NATIONAL RADIO ASTRONOMY OBSERVATORY
Charlottesville, Virginia

March 28, 1975

VLA COMPUTER MEMORANDUM #120

SUPERBEE HIVE CONTROLLER

G. Hunt

1 The User's Guide to the Super Bee Symbiont Task

When communicating with any program or processor via the Super Bees, it is strongly advised that this be done using the SB task, in which case the Logical Device Name is SB1, SB2, etc. The task provides the user with many editing facilities which are not otherwise available.

The computer signals its readiness to receive data by positioning the cursor at the beginning of the line following its original position and by clearing the INS CHAR condition if set. It is then possible to type a line or record consisting of Displayable Characters and the line may be edited using the Function Characters.

The Displayable Characters comprise the numerical digits, the lower case alphabet, the upper case alphabet, as well as the following 32 special characters:-

```
!"#$%&'()*+,-./
:;<=>?[']¤_|{|}
and blank.
```

Any other character is interpreted as a Function Character. Only a few of these are recognised by the software, namely those listed in Table 1.1, which have the functions described there. All other characters affect the input Buffer in no way, and are merely echoed with the (barely audible) BEL (or squeek).

A Buffer is built in the buffer area of the requisite program or processor which is a core image of the line currently appearing on the screen. A Buffer Pointer is maintained and is updated to correspond to the current cursor position.

The SB task always has a read queued on each of the terminals, thus it can react to a number of Function Characters at any time. The following characters are implemented in this mode:-

Function 1
CNTL S
Function 2
CNTL Q
Function 8 (Consol Interrupt).

Function 1 and Function 2 (also CNTL S and CNTL Q) may be used to hold the terminal to examine a listing, but one should be careful to clear the hold before leaving. Have pity on the next innocent user! Function 8 is only effective when the system CI file is defaulted to the terminal. It is not possible to bring up the Operator Communication Task (OC) while sitting at another terminal, nor is it possible to bring up the OC of a remote CPU.

Table 1.1

BS (Control H) Backspace	The Buffer Pointer is decremented by 1. If the Buffer Pointer already points at position one, this character is ignored.
HT (Control I) Tab	The Buffer Pointer is advanced to the next tab position. If no further tab position is defined, this character is ignored.
LF (Control J) Linefeed	This is an End-of-Record Character.
CR (Control M) Carriage Return	This is an End-of-Record Character.
ESC (Control [) Escape Control	The character following this will be treated as an Escape Sequence Character. Should the next character also be an ESC character, it will be ignored.
US (Control _) Newline	This is an End-of-Record Character.
TAB SET (ESC 1) Tab Set	A tab position is defined at the position defined by the Buffer Pointer. If there is no room in the tab table for a new tab position, this character is ignored.
TAB CLEAR (ESC 2) Tab Clear	A tab position defined at the position of the Buffer Pointer is cleared. Should there be no such tab position, this character is ignored.

Table 1.1 (Cont.)

-> (ESC C) Advance Cursor	The Buffer Pointer is incremented by 1. If the Buffer Pointer already points to the last character in the Buffer, this character is handled as an End-of-Record Character.
<- (ESC D) Backspace	The Buffer Pointer is decremented by 1. If the Buffer Pointer already points at position one, this character is ignored.
CLEAR (ESC E) Clear	This is an Erase Character which also causes all tab positions to be reset and the screen to be cleared.
HOME (ESC H) Home Cursor	This is an Erase Character which does not clear the current line on the screen, but repositions the cursor at the top left-hand corner of the screen.
ERM (ESC J) Erase to end of memory	This is an Erase Character which clears both the Buffer and the screen from the current position. It may be used in conjunction with HOME to clear the Buffer and the screen, thus performing the same function as CLEAR with the important exception that the tab positions are not reset.
EOL (ESC K) Erase to end of line	This is an Erase Character which clears both the Buffer and the current line on the screen from the current position.
DEL LINE (ESC M) Delete Line	This is an Erase Character.

Table 1.1 (Cont.)

DEL CHAR (ESC P) Delete Character	The character at the cursor position is deleted from the Buffer and the screen. All subsequent characters in the Buffer are relocated one position earlier to fill the gap.
INS CHAR ON (ESC Q) OFF (ESC R) Insert Character	When the INS CHAR key is depressed the insert mode is toggled. When the lamp on the key is on, the insert mode is active. In the active mode any character transmitted will cause the current and all subsequent characters to be relocated one position later in the Buffer; the cursor and the Buffer Pointer are also advanced. Characters thus caused to be relocated beyond the end of the Buffer will be lost, however they will remain on the screen. This facility should therefore be used sparingly and with care. The INS CHAR mode will be cleared when a new read command is issued, but not after an Erase Character.
BACK TAB (ESC I) Back Tab	The Buffer Pointer is reset to the previous defined tab position. If the Buffer Pointer is already at position one, this character is ignored.
Function 3 (ESC r) Tab Initiallize	The screen and the Buffer are cleared, and the tabs are set to the standard positions 1,8,20,35,73 on the screen and in the Buffer.
DEL (or ESC DEL) Rub Out	This is an Erase Character.

Table 1.1 (Cont.)

DC1 (CNTL Q) Screen Free	The Screen Hold condition is cleared.
DC3 (CNTL S) Screen Hold	The Screen Hold condition is set; all further data transfer requests are held up until the hold condition is cleared.
Function 1 (ESC p) Screen Hold	The Screen Hold condition is set; all further data transfer requests are held up until the hold condition is cleared.
Function 2 (ESC q) Screen Free	The Screen Hold condition is cleared.
Function 8 (ESC w) Consol Interrupt Simulation	The Operator Communications Task is activated. If this is typed during a read operation, it also acts as an End-of-Record Character. The OC Task is activated only if the CI file of the attendant CPU is defaulted to the terminal. It is not possible to activate the OC Task in a remote CPU.

1.1 Escape Sequence Characters

An Escape Sequence Character comprises two characters, namely ESC followed by any other single character. If the second character is ESC it will be ignored by the software; if, however, it is a control character, the preceeding ESC will be ignored. Any other second character will lead to the pair being considered as a single character. All special function keys generate this two character code.

1.2 End-of-Record Character

The following characters are End-of-Record Characters unconditionally:-

LF
CR
NEWLINE (US)
Function 8 (Consol Interrupt)

The following character is an End-of-Record Character conditionally (see Table 1.1):-

->

When any of these is encountered, the record is considered terminated. The Buffer will remain as it stands, independent of the Buffer Pointer. Thus, having inserted a character, there is no need to reposition the Buffer Pointer after the last character prior to depressing a key producing an End-of-Record Character.

1.3 Erase Character

The following characters are Erase Characters unconditionally:-

DEL LINE
DEL

The following characters are Erase Characters with other special functions (see Table 1.1):-

CLEAR
HOME
FUNCTION 3 (Tab Initiallize)

An Erase Character causes the Buffer to be cleared to blanks, the Buffer Pointer to be repositioned at zero, and (except HOME) the current line to be cleared on the screen.

The following characters cause the Buffer to be cleared to blanks from the Buffer Pointer position to the end of the Buffer, and the screen to be cleared from the cursor position:-

EOL
ERM

1.4 Tabulation

There is space reserved in memory for 8 tab positions per terminal, the first one of which is always set to 1. Since these tab positions are memory resident, it is advised that, each time a user starts input, CLEAR or Tab Initiallize (Function 3) should be transmitted before any tabulation is done. This has the function of clearing both the software

(memory) and hardware (terminal) tabs, and, in the case of Function 3, of setting the standard tab positions.

The tab table comprises 8 contiguous bytes. The tabs that have been set are stored in ascending order, and vacant bytes are marked by having the top bit set (i.e. #80). When a TAB SET is transmitted the program searches the table for the correct location and, should there be tabs defined further along the input line, it inserts the tab position, relocating the current and all subsequent entries one location later. Any tab position previously in the last entry location will be lost in memory but not on the terminal. When a TAB CLEAR is transmitted, the program searches for the correct location and removes the entry by relocating all subsequent entries one location earlier. The last byte is reset vacant. It is thus obvious that by excessive and injudicious use of TAB SET and TAB CLEAR the tab positions in memory and on the terminal may differ radically: YOU HAVE BEEN WARNED. For most purposes, however, only 4 or 5 tab positions are required at any one time and no problem should arise.

The TAB and BACK TAB cause the tab table to be searched for the next defined tab position following or preceeding the current Buffer Pointer respectively. A BACK TAB when the Buffer Pointer stands at one is ignored; a TAB when the Buffer Pointer is further than the last defined tab position in the tab table is ignored.

1.5 File Control Functions

The standard file control functions may be used on any file assigned to the SB task. The effect of using these functions is given in Table 1.2.

Table 1.2

Function	Transmitted Characters	Effect
REWind	HOM ERM	Clears screen, leaving tabs unchanged.
BKFile	ESC V (PREV PAGE)	Resets the cursor to the top of the previous "page".
BKRecord	CR LF ESC A ESC A	Resets the cursor to the start of the previous line.
AVRecord	CR LF	Resets the cursor to the start of the next line.
AVFile	ESC U (NEXT PAGE)	Resets the cursor to the top of the next "page".
WEOf	CR LF $ $	Writes $$ on the next line
HOMe	CR LF CR LF CR LF	Skips 3 lines.

1.6 User's File Table

In the User's File Table (UFT) the option word (UOP) is not tested but assumed to be #A000 for all functions except WRITE. For this function the whole word is passed on to MAX. (This was originally necessary as BASIC uses the character '}' as a terminator). The output buffer is unaltered; any truncation that takes place is due to the maximum length defined for the terminals at system generation. This is currently 80 characters, inclusive the carriage control characters. Thus lines written on the screen by standard ModComp software will have a maximum length of 78 characters, since they typically use a carriage control character followed by #02, which is not transmitted.

The actual number of bytes transferred in any function is stored in UFT word 4 (UBC).

The Device Position Index (UDC) is incremented by 1 for the following functions:

READ
WRITE
AVR
WEO,

it is decremented by 1 for

BKR,

and it is cleared to zero for

REW
BKF
AVF
HOM.

At the start of all transfers the UFT status is re-initiallized with UFT hold (HO), buffer busy (BB), and UFT busy (UB). The hold bit is reset on a READ function as soon as the first character is received (so that the Operator Communications task does not time out!). All three bits are reset when this or any other function is complete.

2 The Programmer's Guide to the Super Bee Symbiont Task

The program is devided logically into 3 main steps. In the first the Physical Device Queue (PDQ) is searched to see if any requests queued by the Operating System (MAX) are for a terminal that is currently not busy. Should this be the case the request is queued internally on the terminal and is removed from the PDQ. In the second step, the internal status of each terminal is examined. If a request was queued at the previous step, the appropriate command is output; if a command was output at a previous time, this is checked for completion. The third step is only for READ commands: after each character has been read, it is examined and interpreted as described in the User's Guide.

2.1 Search of the Physical Device Queue

When the task is entered, it first checks to see if the external controller has been set busy by MAX. If it has not, there are no new requests, and the program proceeds to the next step. If it is set to busy, this indicates that one or more requests have been queued and that their node(s) have been attached to the PDQ. The queue is scanned sequentially from the top, thus dealing with the oldest node first. If the terminal which the node wishes to access is not currently busy, the node is "stolen" from MAX by extracting the node from the PDQ and placing this in the internal controller of the relevant terminal, and setting the status to busy (See Figure 2.1). This section of code is executed with the Taskmaster locked out, so that no further requests can be queued until the whole queue has been scanned.

2.2 Test of Controller Status

The principal section of the task deals with the status of each controller and of the associated transfer requests of all the internal controllers. If a controller was set busy at the previous step, then a transfer is initiated. If the transfer has already been initiated then this transfer is checked for completion. The controllers are scanned sequentially; thus all terminals are checked each time the task is entered. When all controllers have been examined, the Taskmaster is re-enabled, the next task in the CPUQ being normally activated. However, if a transfer has just been completed, and the node returned to MAX, the CPUQ is scanned from the top (to prevent MAX getting upset when he has an I/O node from the OC Task to queue on the SB Task)

2.2.1 Transfer Initiation

Before making a transfer request, all the appropriate bits are set to keep MAX from getting upset viz the Logical Device Busy (LDB) bit is set, and the caller's UPT status is re-initiallized with buffer hold (HO), buffer busy (BB), and UPT busy (UB). (See MAX II/III volume II, appendix A). The setting of the LDB does not seem to be necessary, but it alleviates two problems:

1) The PDQ is scanned by the I/O system to place the nodes from the various tasks in order of priority. The first node (if the Physical Device Status (PDT) is busy) is assumed to be in operation and all subsequent nodes are queued behind this one. Since this node is not in operation (the busy node has been "stolen") the queuing algorithm fails and, if two or more tasks are requesting reads from a terminal it is

impossible to tell which one has queued the read. Setting the LDB makes MAX put all new nodes for the device on the Slough Queue (LSQ), whence they are queued properly when the LDB is reset.

2) When an I/O operation is terminated, MAX resets the LDB and sets the UFT address in the node to zero if he finds it in the PDQ. Since a "stolen" node cannot be found there, the only way for the SB task to determine whether the operation has been terminated is to test the LDB.

A read service is initiallized by transmitting the CR, LF, and INS CHAR OFF characters, clearing the Buffer to blanks, and then making a REX,READ call for single characters. The ModComp handler has been modified to recognise and not to return the ESC character but to return the following character to the SB task with bit 8 (#80) set. Thus all ESC sequence characters are also read as single characters, but must be echoed as double characters.

All other services are initiallized merely by setting the appropriate UFT, and by executing a REX,WRITE call.

2.2.2 Transfer Completion

MAX signals the completion of a transfer by resetting all the timing conditions in the UFT. For all services except READ, if a previously initiated transfer has reached completion, the UFT status is copied into the caller's UFT, the "stolen" node is set not busy by setting the UFT address in the node to zero, and the calling task is released from its I/O hold condition if the REX call was made without the quick return option. The node is then returned to the caller's Available Node Queue and the number of his nodes in use is decremented by one. The dequeuing operation is carried out with all interrupts below level 4 locked out to prevent another REX service of the same task completing and signalling completion by dequeuing another node in the same queue at the same time.

In a READ service, as soon as the output of the initial characters has been completed, the read UFT is tested for completion. The single character returned is examined and interpreted as described in the User's Guide. If the input character was an End-of-Record Character, the Buffer is terminated by storing a NUL (#00) immediately after the last non-blank character, if there is room. If the line is completely blank, the NUL character is stored after the first blank. The process then continues exactly as for the other services. Since all communications with the Super Bees are performed in full duplex mode, any character to be stored on the screen must be echoed by executing a REX,WRITE to the terminal. Some of the characters are not echoed at all, but some are echoed with many characters. This last was one of the main reasons for making the Super Bee handler a symbiont task rather than a simple device handler, since a handler under MAX can echo at most one character at a time excluding the possibility, for example, of clearing the line on the screen when a RUB OUT (DEL) is typed.

Even when no REX service is in operation, the SB task always has a read queued on the terminal, thus allowing a modicum of communication with a terminal that is otherwise dead. At present only a few characters have any meaning in this mode (see User's Guide). These are not echoed; simply a further REX,READ is queued.

2.3 Controllers

The "external" controller is the SB task common area which is used by MAX as a Physical Device Table (PDT). The PDT status is normally used by the device handlers at Service Interrupt (SI) and Data Interrupt (DI). The only bit that is set before an interrupt occurs is the Controller Busy (CB) bit. Since the SB task is a task and not a device there is no SI or DI and the rest of the status is not used. Only the first 5 words of the PDT are used by MAX as with a standard device. The internal option word (SBO) has been appended. At present only bit 0 (HDN) is used to determine the type of CPUQ scanning to be used by the Taskmaster when the SB task exits.

There is one internal controller for each terminal operated by the task. The controller contains all information that is device dependent viz: the status, an internal PDQ, the terminal name, the input character buffer, the echo buffer, the Buffer Pointer, the current tab positions, the UFT tables for the transfer requests to the physical devices, and the non-reentrant code. The Special Option (SO) is set in the input UFT. Quite what this is supposed to do I don't know, but it does allow an odd number of bytes to be read, in this case 1, whereas this is not normally possible.

The following bits in the internal controller are used:

4	BT	Buffer Full. This is set during a read service when a character has been entered into the last position in the Buffer. Since most REX,READs to the Super Bees are made, logically enough, specifying the Buffer to have length 80 characters and since storing the 80th character on the screen causes the cursor to move to the beginning of the subsequent line, the last character is always echoed followed by a BS. The bit is also used to terminate the service when the echo of the final character is complete.
5	XF	A REX service has been initiallized. This is set to tell the task to check the status of the UFTs each time it is entered subsequently.
7	CB	Controller Busy. This indicates that a node has stolen from the external PDQ and that the task cannot accept any requests for this terminal until this bit has been reset.
8	HLD	Screen Hold. This indicates that Function 1 (or CNTL S) was typed and that no more services may be performed on this terminal until the condition has been cleared.
10	INS	Insert Character Mode. During a read service this causes a Storable Character to be stored in the Buffer in the normal manner, but in addition all characters already in the Buffer at or beyond the current position to be relocated one byte later.
12	IWR	Write Initiated. A REX,WRITE call has been made to

the physical device.

13	IRE	Read Initiated. A REX,READ call has been made to the physical device.
14	CW	Write Service. The REX call made to the terminal was REX,WRITE. This is currently set but not otherwise used.
15	CR	Read Service. The REX call made to the terminal was REX,READ.

2.4 Registers

The follwing registers have dedicated functions:

1		Current internal controller address.
3	NODA	Address of current I/O node
4	RCBA	Address of caller's RCB
5	LDTA	Address of current LDT
6	CRXA	Address of REX call
7	UFTA	Address of caller's UFT
9		Link register for internal procedures
10		Input character (READ only)
12		Address of caller's Buffer (READ only)
13		Current Buffer Pointer (READ only)

2.5 Adding more terminals

The SB task has been written so as to make the addition of extra terminals as easy as possible. Five extra macro calls must be added per terminal in the Logical I/O (LIO) Block of the system generation (sysgen), one in the SB task itself, and one in the Task Resource Block (TWA) definition for SB.

In the LIO Block, the macro SUBPARTYLINE is necessary to run the Super Bees using the ModComp device handler, which is primitive but adequate. All the various cludges to the I/O system (IOS) are performed at SI and DI by the code generated by this macro. For each terminal there must also be two SUBCHANNEL macro calls, the first for the output channel, and the second for the input channel. This generates the appropriate PDT for partyline devices. All SUBCHANNEL macro calls must follow the first one, and must do so in the correct order. The LIO Block at present contains the following calls:

```
  SUBPARTYLINE  A,#10
  SUBCHANNEL  A,00,,,NONE,7,2,NOECHO
  SUBCHANNEL  A,01,,,NONE,7,2,NOECHO
  SUBCHANNEL  A,02,,,NONE,7,2,NOECHO
  SUBCHANNEL  A,03,,,NONE,7,2,NOECHO
to which should be added:
  SUBCHANNEL  A,04,,,NONE,7,2,NOECHO
  SUBCHANNEL  A,05,,,NONE,7,2,NOECHO
etc.
```

The LDT for a device is generated with the DEVICE macro. Since the terminals are accessed through the SB task as SB1, SB2, etc. one DEVICE macro call must exist for each such name. The SB task transfers data

using the logical device names BW1, BR1, etc. requiring that there be two DEVICE calls for each terminal, one for input, one for output. At present the LIO Block contains the following calls:

```
  DEVICE      SB1,SB,82,#24,TASKDEVICE
  DEVICE      SB2,SB,82,#24,TASKDEVICE
to which should be added:
  DEVICE      SB3,SB,82,#24,TASKDEVICE
etc., also
  DEVICE      BW1,A00,80,#20
  DEVICE      BR1,A01,80,#20
  DEVICE      BW2,A02,80,#20
  DEVICE      BR2,A03,80,#20
to which should be added:
  DEVICE      BW3,A04,80,#20
  DEVICE      BR3,A05,80,#20
etc.
```

In the SB task there must be one internal controller per terminal. This is generated by the SBCONTROLLER macro. At present there are the following controllers defined:

```
  SBCONTROLLER  1
  SBCONTROLLER  2
to which should be added:
  SBCONTROLLER  3
etc.
```

Since the SB task communicates with the physical devices via the logical devices BW1, BR1, etc., these files must be defined in the TWA. A simple macro SBFILE has been written which makes the two task file assignments. At present the following files have been defined:

```
  SBFILE      1
  SBFILE      2
to which should be added:
  SBFILE      3
etc.
```

The parameter in the call of SBCONTROLLER generates a reference to the files BWn and BRn, and the call of SBFILE equates the files BWn and BRn to the logical devices of the same name. The value of n may take any value between 0 and 9. Provided the SBCONTROLLER arguement and the SBFILE arguement correspond to one another and to the logical devices defined by the DEVICE macro, these macro calls may be made in any order, may take any of the 10 possible values and need not be sequential.