MODULE *CommCell*

https://*elixir.bootlin.com*/linux/*v*4.2/source/drivers/tty/serial/*serial_core.c*
http://*www.modbus.org*/docs/*Modbus_over_serial_line_V*1.*pdf*

```
--algorithm RxTx
{
    tx is there is something to send, rx is there is something to receive
  variables
      isIdle = TRUE,
      startMsg = ":",
      end1 = "CR",
      end2 = "LF",
      tx = FALSE,   set by application if the buffer contains a message to send
      rx = FALSE,   set by code below when we've received a full, valid frame ready to be processed
      txReg ∈ 0 .. 255,
      rxReg ∈ 0 .. 255,
      rxBuffCount = 0;
      txBuffCount = 0,
      isValidCRC = TRUE;
      isValidModbus = TRUE;

    Station is idle
    can rx or tx
    tx ------------------------------

  process ( Serial = "Serial" )
  {
      idle: while ( TRUE )
      {
          isIdle := TRUE;
           if we have something to send and we are not receiving
           and if the buffer contains a valid Modbus frame (TODO)
          switchToTx: if ( tx = TRUE && txBuffCount \ = 0 && isValidModbus )
          {
               we are no longer idle, put the start message
               character, ";", into the send transmit register
              emission_start: isIdle  := FALSE;
                              txReg   := startMsg;
               while there is stuff to send, keep sending it byte by byte
              emission:   while ( l1 : txBuffCount \ = 0 )
                          {
                              l2: txReg := txBuff ;
                                  txBuffCount := txBuffCount − 1;
                          } ;
               when the register is empty send the CR/LF and go back to idle
              emission_end: txReg := end1;
```

```
            backToIdle: txReg := end2;
                        tx := FALSE ;
                        goto idle;
        } ;
        if we are receiving something (there is a ":" in the rx register)
        switchToRx: if ( rxReg = startMsg )
        {
            init reception: set idle to false, reset the receive buffer counter
            reception:  isIdle := FALSE;
                        rxBuffCount := 0;
                        if at any point before the message is complete we get another ":"
                        restart the process
                        r1: if ( rxReg = startMsg ) { goto reception } ;  can receive ":" multiple times
                        loop until we get the end of the message
                        r2: while ( rxReg \ = "CR" )
                            {
                                if ( rxReg = startMsg ) { goto reception; } ;
                                r3:  rxBuffCount := rxBuffCount + 1;
                                     rxBuff[rxBuffCount] := rxReg;
                            } ;
                        handle end of frame, check if valid modbus (TODO)
                        handle CRC (TODO)
            endFrame:   if ( rxReg = startMsg ) { goto reception; } ;
                        r4:  if ( rxReg = end2 && isValidModbus && isValidCRC )
                             {
                              if we get LF\CR and the CRC is good and the frame is valid modbus
                              then signal to the application that the rx buffer is ready to process
                                rx := TRUE;
                             } ;

        } ;
    } ;
}


}
```

BEGIN TRANSLATION

VARIABLES $isIdle$, $startMsg$, $end1$, $end2$, tx, rx, $txReg$, $rxReg$, $rxBuffCount$, $txBuffCount$, $isValidCRC$, $isValidModbus$, pc

$vars \triangleq \langle isIdle, startMsg, end1, end2, tx, rx, txReg, rxReg, rxBuffCount, txBuffCount, isValidCRC, isValidModbus, pc \rangle$

$ProcSet \triangleq \{\text{"Serial"}\}$

$Init \triangleq$ Global variables
$\land\ isIdle = \text{TRUE}$
$\land\ startMsg = \text{“:”}$
$\land\ end1 = \text{“CR”}$
$\land\ end2 = \text{“LF”}$
$\land\ tx = \text{FALSE}$
$\land\ rx = \text{FALSE}$
$\land\ txReg \in 0 \,..\, 255$
$\land\ rxReg \in 0 \,..\, 255$
$\land\ rxBuffCount = 0$
$\land\ txBuffCount = 0$
$\land\ isValidCRC = \text{TRUE}$
$\land\ isValidModbus = \text{TRUE}$
$\land\ pc = [self \in ProcSet \mapsto \text{“idle”}]$

$idle \triangleq$ $\land\ pc[\text{“Serial”}] = \text{“idle”}$
$\land\ isIdle' = \text{TRUE}$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“switchToTx”}]$
$\land\ \text{UNCHANGED } \langle startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,\ rxReg,$
$rxBuffCount,\ txBuffCount,\ isValidCRC,\ isValidModbus\rangle$

$switchToTx \triangleq$ $\land\ pc[\text{“Serial”}] = \text{“switchToTx”}$
$\land\ \text{IF } tx = \text{TRUE} \,\&\&\, txBuffCount \setminus = 0 \,\&\&\, isValidModbus$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“emission\_start”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“switchToRx”}]$
$\land\ \text{UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,$
$rxReg,\ rxBuffCount,\ txBuffCount,\ isValidCRC,$
$isValidModbus\rangle$

$emission\_start \triangleq$ $\land\ pc[\text{“Serial”}] = \text{“emission\_start”}$
$\land\ isIdle' = \text{FALSE}$
$\land\ txReg' = startMsg$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“emission”}]$
$\land\ \text{UNCHANGED } \langle startMsg,\ end1,\ end2,\ tx,\ rx,\ rxReg,$
$rxBuffCount,\ txBuffCount,\ isValidCRC,$
$isValidModbus\rangle$

$emission \triangleq$ $\land\ pc[\text{“Serial”}] = \text{“emission”}$
$\land\ \text{IF } l1 : txBuffCount \setminus = 0$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“l2”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“emission\_end”}]$
$\land\ \text{UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,\ rxReg,$
$rxBuffCount,\ txBuffCount,\ isValidCRC,$
$isValidModbus\rangle$

$l2 \triangleq$ $\land\ pc[\text{“Serial”}] = \text{“l2”}$

$\land\ txReg' = txBuff$
$\land\ txBuffCount' = txBuffCount - 1$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“emission”}]$
$\land \text{ UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ rxReg,\ rxBuffCount,\ isValidCRC,\ isValidModbus\rangle$

$emission\_end \triangleq \land\ pc[\text{“Serial”}] = \text{“emission\_end”}$
$\land\ txReg' = end1$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“backToIdle”}]$
$\land \text{ UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ rxReg,\ rxBuffCount,\ txBuffCount,\ isValidCRC,\ isValidModbus\rangle$

$backToIdle \triangleq \land\ pc[\text{“Serial”}] = \text{“backToIdle”}$
$\land\ txReg' = end2$
$\land\ tx' = \text{FALSE}$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“idle”}]$
$\land \text{ UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ rx,\ rxReg,\ rxBuffCount,\ txBuffCount,\ isValidCRC,\ isValidModbus\rangle$

$switchToRx \triangleq \land\ pc[\text{“Serial”}] = \text{“switchToRx”}$
$\land \text{ IF } rxReg = startMsg$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“reception”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“idle”}]$
$\land \text{ UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,\ rxReg,\ rxBuffCount,\ txBuffCount,\ isValidCRC,\ isValidModbus\rangle$

$reception \triangleq \land\ pc[\text{“Serial”}] = \text{“reception”}$
$\land\ isIdle' = \text{FALSE}$
$\land\ rxBuffCount' = 0$
$\land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“r1”}]$
$\land \text{ UNCHANGED } \langle startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,\ rxReg,\ txBuffCount,\ isValidCRC,\ isValidModbus\rangle$

$r1 \triangleq \land\ pc[\text{“Serial”}] = \text{“r1”}$
$\land \text{ IF } rxReg = startMsg$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“reception”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“r2”}]$
$\land \text{ UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,\ rxReg,\ rxBuffCount,\ txBuffCount,\ isValidCRC,\ isValidModbus\rangle$

$r2 \triangleq \land\ pc[\text{“Serial”}] = \text{“r2”}$
$\land \text{ IF } rxReg \setminus = \text{“CR”}$
$\text{THEN } \land \text{ IF } rxReg = startMsg$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“reception”}]$

$$
\begin{array}{l}
\qquad\qquad\qquad\quad \text{ELSE} \quad \land\, pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“r3”}] \\
\qquad\qquad\quad \text{ELSE} \quad \land\, pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“endFrame”}] \\
\qquad \land \text{UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,\ rxReg, \\
\qquad\qquad\qquad\quad rxBuffCount,\ txBuffCount,\ isValidCRC,\ isValidModbus\rangle
\end{array}
$$

$$
\begin{array}{lll}
r3 \triangleq & \land\, pc[\text{“Serial”}] & = \text{“r3”} \\
& \land\, rxBuffCount' = rxBuffCount + 1 & \\
& \land\, rxBuff' = [rxBuff \text{ EXCEPT } ![rxBuffCount'] = rxReg] & \\
& \land\, pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“r2”}] & \\
& \land \text{UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,\ rxReg, & \\
& \qquad\qquad\quad txBuffCount,\ isValidCRC,\ isValidModbus\rangle &
\end{array}
$$

$$
\begin{array}{lll}
endFrame \triangleq & \land\, pc[\text{“Serial”}] = \text{“endFrame”} & \\
& \land \text{IF } rxReg = startMsg & \\
& \qquad \text{THEN} \quad \land\, pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“reception”}] & \\
& \qquad \text{ELSE} \quad \land\, pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“r4”}] & \\
& \land \text{UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ rx,\ txReg,\ rxReg, & \\
& \qquad\qquad\quad rxBuffCount,\ txBuffCount,\ isValidCRC, & \\
& \qquad\qquad\quad isValidModbus\rangle &
\end{array}
$$

$$
\begin{array}{lll}
r4 \triangleq & \land\, pc[\text{“Serial”}] = \text{“r4”} & \\
& \land \text{IF } rxReg = end2 \,\&\&\, isValidModbus \,\&\&\, isValidCRC & \\
& \qquad \text{THEN} \quad \land\, rx' = \text{TRUE} & \\
& \qquad \text{ELSE} \quad \land\, \text{TRUE} & \\
& \qquad\qquad\quad \land\, rx' = rx & \\
& \land\, pc' = [pc \text{ EXCEPT } ![\text{“Serial”}] = \text{“idle”}] & \\
& \land \text{UNCHANGED } \langle isIdle,\ startMsg,\ end1,\ end2,\ tx,\ txReg,\ rxReg, & \\
& \qquad\qquad\quad rxBuffCount,\ txBuffCount,\ isValidCRC,\ isValidModbus\rangle &
\end{array}
$$

$$
\begin{array}{lll}
Serial \triangleq & idle \lor switchToTx & \lor\ emission\_start \lor emission \lor l2 \\
& \lor\ emission\_end \lor backToIdle \lor switchToRx \lor reception \lor r1 & \\
& \lor\ r2 \lor r3 \lor endFrame \lor r4 &
\end{array}
$$

$$Next \triangleq Serial$$

$$Spec \triangleq Init \land \Box[Next]_{vars}$$

END TRANSLATION