since this is a popular thing to do when talking to the HSC and friends this routine is global for access by the rest of the monitor.

Call

T1/ Word to be swapped

Return (+1) Always
T1/ Byte swapped word

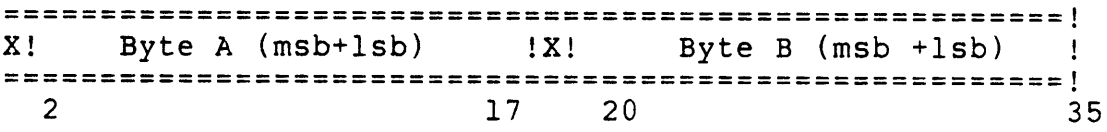
Input format:
2 16 bit half words, left justified



Where A and B are the half words, and (lsb) are the least significant bit and (msb) are the most significant bits.

The contents of bits 32 thru 35 are ignored.

Output format:
2 18 bit half words,



X denotes the 2 bit field within each halfword that will always be zero.

5.5.1.28 SC.OSW (Swap word from 10 to 11 forma

- SC.OSW (Swap word from 10 to 11 format)

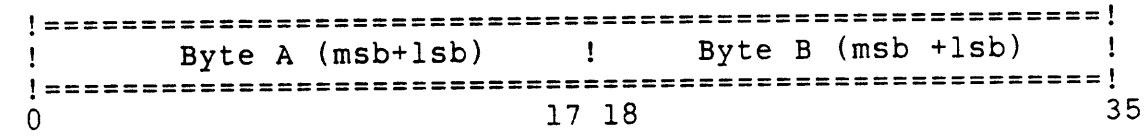
This routine swaps a single word from PDP-10 format to PDP-11 format. Rather than be yet another in the long list of byte swappers, this routine is globally available to the rest of the monitor (in particular SYSAPs).

Call

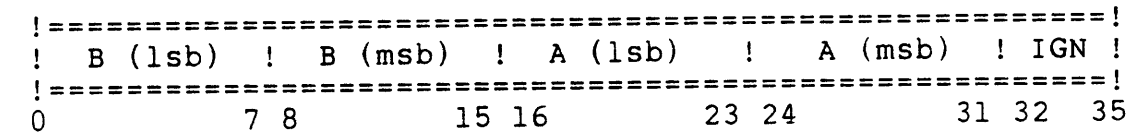
T1/ Word to swap

Return (+1) Always
T1/ Byte swapped word

Input format:
2 18 bit half words,



Output format:
2 16 bit half words, left justified



Where A and B are the half words, and (lsb) are the least significant bits and (msb) are the most significant bits.

5.5.1.29 Return node number given CID (SC.SBI)

- Return node number given CID (SC.SBI) -

This routine returns the node number of the remote end of the given CID.

Call
BLCAL. (SC.SBI,<SS.CID>)

Where:
SS.CID -- Connect ID of target connection

Return (+1) Always
T1/ Local connect ID
T2/ Node number of the remote end of the connection specified

5.5.1.30 Return credit info (SC.RAC)

- Return credit info (SC.RAC) -

This routine returns the current credit levels for a particular connection. This routine is not interlocked, hence the credit counts may be incorrect by the time SCAMPI returns from the PUSHJ. The usual case is SCAMPI is interrupted in the middle of this routine by a message arriving for this connection. This means the receive credits will be wrong. If the SYSAP sends a reply at interrupt level the send credits will be wrong as well. Since most connections maintain a large number of credits this routine will at least give a ballpark figure. If the SYSAP knows that it will not be getting any messages or if it calls this routine CIOFF then the counts will be correct.

```
Call
BLCAL. (SC.RAC,<SS.CID>)

Where:
SS.CID -- Connect ID of connection whose credits
                are desired

Return (+1)
T1/     Error code

Return (+2)
T1/     Send credit
T2/     Receive credit
T3/     Number of datagram buffers queued
```

5.5.1.31 Return local port number (SC.PRT)

- Return local port number (SC.PRT) -

This routine returns the node number of the local node. The assumption is made that this is a KL with only one KLIPA.

Call
No arguments

Return (+1)
Error code

Return (+2)
T1/ Local node number
T2/ Reserved
T3/ Reserved
T4/ Reserved

5.5.1.32 Allocate a datagram buffer (SC.ALD)

- Allocate a datagram buffer (SC.ALD) -

This routine will allocate datagram buffers from the SCA free pool. It is from this pool that all buffers on the port free queues originated. If desired the SYSAP can use buffers from this pool for outgoing datagrams and have the buffer placed on the port free queue on transmission completion.

When more than one buffer has been requested, the buffers are chained together with the link in the zeroth word of each buffer. The link word always points to word zero of the next buffer. The last buffer has a link word of zero.

```
Call
T1/     Number of buffers desired

Return (+1)
T1/     Error code

Return (+2)
T1/     Address of first buffer on chain
T2/     Number of buffers returned
T3/     Address of routine to return buffers
```

5.5.1.33 Allocate a message buffer (SC.ABF)

- Allocate a message buffer (SC.ABF) -

This routine allocates message buffers from the SCA free pool. If desired the SYSAP may use these buffers for message transmission and have the buffer placed on the port free queue on send completion. Also note that these buffers have the invisible area described in the section on SCA buffers.

When more than one buffer has been requested, the buffers are chained together with the link in the zeroth word of each buffer. The link word always points to the zeroth word of the next buffer. The last buffer has a link word of zero.

```
Call
T1/     Number of buffers desired

Return (+1)
T1/     Error code (No buffers returned)

Return (+2)
T1/     Address of first buffer on chain
T2/     Number of buffers returned
T3/     Address of routine to return buffers
```

5.5.1.34 Return a message buffer (SC.RBF)

- Return a message buffer (SC.RBF) -

This routine is used to return buffers to the SCA free pool allocated with SC.ABF. Although buffers can be allocated in chains, they may not be returned that way. A buffer is returned one at a time. Hence the link word of the buffer is ignored.

Call
T1/ Address of buffer to be returned

Return (+1) Always
No data returned, buffer returned OK

5.5.1.35 Return a datagram buffer (SC.RLD)

- Return a datagram buffer (SC.RLD) -

Datagram buffer allocated with SC.ALD must be returned to the SCA free pool with this routine. Although buffers may be allocated in chains, they must be returned one at a time. Hence the link word in each buffer is ignored.

Call
T1/ Address of buffer to be returned

Return (+1) Always
No data returned, buffer has been returned
to SCA free pool

5.5.2 SCA to SYSAP interface

The following is the complete set of callbacks that the SYSAP can expect to see from SCA.

General format of returned information:

```
        T1/     Function code
        T2-T4/  Function code dependant additional data

        Context: List of possible contexts for this callback. I.E.
                        the contexts the SYSAP may see during this callbac
```

- - - - - - - - - -

.SSDGR
Datagram received. SCA is indicating that the CI has filled one of the buf queued for datagram reception with a datagram.

```
        T1/     .SSDGR
        T2/     Connect ID
        T3/     Address of datagram buffer
        T4/     <FLAGS>B5 ! <Addr of routine to return buffer>B35
                (See SC.SDG for flag definitions)

 .MHPKL word of datagram buffer/
                Length of the packet (Bytes for industry compatable, words
                                        for high density)

        Context: Interrupt
```

- - - - - - - - - -

.SSMGR
Message received. SCA is indicateing that you have received a message from CI.

```
        T1/     .SSMGR
        T2/     Connect ID
        T3/     Address of message buffer
        T4/     <FLAGS>B5 ! <Addr of routine to return buffer>B35
                (See SC.SDG for flag definitions)

 .MHPKL word of msg buffer/
                Length of the packet (Bytes for industry compatable, words
                                        for high density)

        Context: Interrupt
```

- - - - - - - - - -

.SSPBC
Port broke connection. The port hardware detected a fatal error for the port virtual circuit on which you had a connection. Thus it broke your connection.

```
        T1/     .SSPBC
        T2/     Connect ID
        T3/     Unused
```

T4/ Unused

Context: Interrupt, scheduler

- - - - - - - - - -

.SSCTL
Connection to listen. The listen done for you has been matched to a remote connect request.
T1/ .SSCTL
T2/ Connect ID
T3/ Address to connection data from remote system
T4/ Unsued

Context: Interrupt

- - - - - - - - - -

.SSCRA
Connection response available. The system to which you sent a connect messa has responded.
T1/ .SSCRA
T2/ Connect ID
T3/ -1 for accepted, 0 for rejected connection
T4/ Reject code (If rejected), pointer to connect data if accep

Context: Interrupt

- - - - - - - - - -

.SSMSC
Message/datagram Send complete. The message/datagram which you requested be sent, has been. The "buffer address" is the address of the buffer you gave SCA to send.
T1/ .SSMSC
T2/ Connect ID
T3/ Address of buffer
T4/ Length of buffer in bytes for ind. compat/words for high de
This is the length of the text portion, and does no include SCA headers.

Context :Interrupt

- - - - - - - - - -

.SSLCL
Little credit left. You have just received a message that put you under receive credit threshold. It is suggested that you queue at least some buff if you expect to get more messages. Note that you will receive one of these calls every time you receive a message after which you are under threshold.
T1/ .SSLCL
T2/ Connect ID
T3/ Number of credits needed to get you over threshold
T4/ Unused

Context: Interrupt

- - - - - - - - - -

.SSNCO
Node came online. You requested to be told of nodes coming online. It happened and now you're being told.

T1/ .SSNCO
T2/ Node number of system that has come online
T3/ Unused
T4/ Unused

Context: Interrupt, scheduler

- - - - - - - - - -

.SSOSD
OK to send data. The connection has been completed to a remote system and now in the OPEN state. Messages and datagram may be sent.

T1/ .SSOSD
T2/ Connect ID
T3/ Unused
T4/ Unused

Context: Interrupt

- - - - - - - - - -

.SSRID
Remote initiated disconnect. The host at the other end of your connection doesn't want to talk to you anymore and has completed an orderly shutdown your connection.

T1/ .SSRID
T2/ Connect ID
T3/ Unused
T4/ Unused

Context: Interrupt

- - - - - - - - - -

.SSCIA
Credit is available. Your message send failed for lack of credit. There is now more credit available for your send.

T1/ .SSCIA
T2/ Connect ID
T3/ Current send credit
T4/ Current receive credit

Context: Interrupt

- - - - - - - - - -

.SSNWO
Node went offline. The remote end of your connection has just dropped offline. After this callback, all data about the connection will be deleted.

T1/ .SSNWO
T2/ CID for connect you had on offline node
T3/ Unused
T4/ Unused

Context: Interrupt, scheduler

- - - - - - - - - -

.SSDMA
Named buffer operation complete. The named buffer operation you requested has been completed. Note that you will NOT be told about the completion of passive requests. I.E. you will not be notified when a request/send data done by a remote completes.

T1/ .SSDMA
T2/ Connect ID of connection tranfers was done for
T3/ Buffer name of named buffer
T4/ Unused

Context: Interrupt

- - - - - - - - - -

.SSDDG
Dropped datagram. SCA received a datagram for this connection and dropped since you had no buffers left. The buffer was returned to the port free que Note that this is a software detected datagram drop. If the port drops the datagram because there are no buffers on the port queue, this call back wil not happen.

T1/ .SSDDG
T2/ Connect ID
T3/ Unused
T4/ Unused

Context: Interrupt

- - - - - - - - - -

.SSMNT
Maintainance data send/receive complete.

T1/ .SSMNT
T2/ Buffer name of completed transfer
T3/ Unused
T4/ Unused

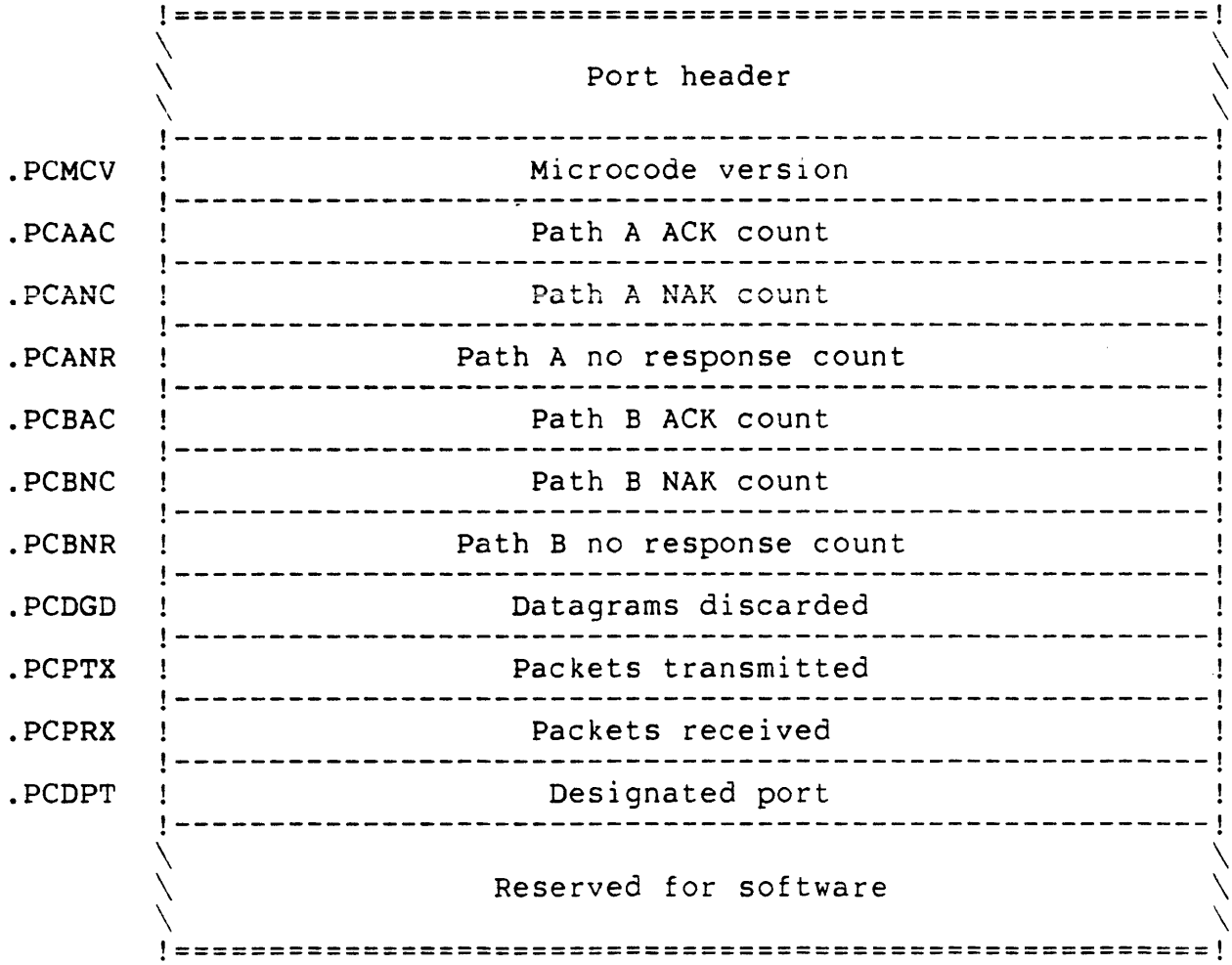
Context: Interrupt

- - - - - - - - - -

.SSRPC
Read port counters response. The request for reading the port counters has completed and here is your answer.

T1/ .SSRPC
T2/ Connect ID of requesting connect
T3/ Address of packet containing counter data
T4/ Unused

The following is the format of the counter data:



6.0 Writing a JSYS SYSAP

This section is a detailed review of the differences between writing monitor level SYSAPs and user level SYSAPs. The concepts are identical for the two SYSAP types, but the implementation details are very different. In the reading of the following sections it is assumed that the sections on monitor level SYSAPs has been read.

6.1 General rules

6.1.1 Communicating with SCA

When writing monitor level code SYSAPs communicate with SCA through subroutine calls. The SYSAP obtains services from SCA by calling routines in SCA, and SCA returns event information by calling routine addresses specified by the SYSAP. When writing JSYS code, services are obtained by executing the JSYS, and SCA communicates event information through a series of PSIs. In all there are four PSIs that the JSYS level SYSAP may have to handle. One each for incoming messages, datagrams, named buffer completions and all other events. If the SYSAP does not require some of these services then the PSI need not be enabled.

6.1.1.1 Using the SCS% with the PSI system

When using the PSI system with SCA keep in mind that the SCS% call to add channel must be done in addition to the usual set of JSYS calls. The SCS% call simply lets SCA know which channels are to be used for which event types. The SIR%, AIC%, EIR% (and their extended third cousins) etc. must still be used to specify and enable the PSI system.

6.1.2 Buffer management

Buffer management for the JSYS SYSAP is very different than that done for monitor SYSAPs. There are still four basic buffer types, messages and datagrams, both of which can be incoming or outgoing.

6.1.2.1 Incoming buffers

Buffers queued for reception of messages and datagrams must be large enough to handle the largest packet possible. The correct buffer size for messages is thirty eight (38) words, for datagrams this size is one hundred fifty two (152) words. These buffers are queued in chains much the way they are in the monitor. These buffers are not placed on the port queues however. Monitor internal buffers are placed on the port queues and data is BLTed to user space when it arrives. When data is moved the length to be BLTed is obtained from the packet. Hence if the

buffer is to short the end of the buffer will be BLTed over. This is why the buffer must be at least as long as the longest packet possible.

6.1.2.2 Outgoing buffers

Buffers used for packets transmission need only be as long as the packet text. There are no other restrictions on these buffers.

6.1.3 The race condition

Something which comes up when coding a monitor SYSAP is that interlocking further CI traffic is easy. JSYS SYSAPs do not have things quite so simple. In particular there is a race that a JSYS SYSAP designer should be aware of. When a JSYS SYSAP does a listen, he is told about a connection to that listen with a PSI. As soon as the connection comes in and the match is found, the monitor changes the state of the connect to connectreceived rather than listen. Hence if another connect request comes in for that SYSAP before it has been told about the first connect request, then the second one will fail at the SCA level since no match can be found even though the SYSAP would likely want to see that connect.

One reasonable solution to this problem is based on an understanding of the problem by both the active and passive sides of the connect mechanism. The listener should open more than one listen (keeping in mind that they do use system resources), and the connector should retry connect attempts if they fail.

6.2 JSYS SYSAP interface

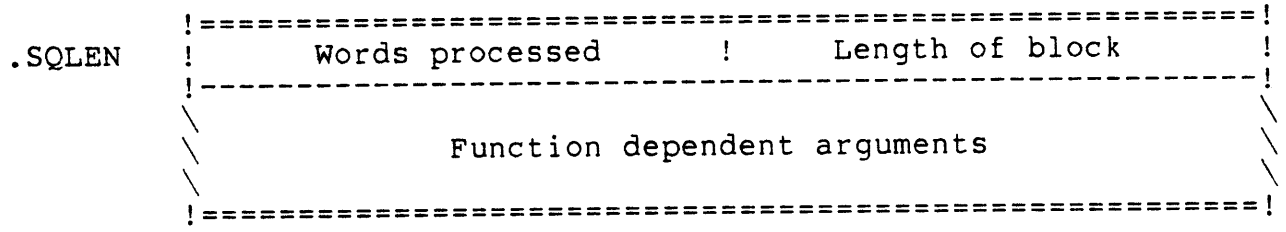
The following is the general format of a call to the SCS% JSYS.

```
Call
T1/     Function code
T2/     Address of arg block


Return (+1) Always
T1/     Function code
T2/     Address of arg block
```

An error generates an illegal instruction trap.

General format of the argument block:



Where the "words processed" is the number of words in the argument block that the monitor actually looked at and used. Hence this field is returned by the monitor, not supplied by the user. The "length of block" is the total length of the supplied block including the header word.

Note that this JSYS requires one or any combination of wheel, maintenance, or network wizard.

6.2.1 Connect (.SSCON)

Connect (.SSCON)

This function requests a connection with another process on the CI. The JSYS will return as soon as the connection request has been sent. You are notified that your connection request was granted or failed via a PSI interrupt.

The argument block has the following format:

.SQLEN	Words processed	Length of block
.SQSPN	Byte pointer to source process name	
.SQDPN	Byte pointer to destination process name	
.SQSYS	Node # of destination	SYSAP field for CID
.SQCDT	Pointer to connection data	
.SQAMC	Address of first buffer on message buffer chain	
.SQADC	Address of first buffer on datagram buffer chain	
.SQRCI	Returned connect ID	

The byte pointer to the source process name is a byte pointer to the ASCII string which is the name of your process. Note that this string must end on a null byte and may be a maximum of 16 bytes long, not including the null byte.

The byte pointer to destination process name is the byte pointer to the name of the process you wish to connect to. This name must also end in a null byte and may be a maximum of 16 bytes, not including the null byte.

The node number is the unique identifier of the system you wish to connect to.

Note that the specified strings may be any valid ASCII byte size (I.E. any byte size equal to or larger than 7 bits). These strings may also be generic byte pointers (-1,,STRNG). If generic byte pointers are given, 7 bit ASCII terminated by a null byte is assumed.

The "SYSAP field for CID" is the right justified value to be placed into the SYSAP field of the connect ID created for the connection. These bits are generally used for connection management by the user program.

The pointer to connection data is the address of a block of data SQ%CDT words long to be sent as connection data. Note that the monitor will copy SQ%CDT words of data from the users address space. Hence a full block SQ%CDT words long should be allocated for the data.

- Error codes -

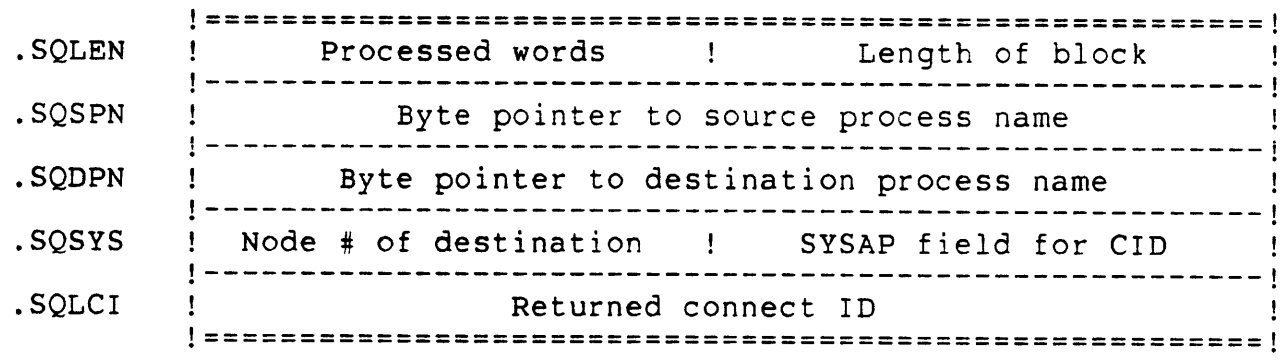
SCSNEP,SCSBTS,SCSISB,SCSNSN,SCSENB,SCSIAB,SCSIBP,SCSNBA

6.2.2 Listen (.SSLIS)

Listen (.SSLIS)

This function listens for a connection. Note that the JSYS does not block. You will be notified of a connect to your listen via a PSI interrupt.

There are a number of options that may be used when doing a listen for a connection. You may listen for a particular process from any system by making the node number -1. You may listen for any process from a particular system by making the byte pointer to the destination process name -1. If the node number and the byte pointer to the destination process name are both -1, then any connect request that is not for a particular process name will match your listen. Naturally you may specify both a node number and a process name and then only that process from the named system will be allowed to connect to your listen.



The fields defined here are identical to those used by the connect call. The only difference is the absence of some of the fields used in the connect call.

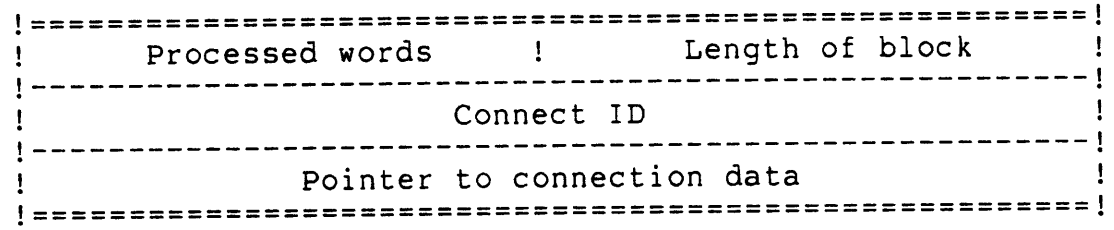
- Error codes -

SCSNBA,SCSNEP,SCSBTS,SCSISB,SCSNSN,SCSENB

6.2.3 Accept (.SSACC)

Accept a connection (.SSACC)

This function tells a remote process that you are granting his request for a connection.



The pointer to connection data is the address of the block of connection data to be sent to the remote system. The monitor will copy SQ%CDT words of data from the users address space as data. Note that this data will be sent directly over the CI and hence the low order four bits are not sent.

- Error codes -

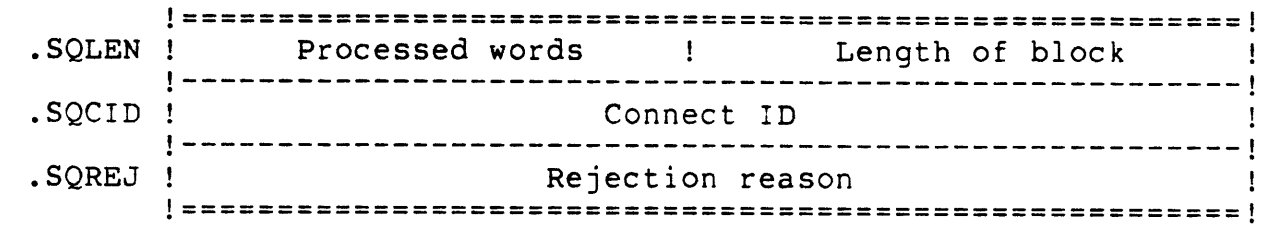
SCSNEP,SCSBTS,SCSIID,SCSCWS,SCSNBA

6.2.4 Reject (.SSREJ)

Reject a connection (.SSREJ)

This function code tells a remote process that you are not allowing a connection to your process.

The argument block for this function has this format:



The rejection reason is a code, invented by the SYSAP, (outside the revserved range /TBD/ codes) which indicates why the connection was rejected.

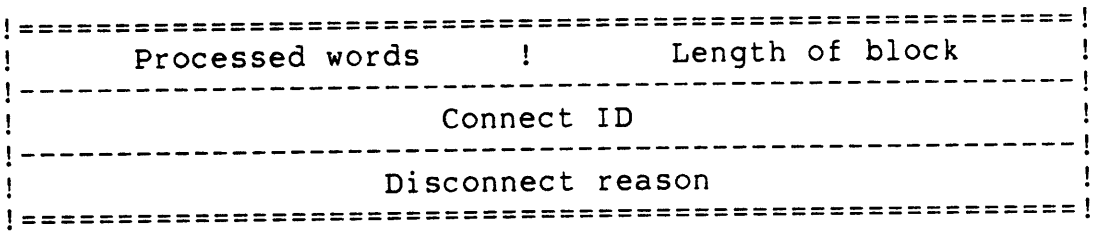
- Error codes -

SCSBTS,SCSIID,SCSNBA,SCSCWS,SCSNEP

6.2.5 Disconnect (.SSDIS)

Disconnect (.SSDIS)

This function closes a connection. Note that the connection must be open to use disconnect. The disconnect function code requires the following arguments:



The disconnect reason is a SYSAP invented code (outside the reserved range /TBD/) which indicates why the connection was disconnected.

- Error codes -

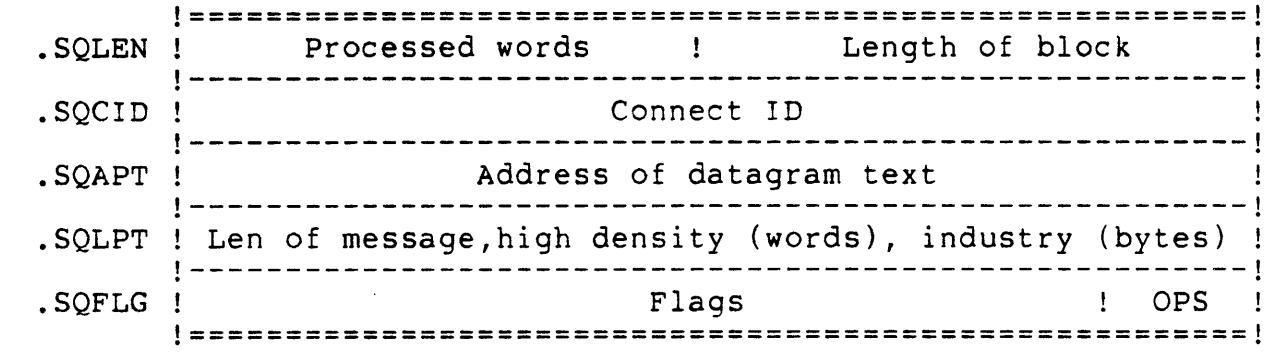
SCSNEP,SCSBTS,SCSIID,SCSCWS,SCSNBA

6.2.6 Send a DG (.SSSDG)

Send a datagram (.SSSDG)

This function code sends a datagram.

The following arguments are required:



OPS is the path selection field. OPS allows the JSYS user to select the path over which the packet is to be sent. The following setting are allowed for OPS:

0 --> Automatic path selection
1 --> Use path A
2 --> Use path B

If automatic path selection is chosen, the port hardware selects the path to use.

The currently defined flags are:

SC%MOD --> Packet transmission mode: 0 = industry compatable
1 = high density

*** Restrictions ***

1. If the datagram is to be sent in industry compatable mode, the text must be packed in left justified, word aligned, eight bit bytes.

2. The datagram text may not be longer than 152 (decimal) words/608 bytes long. The byte count is for industry compatable packets and the word count for high density packets.

- Error codes -

SCSNEP,SCSBTS,SCSIID,SCSDCB,SCSIAA,SCSNBA,SCSCWS,SCAMTL

6.2.7 Queue a DG buffer (.SSQRD)

Queue a buffer for datagram reception (.SSQRD)

This function code queues buffers for datagram reception. Note that in each buffer the first word is the address of the next buffer. The last buffer has zero as the address of the next buffer. This function requires these arguments:

.SQLEN	Processed words	Length of block
.SQCID	Connect ID	
.SQAFB	Address of first buffer on buffer chain	

*** Restrictions ***

1. Datagram buffers must be 152 words in length.

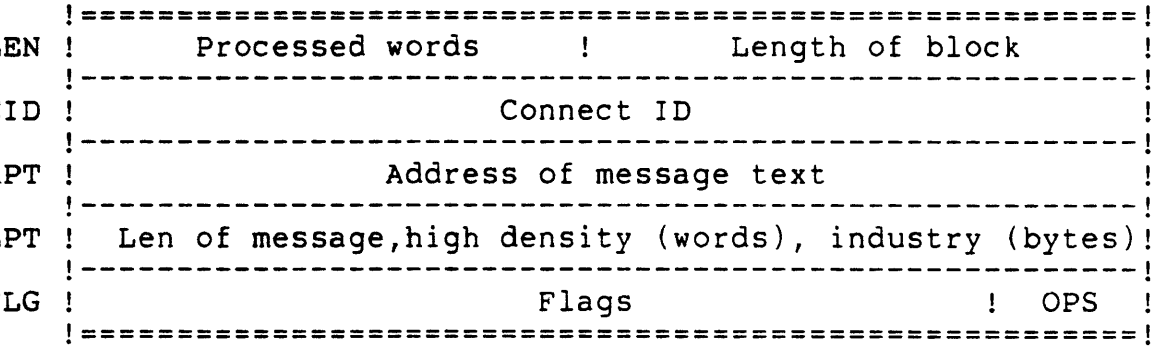
- Error codes -

SCSNEP,SCSIID,SCSNBA,SCSBTS

6.2.8 Send message (.SSSMG)

Send a message (.SSSMG)

This function code sends a message to a remote node.



OPS is the path selection field. OPS allows the JSYS user to select the path over which the packet is to be sent. The following setting are allowed for OPS:

0 --> Automatic path selection
1 --> Use path A
2 --> Use path B

If automatic path selection is chosen, the port hardware selects the path to use.

The flags definitions may be found in the section for .SSSDG.

*** Restrictions ***

1. If this is an industry compatable message then the text must be packed in left justified 8 bit bytes.

2. Message text may not be longer than 38 36 bit words (152 bytes).

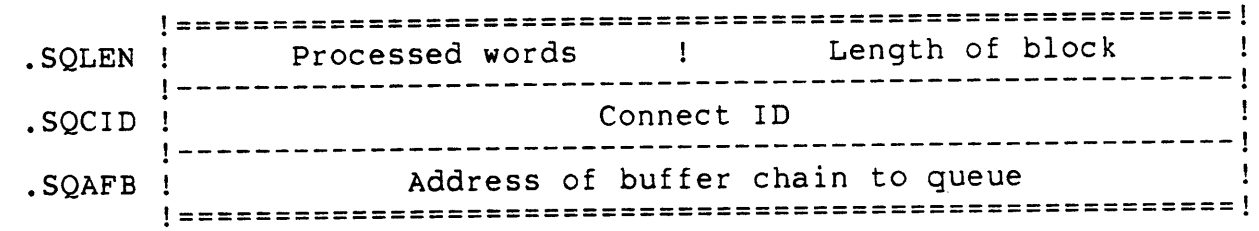
- Error codes -

SCSIID,SCSNEP,SCSNBA,SCSBTS,SCSNSH,SCSDCB,SCSIAA

6.2.9 Queue message receive buffers (.SSQRM)

Queue message receive buffers (.SSQRM)

This function code queues buffers to receive messages. Note that the buffer size is fixed at 38, 36 bit words. It requires the following arguments:



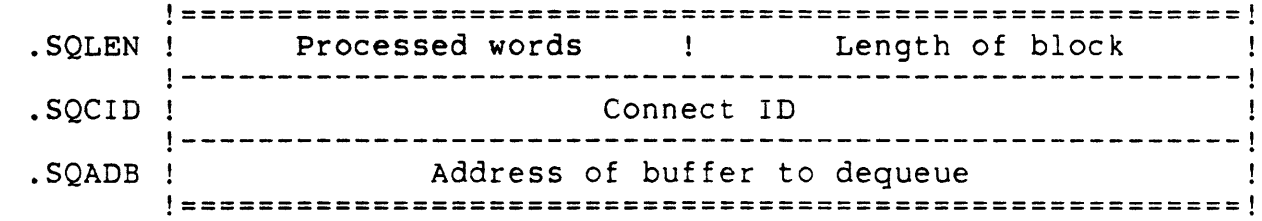
- Error codes -

SCSNEP,SCSIID,SCSNBA,SCSBTS

6.2.10 Cancel DG receive (.SSCRD)

Cancel datagram receive (.SSCRD)

This function code removes a buffer queued for datagram reception. The address that you specify must be the address of a buffer that was previously queued for receiving datagrams (with .SSQRD). If this address is not found by the monitor when it goes to take the buffer out of its queue, the JSYS fails with an illegal instruction trap. This function requires these arguments:



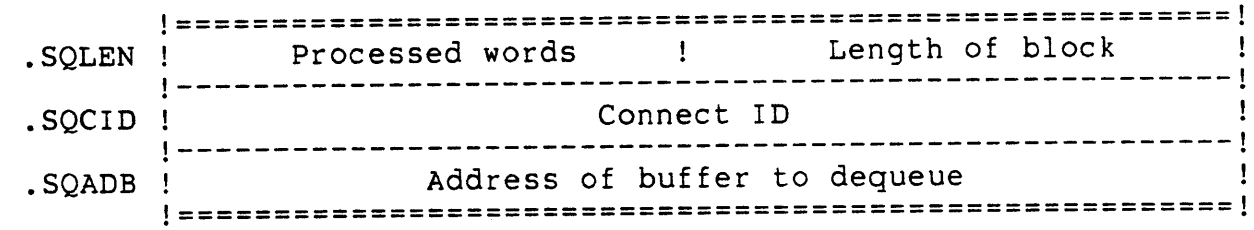
- Error codes -

SCSNSC,SCSNEP,SCSBTS,SCSNSB,SCSNEB

6.2.11 Cancel receive message (.SSCRM)

Cancel receive message (.SSCRM)

The function dequeues a buffer that was queued for message reception. The buffer address that is specified must be of a buffer that you asked to be queued for message reception (with .SSQRM). If the monitor does not find the address specified in the argument block amongst the buffers you queued, the JSYS will fail. This function requires the following arguments:



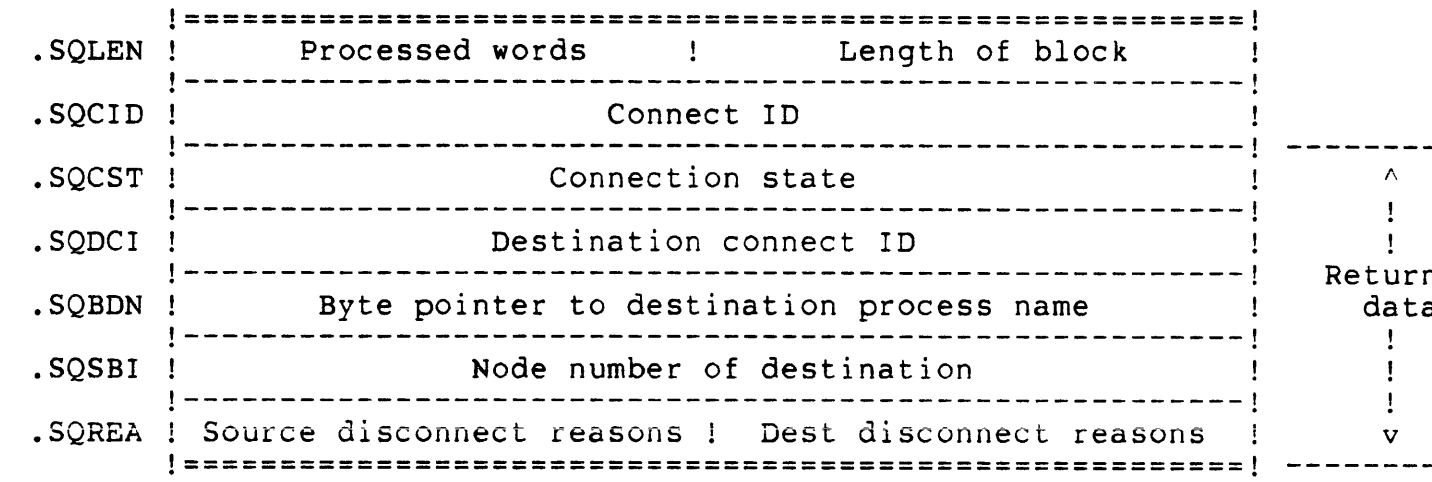
- Error codes -

SCSNSC,SCSNEP,SCSBTS,SCSNSB,SCSNEB

6.2.12 Connect state poll (.SSCSP)

Connect state poll (.SSCSP)

This function returns information about the state of a connection. The argument block to this function includes space for the returned data. The following is the format or the argument block:



Note that the byte pointer to destination process name must be provided by the caller and may be either a real byte pointer or minus one in the left half and the base address of the string in the right half. If the generic byte pointer is used the monitor assumes the string is 7 bit ASCII terminted with a null byte, or the 16th chatacter, whichever comes first.

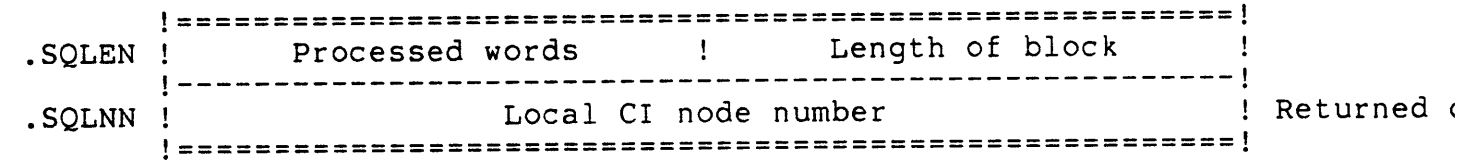
- Error codes -

SCSNEP,SCSBTS,SCSIID

6.2.13 Return local node number (.SSGLN)

Return local node number (.SSGLN)

This function returns the local node number. If the machine does not have a KLIPA, then an illegal instruction is generated. The following is the argument block format:



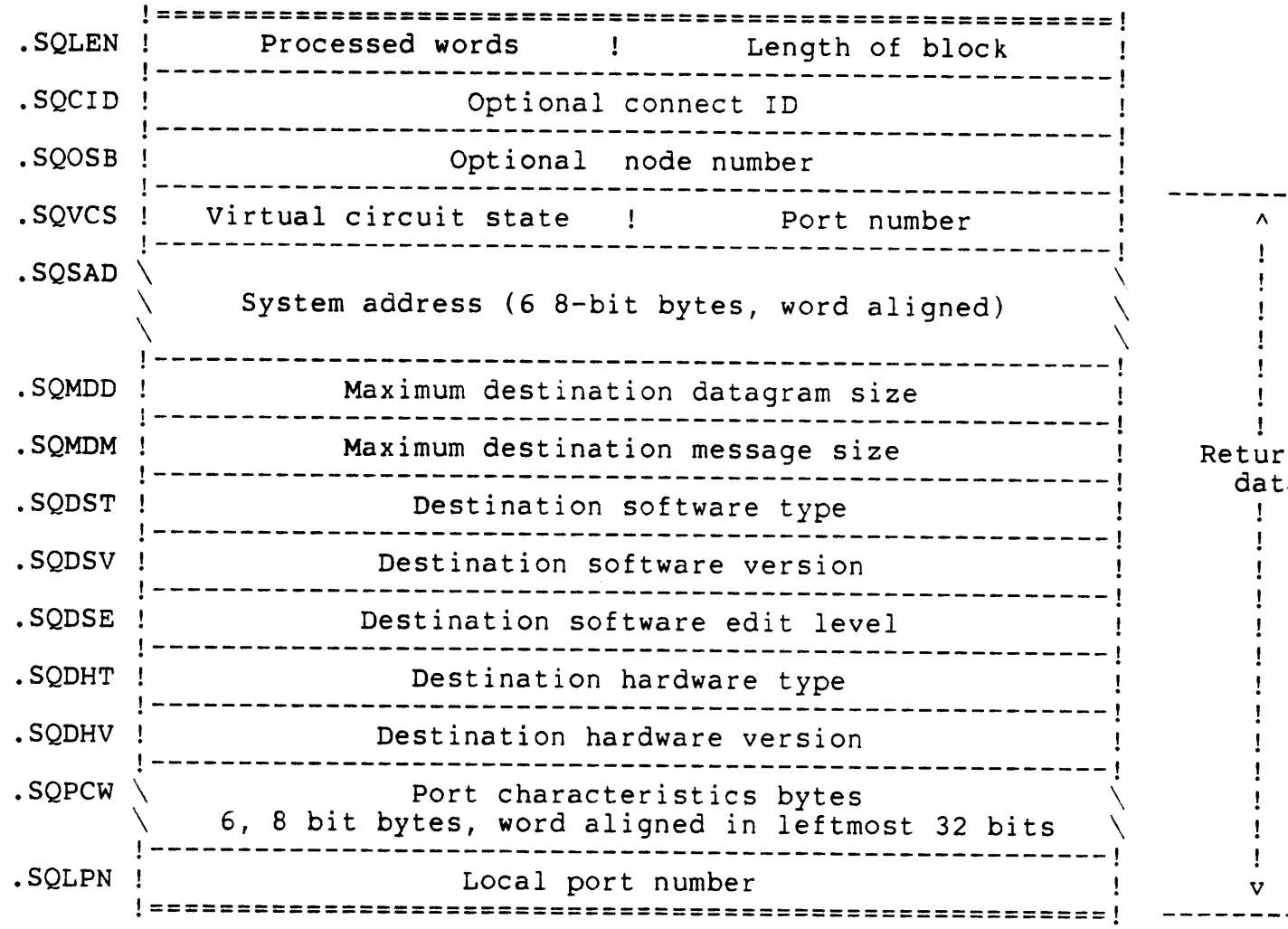
- Error codes -

SCSBTS,SCSNKP

6.2.14 Return configuration data (.SSRCD)

Return configuration data (.SSRCD)

This function returns data about another system. Given the node number the monitor will return data about that system. The data is returned in the block pointed to by T1 which looks like this:



The connect ID and node number are optional in that one or the other must be specified, but not both. If the connect ID field is non-zero then the node number is ignored and the information returned is for the system implied by the connect ID. If the connect ID field is zero then the node number is used. Note that this allows the use of node zero.

If the local port number is not available, -1 will be returned to the user in offset .SQLPN.

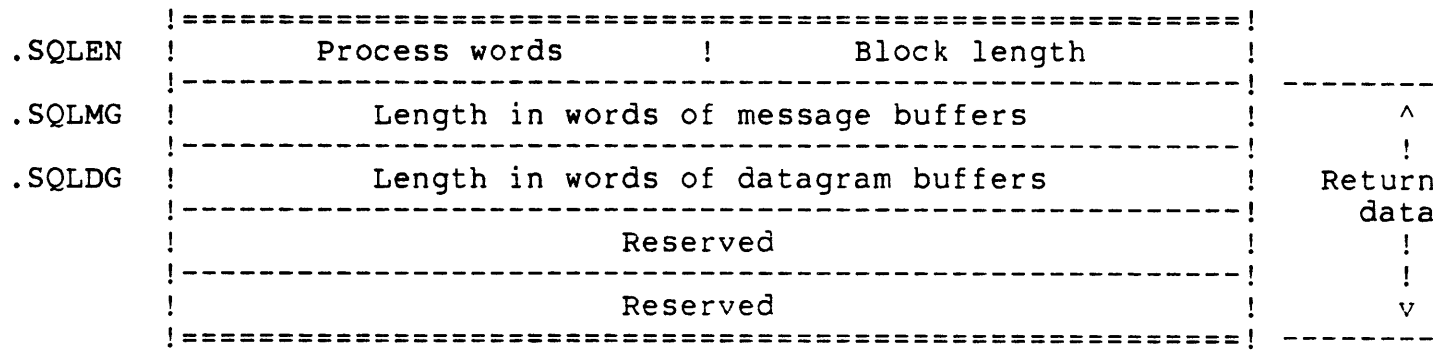
- Error codes -

SCSNEP,SCSBTS,SCSIID,SCSISB

6.2.15 Return buffer sizes (.SSRBS)

Return buffer sizes (.SSRBS)

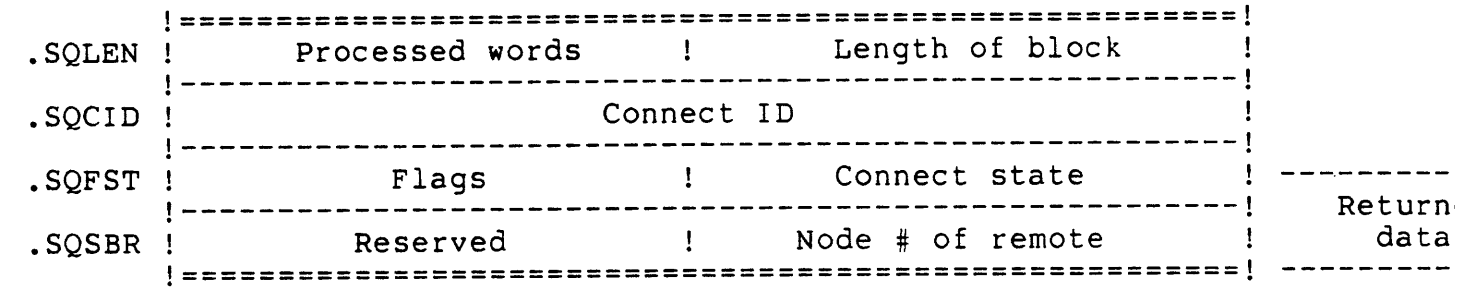
This function returns the sizes of the various buffers SCA expects to see. In particular it returns the size of message and datagram buffers. It requires the following arguments:



6.2.16 Return status information (.SSSTS)

Return status information (.SSSTS)

This function returns status information about a connection. It is intended as a fast form of the .SSCSP function code which returns detailed information about a connection. This function requires the following arguments:



The flags which appear in the flags field are:

1. Message available (SQ%MGA)- There is at least one message available for this connection.

2. Datagram available (SQ%DGA) - There is at least one datagram available for this connection.

3. Named buffer transfer complete (SQ%DMA) - At least one named buffer transfer has completed since the last time the SCS% JSYS was performed with this function code. The bit is cleared in the monitor data base when this function code is performed.

- Error codes -

SCSBTS,SCSNEP,SCSIID

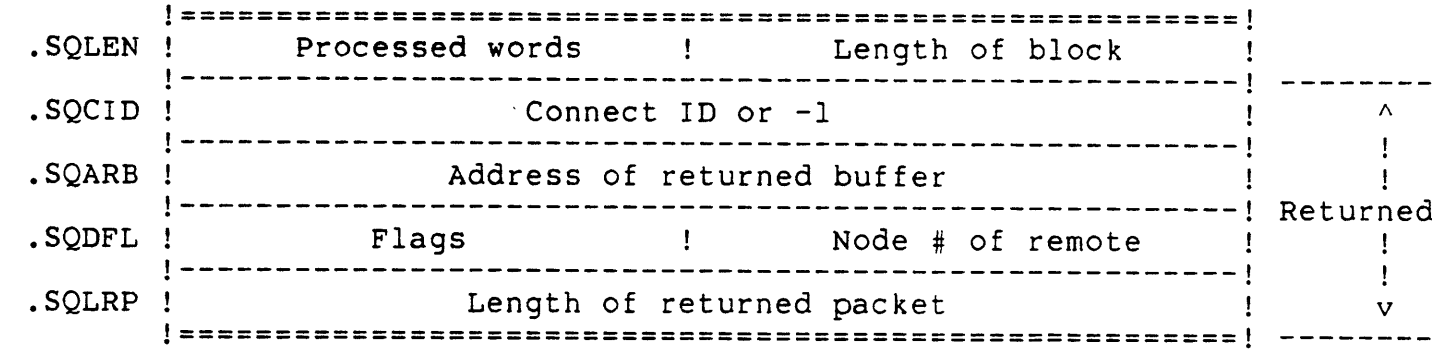
6.2.17 Get a queue entry

There are four functions for removing things from queues. One each for messages, datagrams, named buffer transfer competions, and "all other events". In each case the monitor will issure a PSI for the appropriate fork when when of these queues goes non-empty. The monitor will not issue a PSI for each entry. It is the responsibility of the SYSAP to empty the queue once it has seen the PSI.

6.2.17.1 Receive a message (.SSRMG)

Receive a message (.SSRMG)

This function code returns message text for either the calling fork or the specified connection. If the connect ID field in the argument block is minus one, then the first message found for the calling fork is returned. If the connect ID is anything but minus one (-1), the monitor will use this as a connect ID and return the first message found for that connection. Note that if there is no message available an illegal instruction is generated. The following argument block is required for this function code.



The address of returned buffer is the address at which the monitor has placed your data. This address is one of the addresses previously specifed by a .SSQRM call. If no .SSQRM call was done, then this function code will fail with an illegal instruction trap. If the address is not writable you will also get an illegal instruction trap.

The flag definitions may be found in the section on SC.SDG.

The length of the returned packet is in bytes for an industry compatible mode message, and words for a high density mode packet. It is always the length of the packet text, and does not include the SCA headers.

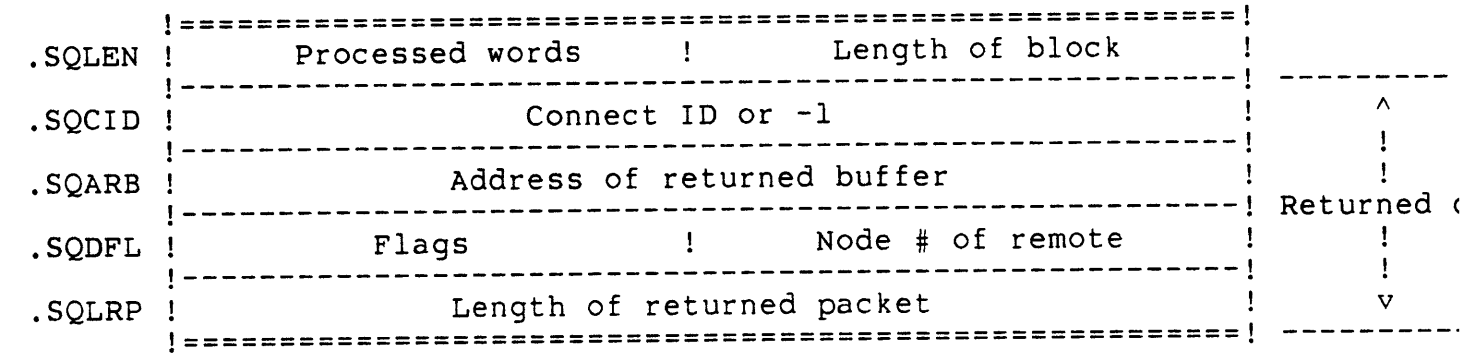
- Error codes -

SCSNEP,SCSBTS,SCSIID,SCSQIE,SCSNUB

6.2.17.2 Receive a datagram (.SSRDG)

Receive a datagram (.SSRDG)

This function code returns datagram text for either the calling fork or the specified connection. If the connect ID field in the argument block is minus one, then the first datagram found for the calling fork is returned. If the connect ID is anything but minus one (-1), the monitor will use this as a connect ID and return the first datagram found for that connection. This is the argument block for this function code:



The address of returned data is one of the addresses provided to the monitor on a .SSQRD call. If no .SSQRD was done or if the address is not writable, the JSYS will fail with an illegal instruction trap. If there are no datagrams available, an illegal instruction is generated.

The flags are returned by the monitor and currently only indicate the data packing mode.

The length of the returned packet is in words if this is a high density datagram, and bytes if an industry compatible datagram.

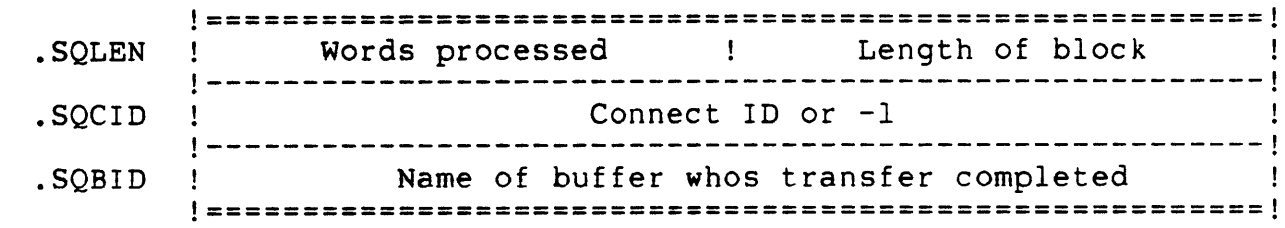
- Error codes -

SCSNEP,SCSBTS,SCSIID,SCSQIE,SCSNDQ,SCSNBA

6.2.17.3 Get entry from data queue (.SSGDE)

Get entry from data queue (.SSGDE)

This function code returns the first entry from the data request complete queue. The argument block follows:



The buffer name indicates which transfer has completed.

- Error codes -

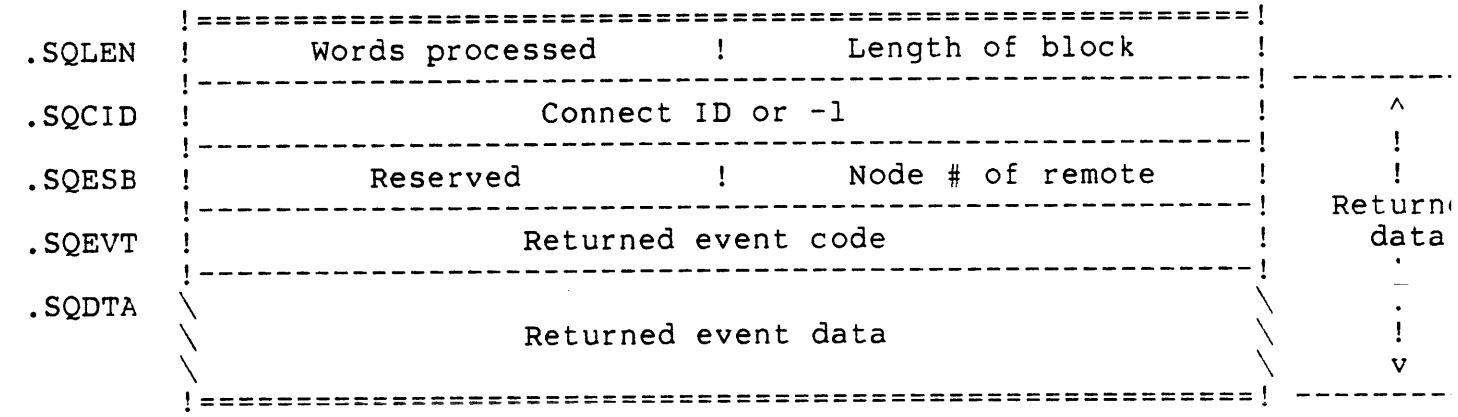
SCSNEP,SCSIID,SCSBTS,SCSIAB

6.2.17.4 Get an entry off the event queue (.SSEVT)

Get an entry off the event queue (.SSEVT)

This function code retrieves the first entry off the event queue. This queue is a record of events that the JSYS user is to be notified about. The user receives an interrupt on the first event. After this events will be added to the end of the queue with no further interrupts generated. It is the responsibility of the user to empty this queue upon receiving an event interrupt.

If a connect ID is specified, then the next event for that connection will be returned. If however the connect ID field is minus one, then the next event for the fork is returned.



The event code describes what event has occured. It is a small integer ranging from zero to a maximum value. Hence it can easily be used as an index into an event dispatch table. The connect ID tells you what connection this event is relevant to. The event data is a block of data returned for each event type.

The general format of the event code descriptions is as follows:
.SExxx -- Text

The .SExxx is the event code and text desribes the code

.SQDTA -- Text
This second line describes the format of the data provided for this event code, starting at word .SQDTA of the argument block.

.SEVCC -- Virtual circuit closure
.SQDTA -- Contains the pertinant node number

.SECTL -- Connect to listen
.SQDTA -- Four words of initial connection data from the remote.

.SECRA -- Connection was accepted

.SQDTA -- Four words of initial connection data from the remote.

.SECRR -- Connection was rejected
.SQDTA -- The reason code

.SEMSC -- Message/datagram send complete
.SQDTA -- address of sent buffer

.SELCL -- Little credit left
.SQDTA -- number of credits required to get you back over threshold

.SENWO -- Node went offline
.SQDTA -- Node number of system which went offline

.SENCO -- Node came online
.SQDTA -- Node number of system which came online

.SEOSD -- OK to send data
.SQDTA -- not used here

.SERID -- Remote initiated disconnect
.SQDTA -- not used here

.SEPBC -- Port broke connection
.SQDTA -- not used here

.SECIA -- Credit is available
.SQDTA -- unsed here

.SEMAX is defined to be equal to the largest event code possible.

- Error codes -

SCSNEP,SCSIID,SCSBTS,SCSIAB

6.2.18 Named buffer overview

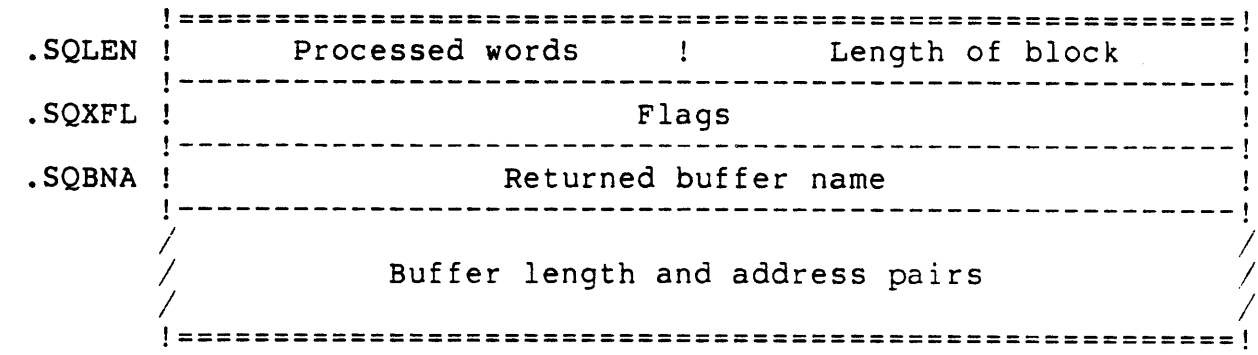
Named buffer overview

To send data to a remote system, the SYSAP must first declare memory to be part of a buffer. A buffer is made of segments. Each segment is a contiguous set of 36 bit words that DO NOT CROSS A PAGE BOUNDARY and are not more than one page long. Once the buffer has been established, the SYSAP must give the remote system the buffer name obtained. N.B. SCA does not give the remote the buffer name, it is the responsibility of the SYSAP to transmit this information.

6.2.19 Map a buffer (.SSMAP)

Map a buffer (.SSMAP)

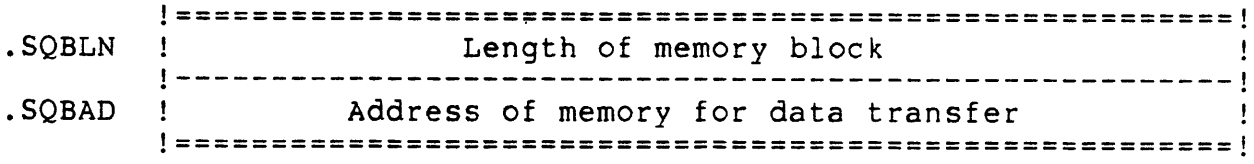
This function code associates a portion of memory with an SCA buffer name to be used in named buffer transfers. This function takes the following arguments:



The current set of flags are as follows:

1. SQ%DMD -- Named buffer mode. This two bit field indicates the desired setting of the mode bits for the buffer. The following settings are defined:
   SQ%DIC -- Industry compatable mode
   SQ%DCD -- Core dump
   SQ%DHD -- High density
   Any other value is illegal.

Buffer length and address pairs have the following format:



The length word in this block has one of two possible values based on the setting of the mode bits. If the buffer mode is core dump or industry compatable the length is in 8 bit bytes. If the mode is high density, the length is in 36 bit words.

The address for data transfer is the address where you expect data from a data transfer to be placed by the CI port microcode.

The buffer name is the descriptor by which all other references to this memory is made.

*** Restrictions ***

1. No buffer segment may cross a page boundary. Hence the longest possible buffer segment is one page.

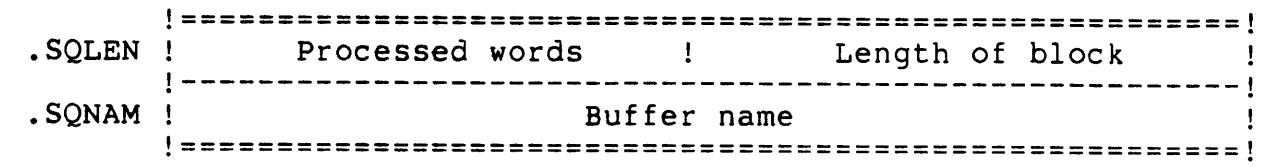
- Error codes -

SCSNEP,SCSBTS,SCSNBA,SCSIAB

6.2.20 Unmap a buffer (.SSUMP)

Unmap a buffer (.SSUMP)

This function will remove from the monitor data base (unmap) a memory block assigned for named buffer transfers. It requires these arguments:



- Error codes -

SCSNEP,SCSBTS

6.2.21 Send data (.SSSND)

Send data (.SSSND)

This function transfers data to a remote host. It is the responsibility of a higher level protocol to arrange the setup of buffer names. The call requires the following arguments:

.SQLEN	Processed words	Length of block
.SQCID	Connect ID for which transfer is to be done	
.SQSNM	Buffer name of send buffer	
.SQRNM	Buffer name of receive buffer	
.SQOFS	Xmit offset	Receive offset

- Error codes -

SCSNEP,SCSBTS,SCSIID

6.2.22 Request data (.SSREQ)

Request data (.SSREQ)

This function tells SCA and the port to get data for the given buffer name. The following arguments are required:

.SQLEN	Processed words	Length of block
.SQCID	Connect ID for which transfer is to be done	
.SQSNM	Buffer name of send buffer	
.SQRNM	Buffer name of receive buffer	
.SQOFS	Xmit offset	Receive offset

The "xmit offset" is the number of bytes/words the sending port should skip before sending data. This count is in bytes for industry compatable/core dump mode sends and words for high density sends

The "receive offset" is the number of bytes/words that the receiver should skip before writing data from the wire. The count is in bytes for industry comapatable/core dump and words for high density mode.

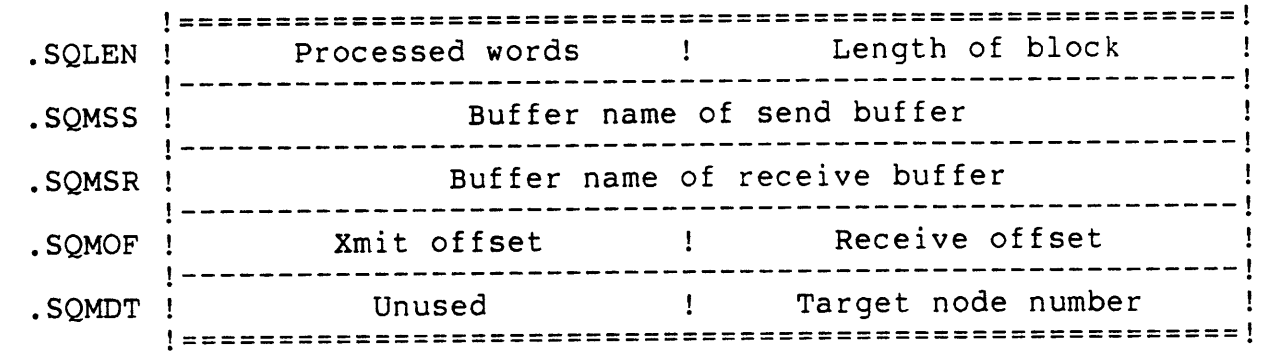
- Error codes -

SCSNEP,SCSIID,SCSBTS

6.2.23 Maintainance data send (.SSMDS)

Maintainance data read (.SSMDS)

This function requests a maintainance data send. It works much the way named buffer transfers do in that the .SSMAP call must be done first to set up a buffer. The buffer name returned by .SSMAP is used in this call. Note that the remote must provide the buffer name it has allocated for sending the data as well.

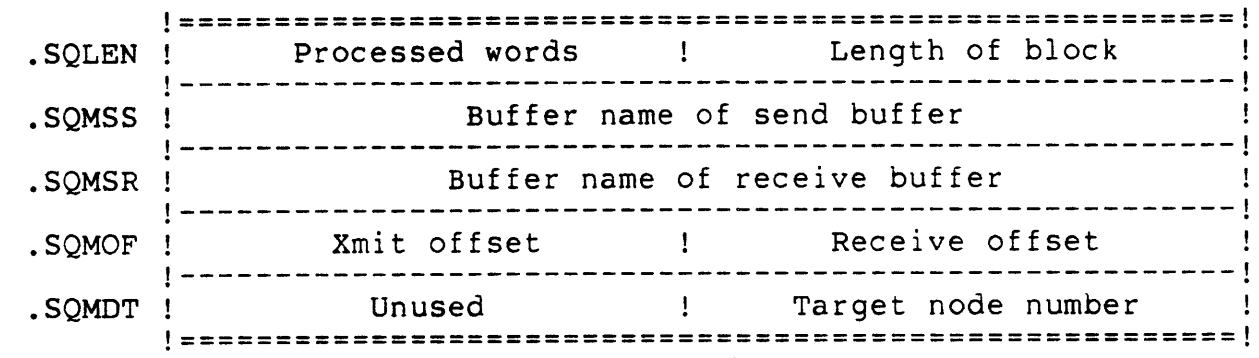


The xmit offset and receive offset are identical to those used for named buffer transfers.

6.2.24 Maintainace data read (.SSMDR)

Maintainance data read (.SSMDR)

This function requests a maintainance data read. It works much the way named buffer transfers do in that the .SSMAP call must be done first to set up a buffer. The buffer name returned by .SSMAP is used in this call. Note that the remote must provide the buffer name it has allocated for sending the data as well.

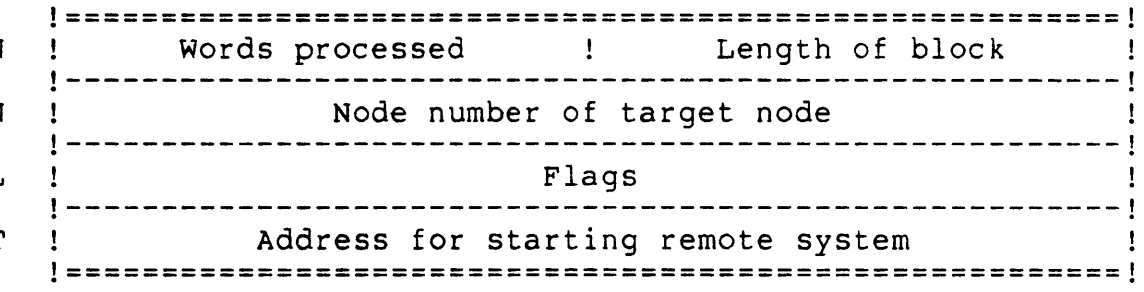


The xmit offset and receive offset are identical to those used for named buffer transfers.

6.2.25 Start a remote system (.SSSRS)

Start a remote system (.SSSRS)

This function will start a remote node. A start address may be specified or the default can be used.



The currently defined flags are:

1. SQ%DSA -- Use the default start address. If this bit is lit, the default address is used to start the target node. If the bit is off, the the address specified in word .SQAST of the argument block.

6.2.26 Reset a remote system (.SSRRS)

Reset a remote system (.SSRRS)

This function resets a remote node.

.SQLEN	Words processed	Length of block
.SQNTN	Node number of target node	
.SQRFL	Flags	

The currently defined flags are:
SQ%FRB -- Force reset. If this bit is not set, the reset is only done if the local host was the last host to send a reset to the target node. If the bit is on, the reset is done no matter who last reset the target node.

6.2.27 Set port counters (.SSSPC)

Set port counters (.SSSPC)

This function sets the port's performance counters. The following arguments are required:

.SQLEN	Number of words processed / Length of block	
.SQNOD	Node number	
.SQPCW	Port counter control word	
.SQCPC	Number of times the counters have been set/cleared	(Returne

Node number: Node number or 377 for all nodes

Port counter control word, bits as follows:

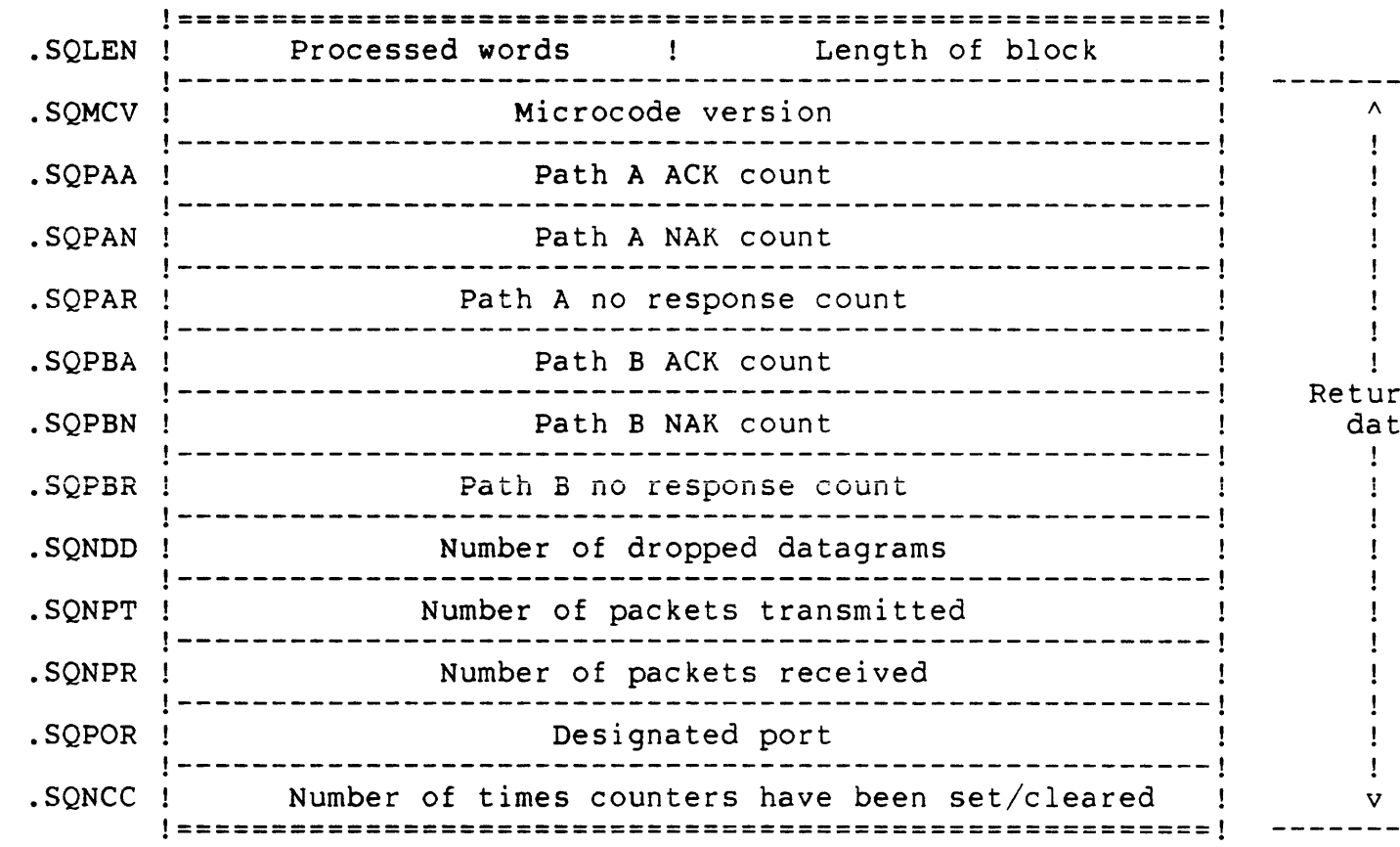
B0	If on, count ACKs received on Path A.
B1	If on, clear the counter.
B2	If on, count NAKs received on Path A.
B3	If on, clear the counter.
B4	If on, count NO_RSPs received on Path A.
B5	If on, clear the counter.
B6	If on, count ACKs received on Path B.
B7	If on, clear the counter.
B8	If on, count NAKs received on Path B.
B9	If on, clear the counter.
B10	If on, count NO_RSPs received on Path B.
B11	If on, clear the counter.
B12	The count of discarded datagrams because of no DGFree Queue entries.
B13	If on, clear the counter.
B14	Count the packets transmitted to the designated port.
B15	If on, clear the counter.
B16	Count the packets received from the designated port.
B17	If on, clear the counter.

The count of times the counters have been set/cleared reflects the number of SCS% JSYS that have been done to set/clear the port counters, not including the current one. I.E. the call just executed is not included in the returned count.

6.2.28 Read port counters (.SSRPC)

Read port counter (.SSRPC)

This function reads the maintenance counters out of the port hardware. The following argument block is used:



Only the function code is passed to the monitor. The rest of this block is information returned by the monitor.

- Error codes -

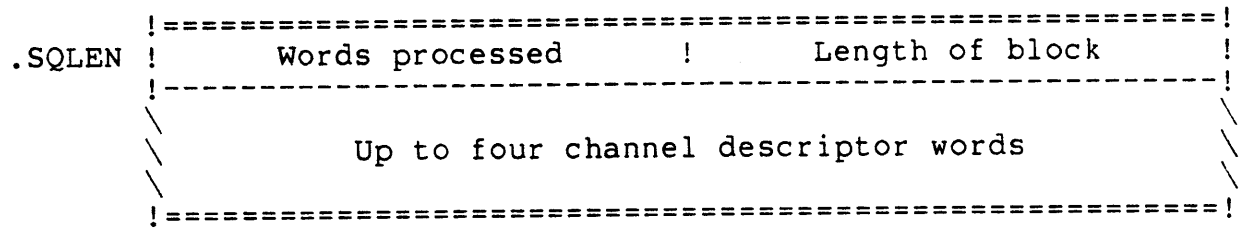
SCSNEP,SCSBTS,SCSNBA

6.2.29 Add interrupt channels (.SSAIC)

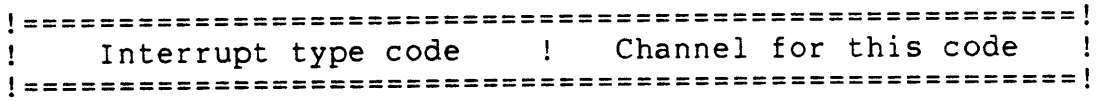
Add interrupt channels (.SSAIC)

All notifications from SCA happen on five PSI channels. These channels are set for: datagram available, message available, named buffer transfer complete, events, and configuration change. The events are detailed in the section on the .SSEVT funtion code. The only function performed by the .SSAIC code, is to inform SCA of which channels you wish to use for the variuos interrupts. It does not activate any of the channels, enable the PSI system, or set LEVTAB and CHNTAB for you.

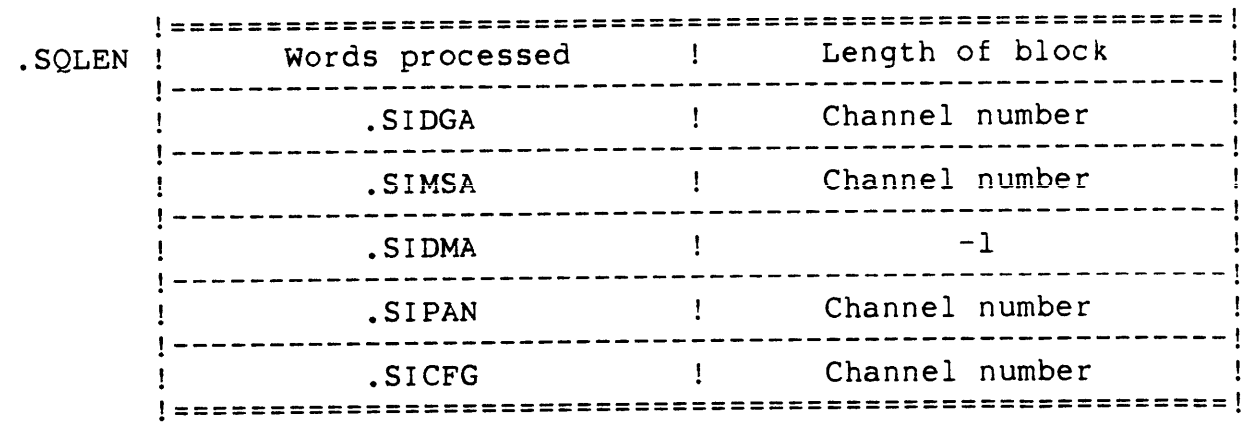
The argument block used to associate events with PSI channels has this format:



Where channel descriptor words have the following format:



The interrupt type code is a small integer that indicates an event type. The channel number field allows you to enable and disable interrupts for the given event type. If the channel number is -1 then interrupts are disabled. If it is an integer between zero and thirty five, then that channel will interrupt on the given event type. The next box is an example of what the maximum size block would look like.



In this example interrupts for message available, datagram available, all other events are being enabled. Interrupts on DMA transfer complete are being disabled. Note that only the first word of this block need

appear in this position. The channel descriptor words may appear in any order. There must be at least one but not more than four descriptor words in a block.

To set up the PSI system for use with SCA, the following JSYS's must be done:

1. SIR%/XSIR%

2. AIC%

3. EIR%

4. SCS% with the .SSAIC function code

- Error codes -

SCSBTS,SCSNEP,SCSNSP