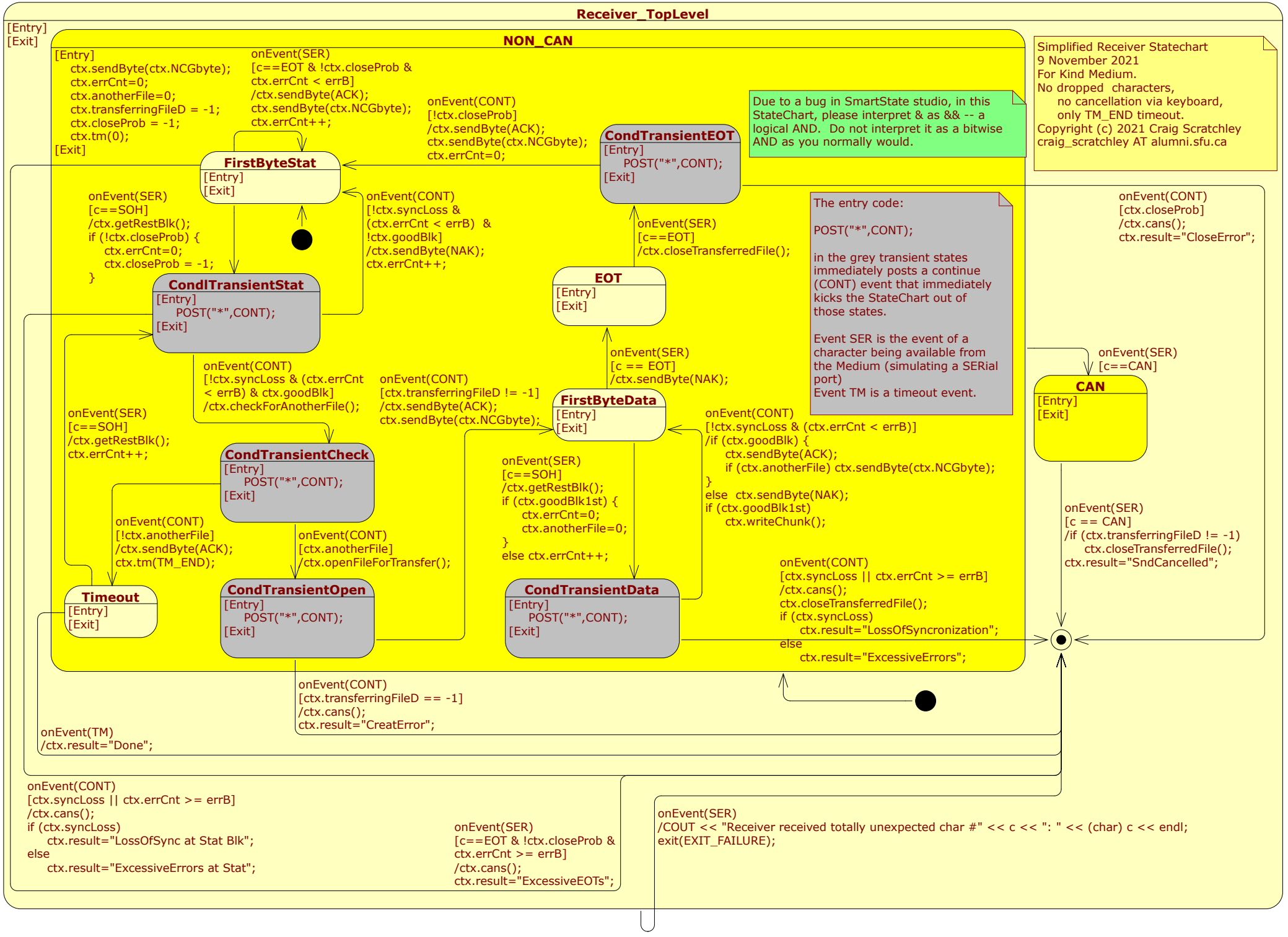
Receiver_TopLevel
[Entry]
[Exit]
NON_CAN
[Entry]
ctx.sendByte(ctx.NCGbyte);
ctx.errCnt=0;
ctx.anotherFile=0;
ctx.transferringFileD = -1;
ctx.closeProb = -1;
ctx.tm(0);
[Exit]
onEvent(SER)
[c==EOT & !ctx.closeProb &
ctx.errCnt < errB]
/ctx.sendByte(ACK);
ctx.sendByte(ctx.NCGbyte);
ctx.errCnt++;
onEvent(CONT)
[!ctx.closeProb]
/ctx.sendByte(ACK);
ctx.sendByte(ctx.NCGbyte);
ctx.errCnt=0;
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.
Simplified Receiver Statechart
9 November 2021
For Kind Medium.
No dropped characters,
no cancellation via keyboard,
only TM_END timeout.
Copyright (c) 2021 Craig Scratchley
craig_scratchley AT alumni.sfu.ca
FirstByteStat
[Entry]
[Exit]
CondTransientEOT
[Entry]
POST("*",CONT);
[Exit]
onEvent(SER)
[c==SOH]
/ctx.getRestBlk();
if (!ctx.closeProb) {
ctx.errCnt=0;
ctx.closeProb = -1;
}
onEvent(CONT)
[!ctx.syncLoss &
(ctx.errCnt < errB) &
!ctx.goodBlk]
/ctx.sendByte(NAK);
ctx.errCnt++;
onEvent(SER)
[c==EOT]
/ctx.closeTransferredFile();
onEvent(CONT)
[ctx.closeProb]
/ctx.cans();
ctx.result="CloseError";
The entry code:
POST("*",CONT);
in the grey transient states immediately posts a continue (CONT) event that immediately kicks the StateChart out of those states.
Event SER is the event of a character being available from the Medium (simulating a SERial port)
Event TM is a timeout event.
CondITransientStat
[Entry]
POST("*",CONT);
[Exit]
EOT
[Entry]
[Exit]
onEvent(SER)
[c==CAN]
CAN
[Entry]
[Exit]
onEvent(CONT)
[!ctx.syncLoss & (ctx.errCnt
< errB) & ctx.goodBlk]
/ctx.checkForAnotherFile();
onEvent(CONT)
[ctx.transferringFileD != -1]
/ctx.sendByte(ACK);
ctx.sendByte(ctx.NCGbyte);
onEvent(SER)
[c == EOT]
/ctx.sendByte(NAK);
FirstByteData
[Entry]
[Exit]
onEvent(SER)
[c==SOH]
/ctx.getRestBlk();
ctx.errCnt++;
CondTransientCheck
[Entry]
POST("*",CONT);
[Exit]
onEvent(CONT)
[!ctx.syncLoss & (ctx.errCnt < errB)]
/if (ctx.goodBlk) {
ctx.sendByte(ACK);
if (ctx.anotherFile) ctx.sendByte(ctx.NCGbyte);
}
else ctx.sendByte(NAK);
if (ctx.goodBlk1st)
ctx.writeChunk();
onEvent(SER)
[c==SOH]
/ctx.getRestBlk();
if (ctx.goodBlk1st) {
ctx.errCnt=0;
ctx.anotherFile=0;
}
else ctx.errCnt++;
onEvent(SER)
[c == CAN]
/if (ctx.transferringFileD != -1)
ctx.closeTransferredFile();
ctx.result="SndCancelled";
onEvent(CONT)
[!ctx.anotherFile]
/ctx.sendByte(ACK);
ctx.tm(TM_END);
onEvent(CONT)
[ctx.anotherFile]
/ctx.openFileForTransfer();
Timeout
[Entry]
[Exit]
CondTransientOpen
[Entry]
POST("*",CONT);
[Exit]
CondTransientData
[Entry]
POST("*",CONT);
[Exit]
onEvent(CONT)
[ctx.syncLoss || ctx.errCnt >= errB]
/ctx.cans();
ctx.closeTransferredFile();
if (ctx.syncLoss)
ctx.result="LossOfSyncronization";
else
ctx.result="ExcessiveErrors";
onEvent(CONT)
[ctx.transferringFileD == -1]
/ctx.cans();
ctx.result="CreatError";
onEvent(TM)
/ctx.result="Done";
onEvent(CONT)
[ctx.syncLoss || ctx.errCnt >= errB]
/ctx.cans();
if (ctx.syncLoss)
ctx.result="LossOfSync at Stat Blk";
else
ctx.result="ExcessiveErrors at Stat";
onEvent(SER)
[c==EOT & !ctx.closeProb &
ctx.errCnt >= errB]
/ctx.cans();
ctx.result="ExcessiveEOTs";
onEvent(SER)
/COUT << "Receiver received totally unexpected char #" << c << ": " << (char) c << endl;
exit(EXIT_FAILURE);