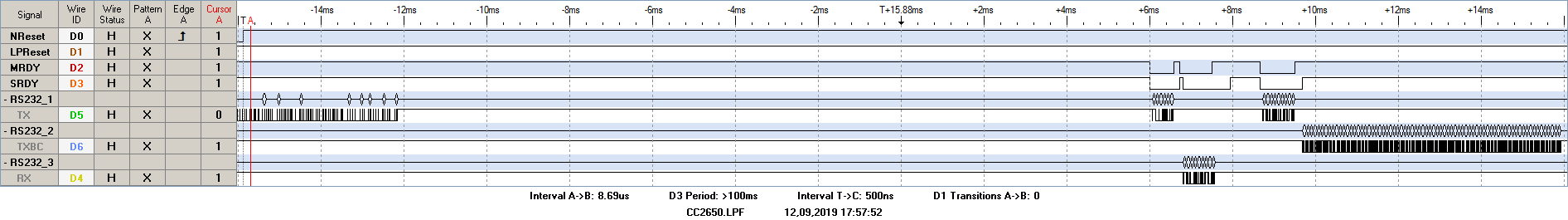
AP\_Init()

|  |
| --- |
| // Initialize serial link and GPIO to Bluetooth module  // see GPIO.c file for hardware connections  // reset the Bluetooth module and initialize connection  // Input: none  // Output: APOK on success, APFAIL on timeout  int AP\_Init(void){  int bwaiting;  int count = 0;  GPIO\_Init(); // MRDY, SRDY, reset  UART1\_Init();  fcserr = 0; // number of packets with FCS errors  TimeOutErr = 0; // debugging counts of no response error  NoSOFErr =0 ; // debugging counts of no SOF error  bwaiting = 1; // waiting for reset  while(bwaiting){  AP\_Reset();  count = 0; // should get SNP power up within 30 ms (duration is arbitrary and 'count' value is uncalibrated)  while(count < 600000){  if(AP\_RecvStatus()){  AP\_RecvMessage(RecvBuf,RECVSIZE);  if((RecvBuf[3]==0x55)&&(RecvBuf[4]==0x01)){  count = 600000;  bwaiting = 0; // success  }  }  count = count + 1;  }  } |
| //------------AP\_Reset------------  // reset the Bluetooth module  // with MRDY high, clear RESET low for 10 ms  // Input: none  // Output: none  void AP\_Reset(void){  ClearReset(); // RESET=0  SetMRDY(); // MRDY=1  Clock\_Delay1ms(10);  SetReset(); // RESET=1  } |
| //------------AP\_RecvStatus------------  // check to see if Bluetooth module wishes to send packet  // Inputs: none  // Outputs: 0 if no communication needed,  // nonzero for communication ready  uint32\_t AP\_RecvStatus(void){  return (ReadSRDY()==0);  } |
| /------------AP\_RecvMessage------------  // receive a message from the Bluetooth module  // 1) receive NPI package  // 2) Wait for entire message to be received  // Input: pointer to empty buffer into which data is returned  // maximum size (discard data beyond this limit)  // Output: APOK if ok, APFAIL on error (timeout or fcs error)  int AP\_RecvMessage(uint8\_t \*pt, uint32\_t max){  uint8\_t fcs; uint32\_t waitCount; uint8\_t data,cmd0,cmd1;  uint8\_t msb,lsb;  uint32\_t size,count,SOFcount=10;  // 1) wait for SRDY to be low  waitCount = 0;  while(ReadSRDY()){  waitCount++;  if(waitCount>APTIMEOUT){  TimeOutErr++; // no response error  return APFAIL; // timeout??  }  }  // 2) Make MRDY=0  ClearMRDY();  // 3) receive NPI package  // wait for SOF  do{  data = UART1\_InChar();  SOFcount--;  if(SOFcount==0){  SetMRDY(); // MRDY=1  NoSOFErr++; // no SOF error  return APFAIL;  }  }while(data != SOF);  \*pt = data; pt++;  fcs = 0;  // get size, once we get SOF, it is highly likely for the rest to come  lsb = UART1\_InChar(); \*pt = lsb; fcs=fcs^lsb; pt++;  msb = UART1\_InChar(); \*pt = msb; fcs=fcs^msb; pt++;  // get command  cmd0 = UART1\_InChar(); \*pt = cmd0; fcs=fcs^cmd0; pt++;  cmd1 = UART1\_InChar(); \*pt = cmd1; fcs=fcs^cmd1; pt++;  count = 5;  size = (msb<<8)+lsb;  // get payload  for(int i=0;i<size;i++){  data = UART1\_InChar();  fcs = fcs^data;  count++;  if(count <= max){  \*pt = data;  pt++; // payload  }  }  // get FCB  data = UART1\_InChar();  count++;  if(count <= max){  \*pt = data;  }  if(data != fcs){  fcserr++;  SetMRDY(); // MRDY=1  return APFAIL;  }  // 4) Make MRDY=1  SetMRDY(); // MRDY=1  // 5) wait for SRDY to be high  waitCount = 0;  while(ReadSRDY()==0){  waitCount++;  }  return APOK;  } |

nReset

Startzyklus