CET BASIC Communication Port Facilities
W32 App Builder Version
Programmer's Guide

I. Introduction

The standard release of CET W32 Application Builder contains support for communications ports under Windows NT, Windows 95, and Windows 3.11/WFWG. Opening and using a communications port is achieved by the same means as other versions of CET BASIC. W32 App Builder also provides a number of native functions for the control of the communications ports, and for determining their status.

II. Using the OPEN Statement for COM Ports

CET BASIC recognizes the special filename “COM” and “COMx”, where x is a number designating the communication port whose opening is required. In CET BASIC, a communications port may be opened as:

```
OPEN #1:”COM”, UPDATE SEQUENTIAL
```

The affect of this statement is to open COM1 for input and output operations. Once a COM file is opened all relevant I/O statements may be used with it, for example:

```
LINPUT #1: A$
INPUT #1: A$, B%, C
GET #1: A%
PUT #1: B%
PRINT #1: Q$
```

The byte stream presented to the communications port is identical to that which would be presented to a file or printer with B_MAPPRT option enabled. The default behavior of the PRINT statement is to append a CR/NL to each line, when it is terminated with out a semicolon. This can be prevented, and only a CR will be output if the RAWMODE option is included in the OPEN statement.

III. Setting and Getting Communications Port Attributes

When a communications port is opened, it is initialized with a default baudrate (19200), a default parity (None), a default word length (8), and a default number of stop bits (2). These communications parameters can be changed before opening a communications port by using the cComSetAttributes function, as:

```
CALL cComSetAttributes(CHANNEL%, BAUDRATE%, PARITY$, WORDLEN%, STOPBITS%)
```

The BAUDRATE% parameter can be of any numeric type, and is the actual baud rate (e.g., 19200).
The PARITY% parameter must be ‘N’ for none, ‘E’ for even, ‘O’ for odd, ‘M’ for mark, and ‘S’ for space.
The WORDLEN% parameter can be of any numeric type, and should be 5, 6, 7, or 8.
The STOPBITS% parameter can be of any numeric type, and should be 1 or 2.

You may wish to override the default flow control that is established in the Control Panel definition for a communications port. The flow control can be set by calling one or more of the following routines, and using a one (1) to enable flow control and a zero (0) to disable it for the particular flow control regimen.

```
CALL cComUseDtrDsr(CHANNEL%, DTRDSR%)
CALL cComUseRtsCts(CHANNEL%, RTSCTS%)
CALL cComUseXonXoff(CHANNEL%, XONXOFF%)
```

It is possible to enable or disable Data-Terminal-Ready by using the cComSetDtr() function. The DTR% variable should be one (1) for enable, and zero (0) for disable, and the syntax is:

CALL cComSetDtr(CHANNEL%, DTR%)

Four functions are available for determining the status of flow-control settings. These functions are:

CALL cComGetCts(CHANNEL%, ADDROF(CTS%))
CALL cComGetDsr(CHANNEL%, ADDROF(DSR%))
CALL cComGetCd(CHANNEL%, ADDROF(CD%))

Note that anywhere that an integer variable has been used in the examples above, any numeric variable or value could be used, including decimal, binary or byte variable types.

IV. Some Useful Functions

Some of the functions of the CET W32 Communications Library have been added to base support for W32 App Builder. These functions include:

call cComIsRxReady(CHANNEL%, ADDROF(STATUS%))

The cComIsRxReady function can be used to determine if any input is available on specific communications port. A one (1) is returned for the value of STATUS% if there are available characters for reading, and a zero is returned otherwise. When STATUS% is non-zero, it is guaranteed that at least one character is available for reading. This means that a GET statement will return a (non-zero) value. There may, however, not be enough characters available to completely and immediately satisfy a LINPUT or INPUT on the communications channel.

CALL cBlockSender(CHANNEL%, BLOCK%)

The cBlockSender function can be used to stop the sender on a communications channel. The means of achieving this blocking of further incoming characters depends upon the flow-control set for the open channel. .

IV. Data Transfer Functions

The Communications Library functions which transmit and receive files include protocol-related routines for Kermit, Zmodem, Xmode, and Ymodem. The syntax for the File Receive routines are :

call cComKermitReceive(port%,SubAddrOf(MSGSUB), SubAddrOf(XFRSUB),ADDROF(nrcvd%))
call cComZmodemReceive(port%,SubAddrOf(MSGSUB), SubAddrOf(XFRSUB),ADDROF(nrcvd%))
call cComXmodemReceive(port%,FilNamLst$,SubAddrOf(MSGSUB), SubAddrOf(XFRSUB),ADDROF(nrcvd%))
call cComYmodemReceive(port%,SubAddrOf(MSGSUB), SubAddrOf(XFRSUB),ADDROF(nrcvd%))

where

port% is the port number opened by an OPEN statement
FilNamLst$ is a list of file names, separated by commas or white space
MSGSUB is the name of a CET BASIC (or C) subroutine that should be called when a message is available from the routine called and should be zero (0) if you do not want to be notified.
XFRSUB is the name of a CET BASIC (or C) subroutine that that should be called every time a block is received by the routine called and should be zero (0) if you do not want to be notified.
Nrcvd% is the number of files received

The syntax for the File Receive routines are :

call cComKermitSend(port%, FilNamLst$,SubAddrOf(MSGSUB), SubAddrOf(XFRSUB),ADDROF(nsent%))
call cComZmodemSend(port%,FilNamLst$,SubAddrOf(MSGSUB), SubAddrOf(XFRSUB),ADDROF(nsent%))
call cComXmodemSend(port%,FilNamLst$,SubAddrOf(MSGSUB), SubAddrOf(XFRSUB),ADDROF(nsent%))
call cComYmodemSend(port%,FilNamLst$,SubAddrOf(MSGSUB), SubAddrOf(XFRSUB),ADDROF(nsent%))

where

port% is the port number opened by an OPEN statement
FilNamLst$ is a list of file names, separated by commas or white space, and can include wild cards
MSGSUB is the name of a CET BASIC (or C) subroutine that should be called when a message is available from the routine called and should be zero (0) if you do not want to be notified.
XFRSUB is the name of a CET BASIC (or C) subroutine that that should be called every time a block is sent by the routine called, and should be zero (0) if you do not want to be notified.
Nrcvd% is the number of files sent

You will note that in our sample routine CETBBS.B, we have set the SubAddrOf() addresses to zero. We are still testing this feature, and will introduce CET Subroutines with parameters to better accommodate routines such as these.

Note that the file(s) which are received by cComXxxReceived() routines come into the current directory, unless the sending file name is a complete path name. The same is true for the send routines.

The last parameter, Nrcvd% or Nsent%, will return the number of files transferred. This parameter will be a negative number which is an error code if the transfer fails.

A sample program segment using the Kermit protocol is shown below.

```
 10      ToFile$ = "JLBTEST.FIL"
 20      REM
 30      REM This routine should be entered with
 40      REM ToFile$ set to the filename desired
 50      REM
 60      REM
 70      REM First make the directory if it isn't here
 80      REM
 90      ON ERROR goto AlreadyHere
100      call Bmkdir(".\kmtxfer")
110      goto NextStep
120 AlreadyHere:
130      resume NextStep
140 NextStep:
150      REM
160      REM Change directories for the kermit receive
170      REM
180      call Bchdir("kmtxfer")
190      REM
200      REM Put the Kermit Receive Here
210      REM
220      REM call cComKermitReceive ... with parameters
230      REM
231      REM !we will emulate by copying a file into the directory
232      REM
         CALL Bcopy("..\testfile  kmtfile")
235      REM
240
250      REM
260      REM Find out what got transferred
270      REM
280
290      CSH("DIR > ..\Dir.Lst")
300
310      REM
320      REM Find out what got transferred
330      REM
340
350      REM The directory listing will look like
360
370      REM  Volume in drive D is CETDEV
380      REM  Volume Serial Number is 443B-F946
390      REM  Directory of D:\transformit\kmtxfer
400      REM
410      REM .              <DIR>        01-09-97  1:31p .
420      REM ..             <DIR>        01-09-97  1:31p ..
430   REM SOMEFILE XXX             0  01-09-97  1:48p TOUCHED.TCH
440      REM          1 file(s)            330 bytes
450
460      REM
470      REM
480
490      REM Open the file and read back the file name
500
510      OPEN #20:"..\Dir.Lst", input sequential
520      WHILE 1
530            LINPUT #20: L$
540            IF EOF(20)
550                  Print "Whoa, didn't get the file!"
560                  quit
```

```
570            IFEND
575            IF TRIM(L$)="" THEN CONTINUE
580            IF L$[1:1]<>" " AND L$[1:1]<>"."
585                  FirstPart$ = TRIM(L$[1:8])
586                  SecndPart$ = TRIM(L$[10:12])
590                  FromFile$= FirstPart$
600                  IF SecndPart$<> ""
                     FromFile$ = FirstPart$+"."+SecndPart$
               IFEND
620                  BREAK
630            IFEND
640      WEND

645      ON ERROR goto CopyFailed
646      call Bcopy(FromFile$+" ..\"+ToFile$)
647      goto CopyOK
649 CopyFailed:
650      print "Copy Failed!"
655      QUIT

656 CopyOK:
660      call Berase(FromFile$)
670      call Bchdir("..")
```