Sender StateChart for XMODEM file-transfer protocol.
    November, 2020
Real events are either
    KB_C (cancel command received from keyboard, higher priority)
    SER (serial character recieved, lower priority),
    TM (timeout occurred)

Unless a fast simulation has been chosen ...

TM_VL (Very Long timeout) gives 60 seconds
TM_SOH (normal timeout waiting for SOH) gives 10 seconds
TM_CHAR (inter-character timeout) gives 1 second
TM_2CHAR gives a period longer than the
        inter-character timeout of 1 sec.

CompleteSender_TopLevel

[Entry]
if (!(ctx.openErr = (ctx.openFileToTransfer() == -1))) {
    ctx.Crcflg=true;
    ctx.prep1stBlk();
    ctx.errCnt=0;
    ctx.firstCrcBlk=true;
    ctx.KbCan = false;
    ctx.tm(TM_VL);
}
[Exit]

ctx.can8() should not be called immediately
after the crc16/checksum for a block has
been written to the medium, or some CAN
characters may be purged.

onEvent(KB_C)
[!ctx.KbCan]
/ctx.KbCan = true;
ctx.tmRed(TM_VL - TM_2CHAR);

H SERcancelable
[Entry]
[Exit]

START
[Entry]
[Exit]

onEvent(SER)
[!ctx.openErr & (c==NAK || c=='C') & ctx.bytesRd]
/if (c==NAK) {ctx.Crcflg=false;
    ctx.cs1stBlk();
    ctx.firstCrcBlk=false;}
ctx.sendBlkPrepNext();
ctx.tm(TM_VL);

onEvent(SER)
[(c==ACK) & ctx.bytesRd & !ctx.KbCan]
/ctx.sendBlkPrepNext();
ctx.errCnt=0; ctx.firstCrcBlk=false;
ctx.tm(TM_VL);

ACKNAK
[Entry]
[Exit]

onEvent(SER)
[(c==NAK || (c=='C' & ctx.firstCrcBlk)) & (ctx.errCnt
< errB) & !ctx.KbCan]
/ctx.resendBlk();
ctx.errCnt++; ctx.tm(TM_VL);

onEvent(SER)
[(c==ACK) & !ctx.bytesRd & !ctx.KbCan]
/ctx.sendByte(EOT);ctx.errCnt=0;
ctx.tm(TM_VL);ctx.firstCrcBlk=false;

onEvent(SER)
[!ctx.openErr & !ctx.bytesRd &
== NAK) || (c == 'C')]
/if (c==NAK) {ctx.firstCrcBlk=false;}
ctx.sendByte(EOT); ctx.tm(TM_VL);

onEvent(SER)
[c != CAN & !ctx.KbCan]
/ctx.tmPop();

EOT1
[Entry]
[Exit]

onEvent(SER)
[c==NAK & !ctx.KbCan]
/ctx.sendByte(EOT);
ctx.errCnt=0;
ctx.tm(2*TM_SOH);

EOTEOT
[Entry]
[Exit]

onEvent(SER)
[c==NAK & (ctx.errCnt < errB)
& !ctx.KbCan]
/ctx.sendByte(EOT);
ctx.errCnt++;

onEvent(SER)
[(c=='C' & ctx.firstCrcBlk)
& (ctx.errCnt < errB) &
!ctx.KbCan]
/ctx.sendByte(EOT);
ctx.tm(TM_VL);
ctx.errCnt++;

onEvent(TM)
/ctx.closeTransferredFile();
ctx.result="Timeout waiting
for ACK of 2nd EOT";

onEvent(TM)
[!ctx.KbCan]
/ctx.tmPop();

onEvent(SER)
[c == CAN]
/ctx.tmPush(TM_CHAR);

CAN
[Entry]
[Exit]

onEvent(SER)
[c==ACK]
/ctx.sendByte(EOT);
ctx.closeTransferredFile();
ctx.result="1st EOT ACK'd";

onEvent(SER)
[c==ACK]
/ctx.closeTransferredFile();
ctx.result="Done";

onEvent(KB_C)
/ctx.KbCan=true;

onEvent(TM)
/ctx.can8();
ctx.closeTransferredFile();
if (ctx.KbCan)
    ctx.result="KbCancelled";
else
    ctx.result="Timeout";

onEvent(SER)
[c==CAN]
/ctx.clearCan();
ctx.closeTransferredFile();
ctx.result="RcvCancelled";

onEvent(SER)
[ctx.KbCan]
/ctx.can8();
ctx.closeTransferredFile();
ctx.result="KbCancelled";

onEvent(SER)
[ctx.openErr & ((c == NAK) || (c == 'C'))]
/ctx.can8();
ctx.result="OpenError";

Due to a bug in SmartState studio, in
this StateChart, please interpret & as
&& -- a logical AND. Do not interpret it
as a bitwise AND as you normally
would.

onEvent(KB_C)
/if (!ctx.openErr) {ctx.closeTransferredFile(); ctx.result="KbCancelledOpenErr"; }
 else ctx.result="KbCancelledFromSTART";

onEvent(SER)
[((c=='C' & ctx.firstCrcBlk) || c==NAK) & !ctx.KbCan]
/ctx.can8();
ctx.closeTransferredFile();
ctx.result="ExcessiveNAKsOrCs";