THE VRTX/68000 INTERFACE LIBRARY FOR THE ALCYON C COMPILER

Version 3

Document Number 597113001

December 1984

REV.	REVISION HISTORY	PRINT DATE
-ØØ1	Original Issue	12/84

Copyright 1984
Hunter & Ready, Inc.
445 Sherman Avenue
P.O. Box 6Ø8Ø3
Palo Alto, CA 94306-Ø8Ø3
415/326-295Ø
TELEX: 278835 (HRI UR)

THE VRTX/68000 INTERFACE LIBRARY FOR THE ALCYON C COMPILER

This document contains source code listings for the VRTX[1] C interface library to the Alcyon C compiler. This cross-compiler is hosted on VAX/UNIX[2] BSD 4.2 and generates code compatible with an Alcyon cross-assembler that is also available on the system. Both of these tools are part of the Alcyon C cross-compiler system.

This document includes instructions for creating this code as a library and tools that aid in developing high-level language modules for use with VRTX.

---

1. VRTX and VRTX/68000 are registered trademarks of Hunter & Ready, Inc. and may be used only in reference to Hunter & Ready products.

2. UNIX is a registered trademark of Bell Laboratories

TABLE OF CONTENTS

INTRODUCTION

These source listings are in the format accepted by the Alcyon assembler for the 68000 microprocessor. This assembler is available from Alcyon and must be purchased separately from the Alcyon C compiler for the 68000 microprocessor. For additional information on the syntax of the Alcyon C Compiler or the Alcyon Assembler, please refer to the Alcyon C Compiler Manual, V4.7.

The files and information supplied in this document were developed and tested using a VAX 11/750 running UNIX BSD 4.2.

In all cases, the revision number of the Alcyon tools used to develop this library is noted. Updates made by Alcyon may make some of these suggestions no longer applicable. If you have problems, be sure to match Alcyon revision numbers with those found in this document.

COMPILER OPERATION

All listings found in this document were tested using the assembler and compiler found on the VAX 11/750 System running Berkeley 4.2 UNIX. The identifying version numbers are:

Alcyon C Compiler	Rev 4.7
Alcyon Assembler	Rev 4.7

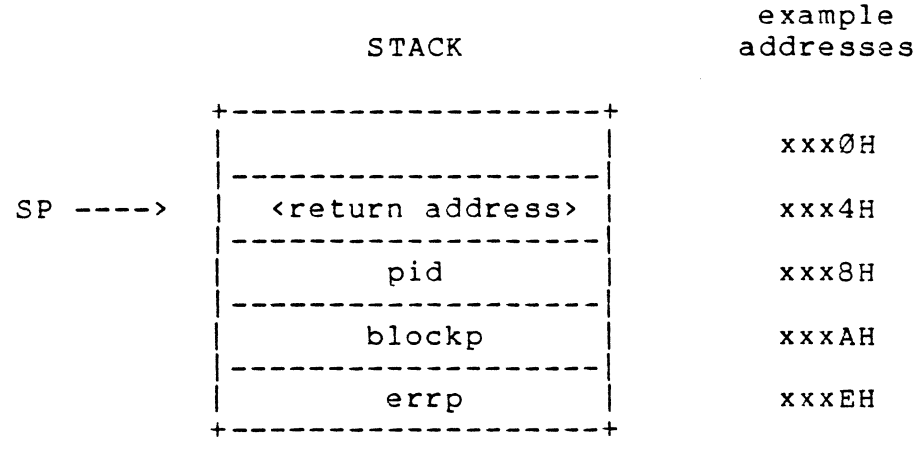
The parameter passing specifics of this compiler and other considerations are described below.

COMPILER CHARACTERISTICS

The Alcyon C Compiler for the 68000 generates the following sizes for the standard C types:

char	8 bits
short	16 bits
int	16 bits
long	32 bits
pointer variables	32 bits

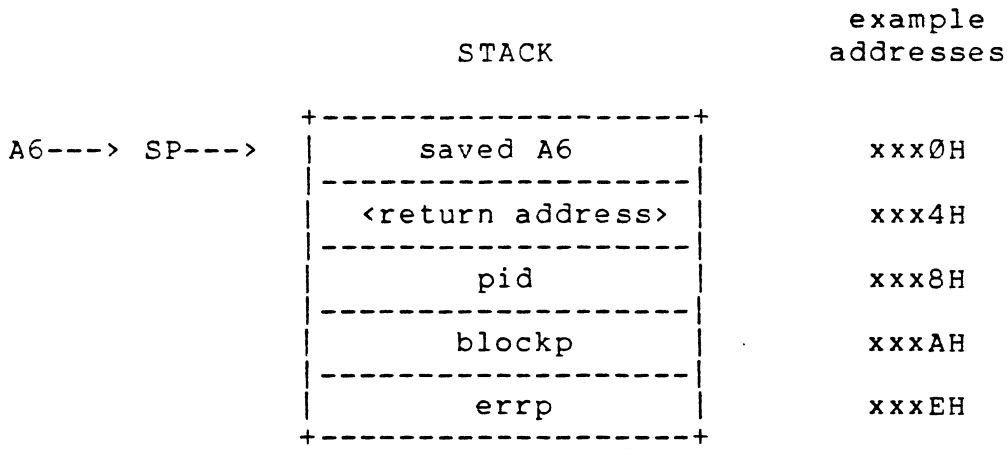
Parameters to procedure and function calls are passed on a stack with the address of the most recently pushed item in SP (A7). The compiler generates code to place the parameters on the stack in reverse order of appearance in the call. That is, the last parameter in the parameter list is pushed onto the stack first. After pushing the parameters onto the stack, execution begins at the called function or procedure. For example, at entry into the sc_rblock procedure code, the call **sc_rblock(pid,blockp,errp)** generates a stack with the following configuration:



The top of the stack, SP, points to the stack location which contains the address that the processor returns to at the conclusion of the interface routine code.

At the beginning of each subroutine, the compiler generates code to save the current contents of register A6 on the stack, and then to load A6 with the new value of the stack pointer. From then on, A6 can be used to access the parameters on the stack. By using A6 as a base address for parameters instead of SP, it is possible to use a fixed offset for each parameter that is independent of the number of items on the stack. The content of A6 at entry into the subroutine code is called the **frame pointer.**

The example below shows what the stack looks like after saving the frame pointer on the stack and loading it with the stack pointer:



In order to access these parameters, the interface routines must know their location in the stack relative to the frame pointer. This value is the **offset** and is a positive integer indicating the number of bytes between the stack top (as pointed to by the frame pointer) and a given parameter. In the above example, the offset for pid is 8 bytes, the offset for blockp is A bytes and the offset for errp is E bytes.

Although char TYPE values are only 8 bits in length, a word (16 bits) is pushed onto the stack when a character variable is passed to a routine. This compiler expects the character value to reside in the low order byte of the word on the stack.

For interface routines which return values (functions), the Alcyon C Compiler for the 68ØØØ expects the single return value in register DØ. The following table shows the register required for the returned TYPEs of parameters.

int	DØ
char	DØ
long	DØ
pointer	DØ

In addition, certain input parameters are actually pointers to locations that are filled by the interface routine. In this situation, the address is accessed off the stack and that location is set to the returned value. This is the case with the **errp** parameter in most of the interface routines.

INTERFACE LIBRARY

This section contains the listings that comprise the C interface library for use with the Alcyon C compiler in VRTX/68000 Version 3 applications. It consists of six assembly language modules listed in the assembly language accepted by the Alcyon Assembler.

ASSEMBLY INSTRUCTIONS

Each file listing in this section contains header information, assembly language instructions and comments. Header information, delineated by the comment indicator '*', is optional but we suggest it be included for future reference and ease of maintenance.

One of the editors resident on the system should be used to enter the code exactly as listed in the file names suggested in the header. Refer to the editor manual for details on editor operation.

Once all the listings are typed into the appropriate files, the next step is to assemble the files into relocatable object modules. This is accomplished by assembling each file individually as in:

```
as68 <myfile>
```

Alternatively, a command file may be created to assemble all the files automatically. The command file should contain the following lines:

```
#
as68 task.s
as68 memory.s
as68 sync.s
as68 clock.s
as68 chario.s
as68 component.s
```

This file, called **assemble**, is created with the editor and then made executable (see UNIX command **chmod**) and executed by typing **assemble** on the command line and pressing the carriage return key. After this command file completes execution, all interface routines are assembled. Additional options and listings may be requested from the assembler.

After all routines are assembled, the interface library is built by using the command **ar68**. Again, it is easiest to create a command file to build the interface library. The following command file, **bldvclib**, places all the interface routines into a library called **vclib.lib**, which stands for VRTX C Library. The

bldvclib command removes any previously built copies of the library vclib.lib before building the new one.

```
#
rm vclib
ar68 r vclib chario.o clock.o memory.o sync.o task.o component.o
        stdio.o
```

The above file is executed by typing **bldvclib** on the command line and pressing the carriage return key. Of course, all the interface library files must be successfully assembled before executing this command file.

SOURCE LISTINGS

The interface listings appear in the following pages. The header information pertains to the file into which the code is typed.

Each assembly language listing shown in the following section contains a header that has important information regarding the operation of the interface routine. The header is surrounded by the assembler comment delimiter, '*', and appears at the beginning of each file. The information:

```
file : <myfile>
```

indicates that the listing should be typed into a file named **myfile**. This particular name is not required; however, for the purposes of examples and command files, this document refers to that listing as myfile.

Before each interface call, the header information details the format of the code as it would be defined for a high-level language routine. It also details the relative offsets on the stack for each parameter passed into the interface routine. For example, the notation for the system call sc_rblock is:

```
*   sc_rblock(pid,blockp,errp)
*       int pid;  int *blockp, *errp;
*
*   pid=8:W, blockp=10:L, errp=14:L
```

This indicates that the procedure call **sc_rblock** has three input parameters: one of size word (W; 16 bits in length) and two of size long (L; 32 bits in length). Input parameter **pid** is located 8 bytes in a positive displacement from the frame pointer (A6 in this implementation). Similarly, **blockp** is located 10 bytes and **errp** is located 14 bytes from the frame pointer. Characters, although treated as 8 bit values by the compiler are always pushed onto the stack as a word (W) length. The character value is found in the high byte of that word. Refer to the previous

section for additional information on the stack and location of parameters within the stack.

Also contained in each header is the command line required to assemble the file.

SPECIAL CONSIDERATIONS

The C interface library does not allow the creation of both User mode and Supervisor mode tasks in the same program. As supplied, the sc_tcreate function creates tasks in User mode. To cause tasks to be created in Supervisor mode, the value of the symbol TASKMODE must be changed to 1.

```
*****************************************************************
*                                                               *
* ALCYON C COMPILER                    VERSION 3.Ø   DATE 11/84 *
*                                                               *
* COPYRIGHT 1984, HUNTER & READY, INC.                          *
*                                                               *
*****************************************************************
*                                                               *
*   FILE : task.s                                               *
*                                                               *
*   VRTX calls :  sc_tcreate                                    *
*                 sc_tdelete                                    *
*                 sc_tsuspend                                   *
*                 sc_tresume                                    *
*                 sc_tpriority                                  *
*                 sc_tinquiry                                   *
*                 sc_lock                                       *
*                 sc_unlock                                     *
*                                                               *
*   Assembly command:   as68 task.s                             *
*                                                               *
*   Compiler : Alcyon C Compiler: Revision 4.7                  *
*   Assembler: Alcyon Assembler:  Revision 4.7                  *
*                                                               *
*****************************************************************


   SECTION     9
   XDEF    _sc_tcreate,_sc_tdelete,_sc_tsuspend,_sc_tresume
   XDEF    _sc_tpriority,_sc_tinquiry,_sc_lock,_sc_unlock

vrtx EQU  Ø          VRTX trap number

* set TASKMODE to 1 for SUPERVISOR MODE

TASKMODE EQU Ø
```

```
*****************************************************************
*                                                               *
*   sc_tcreate(task,tid,pri,errp)                               *
*    int task,tid,pri;  int *errp;                              *
*                                                               *
*   task=8L,   tid=12W,  pri=14W,  errp=16L                     *
*                                                               *
*****************************************************************

_sc_tcreate:
     LINK       A6,#0
     MOVE.L     D3,-(SP)          save registers
     MOVE.L     8(A6),A0          task addr
     MOVE.W     12(A6),D2         tid
     MOVE.W     14(A6),D1         pri
     MOVEQ.L    #TASKMODE,D3
     MOVEQ.L    #$0000,D0         tcreate
     TRAP       #vrtx
     MOVE.L     16(A6),A0         get errp
     MOVE.W     D0,(A0)           store err code
     MOVE.L     (SP)+,D3          restore registers
     UNLK       A6
     RTS
```



```
*************************************************************************
*                                                                       *
*    sc_tdelete(tid/pri,code,errp)                                      *
*        int tid/pri,code;  int *errp;                                  *
*                                                                       *
*    tid/pri=8W,    code=1ØW,  errp=12L                                 *
*                                                                       *
*************************************************************************

_sc_tdelete:
     LINK       A6,#Ø
     MOVE.B     11(A6),D1          code
     LSL.W      #8,D1              shift it left 8 bits
     MOVE.B     9(A6),D1           tid/pri
     MOVEQ.L    #$ØØØ1,DØ          tdelete
     TRAP       #vrtx
     MOVE.L     12(A6),AØ          get errp
     MOVE.W     DØ,(AØ)            store error code
     UNLK       A6
     RTS
```

```
**********************************************************************
*                                                                    *
*   sc_tsuspend(tid/pri,code,errp)                                   *
*      int tid/pri,code;  int *errp;                                 *
*                                                                    *
*   tid/pri=8W,   code=10W,  errp=12L                                *
*                                                                    *
**********************************************************************

_sc_tsuspend:
     LINK      A6,#0
     MOVE.B    11(A6),D1        code
     LSL.W     #8,D1            shift it left 8 bits
     MOVE.B    9(A6),D1         tid/pri
     MOVEQ.L   #$0002,D0        tsuspend
     TRAP      #vrtx
     MOVE.L    12(A6),A0        get errp
     MOVE.W    D0,(A0)          store err code
     UNLK      A6
     RTS
```



```
*************************************************************************
*                                                                       *
*    sc_tresume(tid/pri,code,errp)                                      *
*         int tid/pri, code;  int *errp;                                *
*                                                                       *
*    tid/pri=8W,    code=1ØW,  errp=12L                                 *
*                                                                       *
*************************************************************************

_sc_tresume:
      LINK      A6,#Ø
      MOVE.B    11(A6),D1       code
      LSL.W     #8,D1           shift it left 8 bits
      MOVE.B    9(A6),D1        tid
      MOVEQ.L   #$ØØØ3,DØ       tresume
      TRAP      #vrtx
      MOVE.L    12(A6),AØ       get errp
      MOVE.W    DØ,(AØ)         store err code
      UNLK      A6
      RTS
```

```
**********************************************************************
*                                                                    *
*   sc_tpriority(tid,pri,errp)                                       *
*     int tid,pri;  int *errp;                                       *
*                                                                    *
*   tid=8W,   pri=10W,  errp=12L                                     *
*                                                                    *
**********************************************************************

_sc_tpriority:
     LINK       A6,#0
     MOVE.W     8(A6),D1        tid
     MOVE.W     10(A6),D2       pri
     MOVEQ.L    #$0004,D0       tpriority
     TRAP       #vrtx
     MOVE.L     12(A6),A0       get errp
     MOVE.W     D0,(A0)         store err code
     UNLK       A6
     RTS
```



```
*****************************************************************
*                                                               *
*   int *sc_tinquiry(pinfo,tid,errp)                            *
*       int *pinfo, tid; int *errp;                             *
*                                                               *
*   pinfo=8L,   tid=12W,  errp=14L                              *
*                                                               *
*****************************************************************

_sc_tinquiry:
     LINK       A6,#0
     MOVE.L     D3,-(SP)        save registers
     MOVE.W     12(A6),D1       tid
     MOVEQ.L    #$0005,D0       tinquiry
     TRAP       #vrtx
     MOVE.L     14(A6),A1       get errp
     MOVE.W     D0,(A1)         store err code
     MOVE.L     8(A6),A1        get pinfo
     MOVE.W     D1,(A1)+        store id
     MOVE.W     D2,(A1)+        store pri
     MOVE.W     D3,(A1)         store status
     MOVE.L     A0,D0           return tcb addr
     MOVE.L     (SP)+,D3        restore registers
     UNLK       A6
     RTS
```

```
*************************************************************************
*                                                                       *
*    sc_lock()                                                          *
*                                                                       *
*************************************************************************

_sc_lock:
      LINK        A6,#0
      MOVEQ.L     #$0020,D0         lock
      TRAP        #vrtx
      UNLK        A6
      RTS
```



```
*************************************************************************
*                                                                       *
*    sc_unlock()                                                        *
*                                                                       *
*************************************************************************

_sc_unlock:
      LINK        A6,#Ø
      MOVEQ.L     #$ØØ21,DØ         unlock
      TRAP        #vrtx
      UNLK        A6
      RTS

      END
```

```
*****************************************************************
*                                                               *
* ALCYON C COMPILER                    VERSION 3.Ø   DATE 11/84 *
*                                                               *
* COPYRIGHT 1984, HUNTER & READY, INC.                          *
*                                                               *
*****************************************************************
*                                                               *
*   FILE : memory.s                                             *
*                                                               *
*   VRTX calls :  sc_gblock                                     *
*                 sc_rblock                                     *
*                 sc_pcreate                                    *
*                 sc_pextend                                    *
*                                                               *
*   Assembly command:   as68 memory.s                           *
*                                                               *
*   Compiler : Alcyon C Compiler: Revision 4.7                  *
*   Assembler: Alcyon Assembler:  Revision 4.7                  *
*                                                               *
*****************************************************************


   SECTION     9
   XDEF   _sc_gblock,_sc_rblock,_sc_pcreate,_sc_pextend

vrtx EQU  Ø          VRTX trap number
```

```
*****************************************************************
*                                                               *
*   int *sc_gblock(pid,errp)                                    *
*        int pid;  int *errp;                                   *
*                                                               *
*   pid=8W,   errp=1ØL                                          *
*                                                               *
*****************************************************************

_sc_gblock:
     LINK      A6,#Ø
     MOVE.W    8(A6),D1          get partition #
     MOVEQ.L   #$ØØØ6,DØ         gblock
     TRAP      #vrtx
     MOVE.L    AØ,D1             save AØ
     MOVE.L    1Ø(A6),AØ         get errp
     MOVE.W    DØ,(AØ)           store err code
     MOVE.L    D1,DØ             return ptr to block
     UNLK      A6
     RTS
```

```
*****************************************************************
*                                                               *
*   sc_rblock(pid,blockp,errp)                                  *
*       int pid; int *blockp,*errp;                             *
*                                                               *
*   pid=8W,   blockp=10L,  errp=14L                             *
*                                                               *
*****************************************************************

_sc_rblock:
     LINK       A6,#0
     MOVE.W     8(A6),D1        get partition #
     MOVE.L     10(A6),A0       address of block
     MOVEQ.L    #$0007,D0       rblock
     TRAP       #vrtx
     MOVE.L     14(A6),A0       get errp
     MOVE.W     D0,(A0)         store err code
     UNLK       A6
     RTS
```



```
*****************************************************************
*                                                               *
*   sc_pcreate(pid,paddr,psize,bsize,errp)                      *
*        int pid; int *paddr,*psize; int bsize; int *errp;      *
*                                                               *
*   pid=8W,   paddr=10L,  psize=14L,  bsize=18W,  errp=20L      *
*                                                               *
*****************************************************************

_sc_pcreate:
     LINK      A6,#0
     MOVE.L    D3,-(SP)        save registers
     MOVE.W    8(A6),D1        pid
     MOVE.L    10(A6),A0       paddr
     MOVE.L    14(A6),D2       psize
     MOVE.W    18(A6),D3       bsize
     MOVEQ.L   #$0022,D0       pcreate
     TRAP      #vrtx
     MOVE.L    20(A6),A0       get errp
     MOVE.W    D0,(A0)         store err code
     MOVE.L    (SP)+,D3        restore registers
     UNLK      A6
     RTS
```



```
*****************************************************************
*                                                               *
*   sc_pextend(pid,paddr,psize,errp)                            *
*       int pid; int *paddr, *psize, *errp;                     *
*                                                               *
*   pid=8W,   paddr=10L,  psize=14L,  errp=18L                  *
*                                                               *
*****************************************************************

_sc_pextend:
     LINK       A6,#0
     MOVE.W     8(A6),D1        pid
     MOVE.L     10(A6),A0       paddr
     MOVE.L     14(A6),D2       psize
     MOVEQ.L    #$0023,D0       pextend
     TRAP       #vrtx
     MOVE.L     18(A6),A0       get errp
     MOVE.W     D0,(A0)         store err code
     UNLK       A6
     RTS

     END
```



```
*************************************************************************
*                                                                       *
* ALCYON C COMPILER                        VERSION 3.Ø   DATE 11/84     *
*                                                                       *
* COPYRIGHT 1984, HUNTER & READY, INC.                                  *
*                                                                       *
*************************************************************************
*                                                                       *
*   FILE : sync.s                                                       *
*                                                                       *
*   VRTX calls :  sc_post                                               *
*                 sc_pend                                               *
*                 sc_accept                                             *
*                 sc_qpost                                              *
*                 sc_qpend                                              *
*                 sc_qaccept                                            *
*                 sc_qcreate                                            *
*                 sc_qinquiry                                           *
*                                                                       *
*   Assembly command:    as68 sync.s                                    *
*                                                                       *
*   Compiler : Alcyon C Compiler: Revision 4.7                          *
*   Assembler: Alcyon Assembler:  Revision 4.7                          *
*                                                                       *
*************************************************************************


   SECTION     9
   XDEF    _sc_post,_sc_pend,_sc_accept,_sc_qpost,_sc_qpend
   XDEF    _sc_qaccept,_sc_qcreate,_sc_qinquiry

vrtx EQU  Ø          VRTX trap number
```

```
**********************************************************************
*                                                                    *
*   sc_post(mboxp,msg,errp)                                          *
*     char *mboxp, *msg; int *errp;                                  *
*                                                                    *
*   mboxp=8L,   msg=12L,  errp=16L                                   *
*                                                                    *
**********************************************************************

_sc_post:
     LINK       A6,#Ø
     MOVE.L     8(A6),AØ        get mailbox address
     MOVE.L     12(A6),D1       get msg
     MOVEQ.L    #$ØØØ8,DØ       post
     TRAP       #vrtx
     MOVE.L     16(A6),AØ       get errp
     MOVE.W     DØ,(AØ)         store err code
     UNLK       A6
     RTS
```



```
*****************************************************************
*                                                               *
*   char *sc_pend(mboxp,timeout,errp)                           *
*           char *mboxp; long timeout; int *errp;               *
*                                                               *
*   mboxp=8L,   timeout=12L,   errp=16W                         *
*                                                               *
*****************************************************************

_sc_pend:
     LINK       A6,#Ø
     MOVE.L     8(A6),AØ        get mailbox address
     MOVE.L     12(A6),D1       get timeout value
     MOVEQ.L    #$ØØØ9,DØ       pend
     TRAP       #vrtx
     MOVE.L     16(A6),AØ       get errp
     MOVE.W     DØ,(AØ)         store err code
     MOVE.L     D1,DØ           return msg
     UNLK       A6
     RTS
```

```
*****************************************************************
*                                                               *
*   char *sc_accept(mboxp,errp)                                 *
*         char *mboxp; int *errp;                               *
*                                                               *
*   mboxp=8L,   errp=12L                                        *
*                                                               *
*****************************************************************

_sc_accept:
     LINK A6,#0
     MOVE.L    8(A6),A0        address of mailbox into A0
     MOVEQ.L   #$0025,D0       accept
     TRAP      #vrtx
     MOVE.L    12(A6),A0       get errp
     MOVE.W    D0,(A0)         store err code
     MOVE.L    D1,D0           return msg
     UNLK A6
     RTS
```



```
*********************************************************************
*                                                                   *
*   sc_qpost(qid,msg,errp)                                          *
*        int qid; char *msg; int *errp;                             *
*                                                                   *
*   qid=8W,   msg=10L,  errp=14L                                    *
*                                                                   *
*********************************************************************

_sc_qpost:
     LINK      A6,#0
     MOVE.W    8(A6),D1        get qid
     MOVE.L    10(A6),D2       get msg
     MOVEQ.L   #$0026,D0       qpost
     TRAP      #vrtx
     MOVE.L    14(A6),A0       get errp
     MOVE.W    D0,(A0)         store err code
     UNLK      A6
     RTS
```

```
**********************************************************************
*                                                                    *
*   char *sc_qpend(qid,timeout,errp)                                 *
*         int qid; long timeout; int *errp;                          *
*                                                                    *
*   qid=8W,   timeout=1ØL,   errp=14L                                *
*                                                                    *
**********************************************************************

_sc_qpend:
     LINK      A6,#Ø
     MOVE.W    8(A6),D1        get qid
     MOVE.L    1Ø(A6),D2       get timeout value
     MOVEQ.L   #$ØØ27,DØ       qpend
     TRAP      #vrtx
     MOVE.L    14(A6),AØ       get errp
     MOVE.W    DØ,(AØ)         store err code
     MOVE.L    D2,DØ           return msg
     UNLK      A6
     RTS
```



```
*********************************************************************
*                                                                   *
*   char *sc_qaccept(qid,errp)                                      *
*           int qid; int *errp;                                     *
*                                                                   *
*   qid=8W,   errp=1ØL                                              *
*                                                                   *
*********************************************************************

_sc_qaccept:
     LINK       A6,#Ø
     MOVE.W     8(A6),D1          get qid
     MOVEQ.L    #$ØØ28,DØ         qaccept
     TRAP       #vrtx
     MOVE.L     1Ø(A6),AØ         get errp
     MOVE.W     DØ,(AØ)           store err code
     MOVE.L     D2,DØ             return msg
     UNLK       A6
     RTS
```

```
*****************************************************************
*                                                               *
*   sc_qcreate(qid,size,errp)                                   *
*      int qid, size; int *errp;                                *
*                                                               *
*   qid=8W,   size=10W,   errp=12L                              *
*                                                               *
*****************************************************************

_sc_qcreate:
     LINK      A6,#0
     MOVE.W    8(A6),D1        get qid
     MOVE.W    10(A6),D2       get size
     MOVEQ.L   #$0029,D0       qcreate
     TRAP      #vrtx
     MOVE.L    12(A6),A0       get errp
     MOVE.W    D0,(A0)         store err code
     UNLK      A6
     RTS
```



```
*****************************************************************
*                                                               *
*   char *sc_qinquiry(qid,countp,errp)                          *
*       int qid; int *countp,*errp;                             *
*                                                               *
*   qid=8W,    countp=10L,     errp=14L                         *
*                                                               *
*****************************************************************

_sc_qinquiry:
     LINK      A6,#0
     MOVE.L    D3,-(SP)        save register
     MOVE.W    8(A6),D1        get qid
     MOVEQ.L   #$002A,D0       qinquiry
     TRAP      #vrtx
     MOVE.L    14(A6),A0       get pointer to error code number
     MOVE.W    D0,(A0)         store err code
     MOVE.L    10(A6),A0       get pointer to count
     MOVE.W    D3,(A0)         store count
     MOVE.L    D2,D0           return msg
     MOVE.L    (SP)+,D3        restore registers
     UNLK      A6
     RTS

     END
```

```
*****************************************************************
*                                                               *
* ALCYON C COMPILER                    VERSION 3.Ø   DATE 11/84 *
*                                                               *
* COPYRIGHT 1984, HUNTER & READY, INC.                          *
*                                                               *
*****************************************************************
*                                                               *
*   FILE : clock.s                                              *
*                                                               *
*   VRTX calls :  sc_gtime                                      *
*                 sc_stime                                      *
*                 sc_delay                                      *
*                 sc_tslice                                     *
*                                                               *
*   Assembly command:    as68 clock.s                           *
*                                                               *
*   Compiler : Alcyon C Compiler: Revision 4.7                  *
*   Assembler: Alcyon Assembler:  Revision 4.7                  *
*                                                               *
*****************************************************************


   SECTION     9
   XDEF    _sc_gtime,_sc_stime,_sc_delay,_sc_tslice

vrtx EQU  Ø          VRTX trap number
```

```
*****************************************************************
*                                                               *
*   long sc_gtime()                                             *
*                                                               *
*****************************************************************

_sc_gtime:
     LINK       A6,#Ø
     MOVEQ.L    #$ØØØA,DØ        gtime
     TRAP       #vrtx
     MOVE.L     D1,DØ            return value
     UNLK       A6
     RTS
```

```
*****************************************************************
*                                                               *
*    sc_stime(time)                                             *
*       long time;                                              *
*                                                               *
*    time=8L                                                    *
*                                                               *
*****************************************************************

_sc_stime:
     LINK       A6,#Ø
     MOVE.L     8(A6),Dl          get time
     MOVEQ.L    #$ØØØB,DØ         stime
     TRAP       #vrtx
     UNLK       A6
     RTS
```



```
**********************************************************************
*                                                                    *
*   sc_delay(ticks)                                                  *
*     long ticks;                                                    *
*                                                                    *
*   ticks=8L                                                         *
*                                                                    *
**********************************************************************

_sc_delay:
     LINK      A6,#Ø
     MOVE.L    8(A6),D1        get # of ticks
     MOVEQ.L   #$ØC,DØ         tdelay
     TRAP      #vrtx
     UNLK      A6
     RTS
```

```
**********************************************************************
*                                                                    *
*    sc_tslice(ticks)                                                *
*        int ticks;                                                  *
*                                                                    *
*    ticks=8W                                                        *
*                                                                    *
**********************************************************************

_sc_tslice:
     LINK      A6,#Ø
     CLR.L     Dl
     MOVE.W    8(A6),Dl          get # of ticks
     MOVEQ.L   #$Ø015,DØ         tslice
     TRAP      #vrtx
     UNLK      A6
     RTS

     END
```

```
*****************************************************************
*                                                               *
* ALCYON C COMPILER                    VERSION 3.Ø   DATE 11/84 *
*                                                               *
* COPYRIGHT 1984, HUNTER & READY, INC.                          *
*                                                               *
*****************************************************************
*                                                               *
*   FILE : chario.s                                             *
*                                                               *
*   VRTX calls :  sc_getc                                       *
*                 sc_putc                                       *
*                 sc_waitc                                      *
*                                                               *
*   Assembly command:   as68 chario.s                           *
*                                                               *
*   Compiler :  Alcyon C Compiler: Revision 4.7                 *
*   Assembler:  Alcyon Assembler:  Revision 4.7                 *
*                                                               *
*****************************************************************


   SECTION     9
   XDEF   _sc_getc,_sc_putc,_sc_waitc

vrtx EQU  Ø      .    VRTX trap number
```

```
*****************************************************************
*                                                               *
*   char sc_getc()                                              *
*                                                               *
*****************************************************************

_sc_getc:
     LINK       A6,#Ø
     MOVEQ.L    #$ØØØD,DØ       getc
     TRAP       #vrtx
     CLR.L      DØ
     MOVE.B     D1,DØ           return char, zero-extended
     UNLK       A6
     RTS
```

```
**********************************************************************
*                                                                    *
*   sc_putc(chr)                                                     *
*     char chr;                                                      *
*                                                                    *
*   char=8W                                                          *
*                                                                    *
**********************************************************************

_sc_putc:
     LINK       A6,#0
     MOVE.B     9(A6),D1          get char from stack
     MOVEQ.L    #$000E,D0         putc
     TRAP       #vrtx
     UNLK       A6
     RTS
```



```
*****************************************************************
*                                                               *
*    sc_waitc(char,errp)                                        *
*       char chr; int *errp;                                    *
*                                                               *
*    char=8W,  errp=1ØL                                         *
*                                                               *
*****************************************************************

_sc_waitc:
      LINK      A6,#Ø
      MOVE.B    9(A6),D1        get char
      MOVEQ.L   #$ØØØF,DØ       waitc
      TRAP      #vrtx
      MOVE.L    1Ø(A6),AØ       get errp
      MOVE.W    DØ,(AØ)         store err code
      UNLK      A6
      RTS

      END
```



```
*****************************************************************
*                                                               *
* ALCYON C COMPILER                    VERSION 3.Ø   DATE 11/84 *
*                                                               *
* COPYRIGHT 1984, HUNTER & READY, INC.                          *
*                                                               *
*****************************************************************
*                                                               *
*   FILE : component.s                                          *
*                                                               *
*   VRTX calls :  sc_call                                       *
*                                                               *
*   Assembly command:   as68 component.s                        *
*                                                               *
*   Compiler :  Alcyon C Compiler: Revision 4.7                 *
*   Assembler:  Alcyon Assembler:  Revision 4.7                 *
*                                                               *
*****************************************************************


   SECTION     9
   XDEF   _sc_call

vrtx EQU  Ø         VRTX trap number
```

```
*****************************************************************
*                                                               *
*   sc_call(fcode,pktp,errp)                                    *
*       char *pktp;  int fcode,*errp;                           *
*                                                               *
*   fcode=8L      pktp=10L      errp=14L                        *
*                                                               *
*****************************************************************

_sc_call:
     LINK       A6,#0
     MOVE.L     10(A6),A0       address of parameter packet
     MOVE.W     8(A6),D0        function code
     EXT.L      D0
     TRAP       #vrtx           call VRTX
     MOVE.L     14(A6),A0       get error pointer
     MOVE.W     D0,(A0)         and return error code
     UNLK       A6
     RTS

     END
```


USING THE INTERFACE LIBRARY

Once the assembly files are successfully entered into the library, it is relatively easy to interface your C language modules with the library. This section details any special actions that must be taken to ensure proper execution. Please note that later revisions of the compiler or other supplied software may make these steps obsolete. We show these examples to demonstrate the means whereby the interface library was tested at our facility. Please check the Alcyon revision documentation for the latest information.

COMPILING INSTRUCTIONS

The VRTX interface calls may be placed anywhere in your high-level language modules. Each call must conform to the format as described in Appendix A of the VRTX C Interface Library User's Guide. The C language assumes that all functions return an integer value, unless specifically defined. In most cases, therefore, it is not necessary to define the VRTX functions. However, the following functions should be included at the beginning of each C module where they are accessed:

```
extern long sc_gtime();
extern char sc_getc();
extern char *sc_gblock(),*sc_pend(),*sc_accept();
extern char *sc_tinquiry(),*sc_qinquiry(),*sc_qpend(),*sc_qaccept();
```

All other VRTX functions do not return a value.

If desired, the following file may be included in your high-level C language module. This file is a description of each of the VRTX interface calls and includes the necessary function definitions for the C language. Placing the following lines in a file called "DEFS," for example, and then always including this file in any high-level language module that accesses the VRTX interface library ensures proper definition for all of the calls. The file may be included in the module by placing the line:

```
#include "DEFS"
```

in the definition area of your C module. The symbol # must begin in column one of the line.

```
/*************************************************************/
/* Include file: DEFS                                        */
/* Used in C modules for defining VRTX functions             */
/* Place the following line in the definition area of        */
/* the language module :                                     */
/*                                                           */
/* #include "DEFS"                                          */
/*************************************************************/
/* These routines do not return values and are listed for    */
/* summary purposes                                          */
/*************************************************************/
/*                                                           */
/* sc_tcreate(task,tid,pri &err);        // Task Create      */
/* sc_tdelete(tid/pri,code,&err);        // Task Delete      */
/* sc_tsuspend(tid/pri,code,&err);       // Task Suspend     */
/* sc_tresume(tid/pri,code,&err);        // Task Resume      */
/* sc_tpriority(tid,pri,&err);   // Task Priority Change     */
/*                                                           */
/* sc_lock();                    //Disable Task Rescheduling*/
/* sc_unlock();                  //Enable Task Rescheduling */
/*                                                           */
/* sc_rblock(pid,block,&err);    // Release Memory Block     */
/* sc_pcreate(pid,paddr,psize,bsize,&err);                   */
/*                               // Create Memory Partition */
/* sc_pextend(pid,paddr,psize,&err);                         */
/*                               // Extend Memory Partition */
/* sc_post(&mbox,msg,&err);      // Post Message             */
/* sc_qpost(qid,msg,&err);       // Post Message to Queue    */
/* sc_qcreate(qid,qsize,&err);   // Create Message Queue     */
/*                                                           */
/* sc_stime();                   // Set Time                 */
/* sc_delay(timeout);            // Task Delay               */
/* sc_tslice(ticks);             // Enable Round-Robin       */
/* sc_putc(char);                // Put Character            */
/* sc_waitc(char,&err);          // Wait Character           */
/*                                                           */
/* sc_call(fcode,&pkt,&err);     // Call a Component         */
/*************************************************************/
/* The following routines return values                      */
/*************************************************************/
/*                                                           */
char  *sc_gblock();                     /* Get Memory Block */
char  *sc_tinquiry();                   /* Task Inquiry     */
char  *sc_pend();                       /* Pend for Message */
char  *sc_accept()                      /* Accept Message   */
char  *sc_qpend();                 /* Pend Message from Queue*/
char  *sc_qaccept();              /*Accept Message from Queue*/
char  *sc_qinquiry();                   /* Queue Inquiry    */
long   sc_gtime();                      /* Get Time         */
char   sc_getc();                       /* Get Character    */
```

LINKING INSTRUCTIONS

Once the C module or modules are successfully compiled, the linker is used to integrate the Interface Library. Again, it is best to create a linker command file to link your programs together. The linker for the Alcyon tools is accessed by typing **lo68** on the command line. An example in which the linker combines the interface modules with an object module called demo.o follows:

```
lo68 bsp.o demo.o  vclib
sendc68 c.out > demo.hex
```

bsp.o is the board support routine for the target system. **demo.o** is the application object module to be included in the executable image. **vclib** is the VRTX interface library. **demo.hex** is the name of the output file created by lo68 which is the file downloaded to the target system for execution.

INPUT/OUTPUT ROUTINES

The standard C library provides functions for reading and writing one character at a time. These routines are **getchar** and **putchar.** The function **getchar** fetches the next input character each time it is called and returns that character as its value. The function **putchar** is the complement of **getchar**; it prints a character on some output medium, one character at a time.

Because these routines fit so well into the VRTX system structures, we provide the high-level language equivalent of **getchar** and **putchar** for use with VRTX's internal input and output buffers. These routines make your programs even more portable.

INSTRUCTIONS FOR COMPILING

The listing found at the end of this section contains the high-level C code to implement **getchar** and **putchar.** Place this routine in a file called stdio.c and compile it by typing the command:

```
c68 -c stdio.c
```

LINKING WITH THE I/O ROUTINES

Once the file is successfully compiled, it may be inserted into a library or it may be linked with your application program. Inserting this file into a library is similar to the creation of the VRTX interface library file and you should refer to the **Assembly Instructions** section for further information.

When you choose not to make the stdio routine a library, you link the compiled stdio.c file with your application. The following link command should be used:

```
lo68 bsp.o demo.o stdio.o vclib
sendc68 c.out > demo.hex
```

The listing of the stdio.c file appears on the next page.

```
/*************************************************************/
/*                                                           */
/* File : stdio.c                                            */
/*                                                           */
/* Getchar and Putchar: I/O Routines for use with VRTX       */
/*                                                           */
/*************************************************************/


char sc_getc();
sc_putc ();

/*************************************************************/
/* putchar                                                   */
/*************************************************************/

putchar(c)
char c;
{

if (c == '\n') sc_putc('\r');
sc_putc(c);

}
/*************************************************************/
/* getchar                                                   */
/*************************************************************/

char getchar()
{

char c;

c = sc_getc();
if (c == '\r') c = '\n';
putchar(c);
return(c);

}
/*************************************************************/
```