NETWORK PL1 SUBPROGRAMS

April 21, 1971

Mark Krilanovich
Computer Research Lab
University of California
Santa Barbara, California

PREFACE

The subroutines described in this document have been implemented at UCSB and make the Network (via the NCP) accessible to PL1 programs executing in the 360/75. They are callable from within any PL1 main program or subroutine compiled at UCSB.

A set of assembly-language written subprograms is provided to facilitate the use of the Network by PL1 programmers. They serve as an interface between the user and the Network Control Program (NCP), which supervises all Network operations at this site.

A concept fundamental to PL1 Network operations is that of a completion code variable. Associated with each socket that is not in the closed state is a unique variable, called a completion code variable. This variable serves two purposes: it identifies the local socket referenced, and upon completion of the operation it contains a completion code to indicate the outcome of the operation to the programmer. It may be used at any time for any purpose other than Network operations. Its value at the beginning of a Network operation is immaterial; rather, it is the variable itself that is important. In all Network operations, whenever a completion code variable is called for, the only acceptable attributes for the variable are BINARY FIXED(31,0)ALIGNED.

In general, the subprograms will initiate the operation, and enter the WAIT state (consume no CPU time) until notified by the NCP of the completion of the operation. For such operations, the programmer supplies a time limit, with attributes BINARY FIXED(31,0), which specifies, in tenths of a second, the maximum length of time the subprogram is to wait for the operation to complete. If this time limit is exceeded, a completion code is supplied to indicate the error, and control is returned to the calling program, with the operation still pending.

In those operations requiring a socket to be specified by its socket identifier, the following format is used. The identifier is specified as a BINARY FIXED(31,0) array, singly dimensioned with at least two elements, and with a lower subscript bound of one. The first element is taken to be the site number, and the second element the socket number. A socket number may have any value, positive, negative, or zero. Since negative numbers are represented internally in two's complement form, for the purposes of the gender of a socket, a socket number is even or odd according to whether its absolute value is even or odd, respectively.

Following is a description of each subprogram, its function and calling sequence.

(1) OPEN

By invoking this subprogram, the user requests that the specified local socket be removed from the closed state and thus be prepared to participate in data transfer. There are three distinct operations that can be performed by this subprogram, and these are described below:

(A) CONNECT

By initiating this operation, the user requests that the specified local socket be logically connected to the specified foreign socket, and that data transfer between the two sockets be enabled. The subprogram will wait until the foreign socket either accepts or rejects the connection attempt. This operation is valid only when the socket is in the closed state.

The calling sequence is as follows:

```
CALL @OPEN(cmpcd, time,lclsck,fgnsck,ws);
```

where

'cmpcd' is the completion code variable to be associated with the local socket.
'time' is the length of time the subprogram is to wait for the completion of the operation.
'lclsck' is the socket identifier of the local socket.
'fgnsck' is the socket identifier of the foreign socket.
'ws' is a workspace which has the same attributes as a socket identifier, and, if the operation if successful, will become associated with the local socket for the life of the connection. In response to certain future operations, information may be returned to the programmer in this workspace.

The following completion codes are possible for this operation:

0, The operation was successful and the connection has been established
4, The specified completion code variable is already assigned to a socket; the operation was suppressed
8, The specified local socket is not in the closed

state; the operation was suppressed
12, All communication paths to the specified foreign site are in use; the operation was suppressed
16, Local resources are insufficient to support another connection; the operation was suppressed
20, The connection attempt was rejected by the foreign process
24, The specified local socket is not local to UCSB; the operation was suppressed
28, The specified foreign site is invalid; the operation was suppressed
32, An addressing or protection exception was encountered with respect to the specified workspace; the operation was suppressed
36, Either hardware at the foreign site is inoperative or the NCP's counterpart at the foreign site does not exist or has failed
40, Local and foreign sockets are both either send or receive sockets; the operation was suppressed
44, By operator command, all Network operations were terminated; the socket will be closed
60, An NCP control transmission error occured; the operation was suppressed
252, The specified time limit was exceeded; the operation was initiated but not completed

(B) LISTEN

This operation is a request for notification of any connection attempt directed toward the specified local socket. The subprogram waits until such a call is received, at which time the calling process' socket identifier is returned to the calling program. This operation is valid only when the designated local socket is in the closed state. The calling sequence is as follows:

```
CALL @OPEN(cmpcd, time,lclsck,ws);
```

where

'cmpcd' is the completion code variable to be associated with the local socket.
'time' is the length of time the subprogram is to wait for the completion of the operation.
'lclsck' is the socket identifier of the local socket.
'ws' is a workspace which, if the operation is successful, will contain the socket identifier of the calling socket, and will become associated with

the local socket for the life of the connection.
'ws' has the same attributes as a socket identifier.

The following completion codes are possible for this operation:

0, The operation was successful and a call has been received
4, The specified completion code variable is already assigned to a socket; the operation was suppressed
8, The specified local socket is not in the closed state; the operation was suppressed
12, Local resources are insufficient to support another connection; the operation was suppressed
16, The specified local socket is not local to UCSB; the operation was suppressed
20, An addressing or protection exception was encountered with respect to the specified workspace; the operation was suppressed
44, By operator command, all Network operations were terminated; the socket will be closed
252, The specified time limit was exceeded; the operation was initiated but not completed

(C) ACCEPT

This operation accepts connection with the foreign socket whose call caused successful completion of a previous LISTEN operation by the specified local socket. After completion of this operation, data may be transferred to or from the local socket, depending on its gender. This operation is valid only when the last operation referencing the local socket was a LISTEN operation. The subprogram will wait until the operation is completed.

The calling sequence is as follows:

```
CALL @OPEN(cmpcd, time);
```

where

'cmpcd' is the completion code variable associated with the local socket by a previous LISTEN operation.
'time' is the length of time the subprogram is to wait for the completion of the operation.

The following completion codes are possible for this operation:

0, The operation was successful, and the connection is established
4, The specified local socket is in the closed state (the NCP may have received an abort notification from the foreign process); the operation was suppressed
8, The previous operation specifying the designated local socket was not a LISTEN; the operation was suppressed
12, All communication paths to the specified foreign site are in use; the socket has been returned to the closed state
252, The specified time limit was exceeded; the operation was initiated but not completed

(2) CLOSE

This operation is a request that the specified local socket be returned to the closed state. If the last operation involving this socket was a LISTEN, this operation refuses the foreign process' connection attempt. If the last operation was a CONNECT, the attempt is aborted. If a connection is established, any data in transit from the local socket is allowed to reach the foreign socket and to be either received or flushed before the local socket is closed. The subprogram will wait until the socket has been returned to the closed state.

The calling sequence is as follows:

```
CALL @CLOSE(cmpcd,time);
```

where

'cmpcd' is the completion code variable associated with the local socket.
'time' is the length of time the subprogram is to wait for the completion of the operation.

The following completion codes are possible for this operation:

0, The operation was successful, and the socket has been returned to the closed state
8, The specified completion code variable is not currently assigned to a socket; the operation was suppressed
12, The specified local socket is in the process of being closed; the operation was suppressed, but the local socket will be closed
36, Either hardware at the foreign site is inoperative, or the NCP's counterpart at the foreign site does not exist or

has failed
44, By operator command, all Network operations were terminated; the socket will be closed
60, An NCP control transmission error occured; the operation was aborted
64, A transmission error occured; the operation was aborted, but the socket will be closed
252, The specified time limit was exceeded; the operation was initiated but not completed

(3) SEND

This operation causes data to be sent to the foreign socket. The subprogram will wait until the data has been received by the foreign socket, or until it has been queued locally by the NCP.

The calling sequence is as follows:

```
CALL @WRITE(cmpcd, bfr,len,time[,offset]);
```

where

'cmpcd' is the completion code variable associated with the local socket.
'bfr' is the data to be sent, and must be a singly dimensioned array, either DECIMAL or BINARY, FIXED or FLOAT.
'len' is the number of bits of data to be sent. If 'len' is non-positive, no operation is performed. 'len' has the attribute BINARY FIXED(31,0).
'time' is the length of time the subprogram is to wait for the completion of the operation.
'offset' is the bit offset from the first bit of 'bfr' at which data transmission is to begin, and must have the attributes BINARY FIXED(31,0). If not specified, 'offset' defaults to zero.

The following completion codes are possible for this operation:

0, The operation was successful, and the data has been sent
4, The specified local socket is not a SEND socket; the operation was suppressed
8, The specified completion code variable is not assigned to a socket; the operation was suppressed
12, A previous SEND operation is in progress; the operation was suppressed
16, The connection is not fully open; the operation was suppressed
20, The foreign socket terminated the connection before

completion of the SEND operation; not all data was transmitted
36, Either hardware at the foreign site is inoperative, or the NCP's counterpart at the foreign site does not exist or has failed
44, By operator command, all Network operations were terminated; the socket will be closed
52, One or more interrupts were received from the foreign socket; the operation was suppressed
56, An addressing exception was encountered with respect to the data buffer; the operation was suppressed
60, An NCP control transmission error occured; the operation was suppressed
64, A transmission error occured; the operation was aborted, and the socket will be closed
252, The specified time limit was exceeded; the operation was initiated but not completed

(4) RECEIVE

This operation causes data to be received from the foreign socket. The subprogram will wait until the reception of data is complete.

The calling sequence is as follows:

```
CALL @READ(cmpcd,bfr,len,time[,offset]);
```

where

'cmpcd' is the completion code variable associated with the local socket.
'bfr' is the variable into which the data is to be placed, and must be a singly dimensioned array, DECIMAL or BINARY, FIXED or FLOAT.
'len' is the number of bits of data to be received. If 'len' is non-positive, no operation is performed. 'len' has the attribute BINARY FIXED(31,0).
'time' is the length of time the subprogram is to wait for the completion of the operation.
'offset' is the bit offset from the first bit of 'bfr' at which the first bit of data is to be placed, and must have the attributes BINARY FIXED(31,0). If not specified, 'offset' defaults to zero.

The following completion codes are possible for this operation:

0, The operation was successful and the data has been received

4, The specified local socket is not a receive socket; the operation was suppressed
8, The specified completion code variable is not assigned to a socket; the operation was suppressed
12, A previous RECEIVE operation is in progress; the operation was suppressed
16, The connection is not fully open; the operation was suppressed
20, The foreign socket terminated the connection before completion of the RECEIVE operation; data is unpredictable
36, Either hardware at the foreign site is inoperative, or the NCP's counterpart at the foreign site does not exist or has failed
44, By operator command, all Network operations were terminated; the socket will be closed
24, An addressing or protection exception was encountered with respect to the data buffer; the operation was suppressed
52, One or more interrupts were received from the foreign socket; the operation was suppressed
60, An NCP control transmission error occured; the operation was aborted, and the socket will be closed
252, The specified time limit was exceeded; the operation was initiated but not completed

(5) CHECK

This operation causes the status of the specified local socket to be returned. There is no completion code variable associated with this operation, since it is always successful and the socket is identified by its socket number. This operation is valid at any time and is always completed immediately.

The calling sequence is as follows:

```
CALL @CHECK(lclsck,stat,mnem,fgnsck,deficit);
```

where

'lclsck' is the socket identifier of the local socket.
'stat' is a code for the status, and has the attribute BINARY FIXED(31,0).
'mnem' is a mnemonic for the status, and has the attribute CHARACTER, of any fixed length greater than or equal to eight. If 'mnem' is longer than eight bytes, it will be padded to the right with blanks.
'fgnsck' is the socket identifier of the foreign socket, or zero.

'deficit' is the send/receive deficit in bits, or zero, and has the attributes BINARY FIXED(31,0).

'lclsck' is the only argument the programmer need define; the others are output from the subprogram.

Following are the possible status codes, together with their mnemonics and interpretations.

'STAT'	'MNEM'	MEANING
0	OPEN	A connection is fully established. No SEND/RECEIVE operation is in progress. 'fgnsck' is the socket identifier of the connected socket. 'deficit' is the number of bits queued locally at the socket by the NCP and available to satisfy a future RECEIVE operation, or awaiting output as the result of a previous SEND operation.
1	LISTEN	A LISTEN has been issued.
2	CONNECT	A CONNECT has been issued. 'fgnsck' is the socket identifier of the foreign socket.
3	DECISION	A LISTEN has been completed. 'fgnsck' is the socket identifier of the calling socket.
4	CALL(S)	One or more calls have been received for the local socket. No LISTEN or CONNECT has been issued.
5	I/O	A connection is fully established. A SEND/RECEIVE operation is in progress. 'fgnsck' is the socket identifier of the connected socket. 'deficit' is the number of bits yet to be sent or received.
6	CLOSED	The socket is in the closed state.
7	<--DRAIN	The foreign socket is attempting to close the connection. The NCP has data yet to be read by the local socket. 'fgnsck' is the socket identifier of the connected socket. 'deficit' is the number of bits yet to be received.

8	DRAINED	The foreign socket is attempting to close the connection. The NCP is awaiting arrival at the foreign site of data currently in transit. 'fgnsck' is the socket identifier of the connected socket.
9	CLOSING	The local socket has issued a CLOSE. The NCP is in the process of returning the local socket to the closed state. 'fgnsck' is the socket identifier of the connected socket.
10	DRAIN-->	The local socket has issued a CLOSE. The NCP is completing the last SEND operation before returning the local socket to the closed state. 'fgnsck' is the socket identifier of the connected socket. 'deficit' is the number of bits the NCP has yet to transmit.

(6) IDENTIFY

This operation is used to identify a local socket by its completion code variable. The operation is valid at any time, and is always completed immediately. Since it is always successful, there are no completion codes for the operation, and the contents of the completion code variable are not changed.

The calling sequence is as follows:

```
CALL @ID(cmpcd,lclsck);
```

where

'cmpcd' is the completion code variable to be associated with the local socket
'lclsck' is set to the socket identifier of the local socket if the completion code variable is associated with a socket, or to zero otherwise.

(7) SIGNAL

This operation is used to convey a signal to the foreign process. The significance of the signal is completely user-dependent. The effect is that the next time the foreign socket attempts to initiate a RECEIVE or SEND operation, the operation will be suppressed, and a completion code supplied indicating that a signal had been received. The subprogram will wait until the signal has

been sent to the foreign NCP. This operation is valid only when the socket is fully open.

The calling sequence is as follows:

```
CALL @ID(cmpcd,time);
```

where

'cmpcd' is the completion code variable associated with the local socket.
'time' is the length of time the subprogram is to wait for completion of the operation.

The following completion codes are possible for this operation:

0, The operation was successful, and the signal has been sent
4, The specified completion code variable is not assigned to a socket; the operation was suppressed
8, The connection is not fully open; the operation was suppressed

Certain of the Network subprograms are intended for, although not restricted to, use with the On-Line System Network operators. The following is a general description of these operators:

In all Network operations involving the On-Line System Network operators, there are certain conventions concerning the format of the data sent and received. The data is grouped in 'messages' consisting of three fields, op code, length, and text, in that order. The op code is one byte in length and is a code which indicates how the text field is to be interpreted. The length field is two bytes long, and gives the length, in bits, of the text field, which contains the actual data. (The op code and length fields together are termed a header.)

The following op codes are presently defined:

op code	meaning
0	The op code is a NOP. No text field exists, and the contents of the length field are unpredictable. (This op code is used mainly as a delimiter.)
1	The text field contains EBCDIC characters, one character per byte. The On-Line System Network operators consider the characters as intended for display as soon as the text field has been received.

2 The text contains codes for keypushes, one byte per key. The On-Line operators consider the text as intended for execution as soon as the text field has been received.

3 The same as for an op code of 2, except that the On-Line operators consider that the execution of the keys will be delayed until all data for that receive operation has been received.

The standard format of data sent or received by the On-Line System operators is a string of messages, with the last message indicated by a header with a NOP op code, called a trailer. These conventions are the default situation; any of them may be overridden by appropriate programming.

Following are descriptions of those subprograms intended for use with the On-Line System operators.

(8) WRITE TO ON-LINE CONSOLE

This subprogram causes data, assumed to be characters represented by their EBCDIC codes, to be sent from the specified local socket. The characters are sent in a standard message, preceded by a header, and optionally followed by a trailer. The subprogram waits for the data to be received by the foreign socket, or to be queued locally by the NCP. This operation is valid only when the local socket is a send socket, and is fully open.

The calling sequence is as follows:

```
CALL @WTOLC(cmpcd,bfr,len,level,across,down,time);
```

where

'cmpcd' is the completion code variable associated with the local socket.
'bfr' is the character string to be sent, and must have the attribute STRING, either CHARACTER or BIT, of any length, fixed or varying.
'len', in absolute value, is the number of characters to be sent. If 'len' is positive or zero, the end of the data is indicated by a trailer; if 'len' is negative, no trailer is sent. 'len' has the attributes BINARY FIXED(31,0).
'level' indicates the mode of typing. The absolute value of 'level', if non-zero, is the shift level on which to type, and must be less than or equal to nine. If greater than nine, one is used. If 'level' is positive, 'across'

and down are relative to the current typing location. If 'level' is negative, the carriage is positioned to the upper left-hand corner before typing begins, thus making 'across' and 'down' absolute coordinates. If 'level' is zero, the typing is in case one characters, and a carriage return precedes the positioning for 'across' and 'down'. If the characters to be displayed are case one characters, the header sent will have an op code which indicates characters to be displayed as soon as they are received; otherwise, the op code will indicate buttons to be executed as soon as received. 'level' has the attributes BINARY FIXED(31,0).

'across' is the number of spaces to be moved horizontally across the display tube before beginning to type. 'across' can have any value, positive, negative, or zero, and has the attributes BINARY FIXED(31,0).

'down' is the number of lines to be moved vertically down the display tube before beginning to type. 'down' can have any value, positive, negative, or zero, and has the attributes BINARY FIXED(31,0).

'time' is the length of time the subprogram is to wait for the completion of the operation.

The completion codes for this operation are the same as for SEND.

(9) READ FROM ON-LINE CONSOLE

This subprogram receives data, assumed to be characters represented by their EBCDIC codes, from the foreign socket in one or more standard messages. The subprogram will wait for the data, optionally followed by a trailer, to be received by the local socket. This operation is valid only when the local socket is a receive socket and is fully open.

The calling sequence is as follows:

```
CALL @RFOLC(cmpcd,bfr,len,time);
```

where

'cmpcd' is the completion code variable associated with the local socket.

'bfr' is the variable into which the data is to be placed, and has the attribute CHARACTER, of any fixed length. If the length of 'bfr' is greater than the amount of data received, 'bfr' will be padded to the right with blanks.

'len', in absolute value, is the maximum number of characters to be placed in 'bfr'. If the length fields of the

header(s) received total more data than 'len', the excess data will be received, but will not be placed into 'bfr', and will not be accessible to the program. If 'len' is positive or zero, data will be received until a trailer is encountered. If 'len' is negative, a single message, exclusive of trailer, will be received. 'len' has the attributes BINARY FIXED(31,0).

'time' is the length of time the subprogram is to wait for the completion of the operation.

The completion codes for this operation are the same as for RECEIVE.

(10) WRITE TO ON-LINE CONSOLE WITH REPLY

This subprogram combines the functions of 'WRITE TO ON-LINE CONSOLE' and 'READ FROM ON-LINE CONSOLE'. The subprogram first sends a string of data to the foreign socket from the specified send socket, waits for it to be received by the foreign socket (or queued locally by the NCP), and then waits for a reply directed toward the specified receive socket. The operation is valid only when the sockets have the correct gender and both are fully open.

The calling sequence is as follows:

```
CALL @WTOLCR(cmpcd, bfr, len, level, across, down, time,
             cmpcd2, bfr2, len2);
```

where

'cmpcd' is the completion code variable associated with the local send socket.
'bfr' is as in 'WRITE TO ON-LINE CONSOLE'.
'len' is as in 'WRITE TO ON-LINE CONSOLE'.
'level' is as in 'WRITE TO ON-LINE CONSOLE'.
'across' is as in 'WRITE TO ON-LINE CONSOLE'.
'down' is as in 'WRITE TO ON-LINE CONSOLE'.
'time' is the length of time the subprogram is to wait for completion, individually, of the transmission and reception of data.
'cmpcd2' is the completion code variable associated with the local receive socket.
'bfr2' is as 'bfr' in 'READ FROM ON-LINE CONSOLE'.
'len2' is as 'len' in 'READ FROM ON-LINE CONSOLE'.

The completion codes for the send socket are the same as for the SEND operation, and the completion codes for the receive socket are the same as for the RECEIVE operation.

(11) ERASE

This subprogram causes data constituting an On-Line System command to erase the display tube to be sent from the specified local socket. The data is sent in a single standard message, including an op code indicating characters to be displayed as they are received, and optionally including a trailer. The subprogram waits for the data to be received by the foreign socket, or to be queued locally by the NCP. This operation is valid only when the local socket is a send socket, and is fully open.

The calling sequence is as follows:

```
CALL @ERASE(cmpcd,delay1,delay2,time);
```

where

'cmpcd' is the completion code variable associated with the local socket.
'delay1', in absolute value, is the length of time, in tenths of a second, the subprogram is to pause (in the WAIT state) before sending the erase. If 'delay1' is positive or zero, a trailer will be sent after the erase; if negative, no trailer will be sent. 'delay1' has the attributes BINARY FIXED(31,0).
'delay2', is the length of time, in tenths of a second, the subprogram is to pause after sending the erase and before returning control to the calling program, and has the attributes BINARY FIXED(31,0).
'time' is the length of time the subprogram is to wait for the completion of the operation.

The completion codes for this operation are the same as for SEND.