Lesbrief: Toelichting bij het LIN simulatie-programma

Met behulp van een Arduino Mega wordt een (gesimuleerde) LIN-boodschap uitgezonden. Om de spanning van 5V (Arduino) naar 12V (LIN) om te zetten is gebruik gemaakt van een LIN-UART JA1021 converter (ongeveer 6 euro).

Zowel de Arduino Mega als de LIN-converter zijn bij TinyTronics verkrijgbaar. Reden om uit te gaan van een Mega is dat deze controller over meerdere vrije seriele poorten beschikt. We komen dan niet in de problemen met het uploaden van het PC-programma naar de Arduino. Er wordt gebruik gemaakt van pin 18 (TX1).

Om de LIN-boodchap te bestuderen wordt gebruik gemaakt van een Pico-oscilloscoop die de mogelijkheid heeft om het seriele LIN-signaal te decoderen. Ook met behulp van een zeer goedkope datalogger en het Pulsview programma kan het LIN signaal worden gedecodeerd.

Een voorbeeld LIN-signaal van Pico is gebruikt voor de analyse en bestudering van het LIN signaal (reverse engineering). Het ging dan voornamelijk over de tijdintervallen.

Een complete LIN-boodschap is opgebouwd uit verschillende -te onderscheiden- gedeelten:

-Een break (een laag signaal van ongeveer 14 of 15 bytes) om het begin van een boodschap te markeren.
-Een synchronisatie byte (0x55 hex) nodig om de baudrate te herkennen (meestal 19200 bits/s).
-Een identifier om informatie te geven over de nog te komen data.
-de eigenlijke data (max. 8 bytes).
-een controle-byte (de checksum) nodig om transmissie-fouten te detecteren.

Standaard voor seriele communicatie (TxD en RxD) is het RS232 protocol waarmee ‘bytes’ na elkaar worden verstuurd. De TxD lijn is in rust hoog (+ 5V). Zo’n byte-boodschap begint door de de lijn even laag te maken (startbit) waarna de informatie-byte wordt verstuurd. Hierna wordt de lijn weer even hoog gemaakt (stopbit) en wacht dan op een volgende info-byte. Naar keuze kunnen we 1 of 2 stopbits toepassen. Zie fig. 1

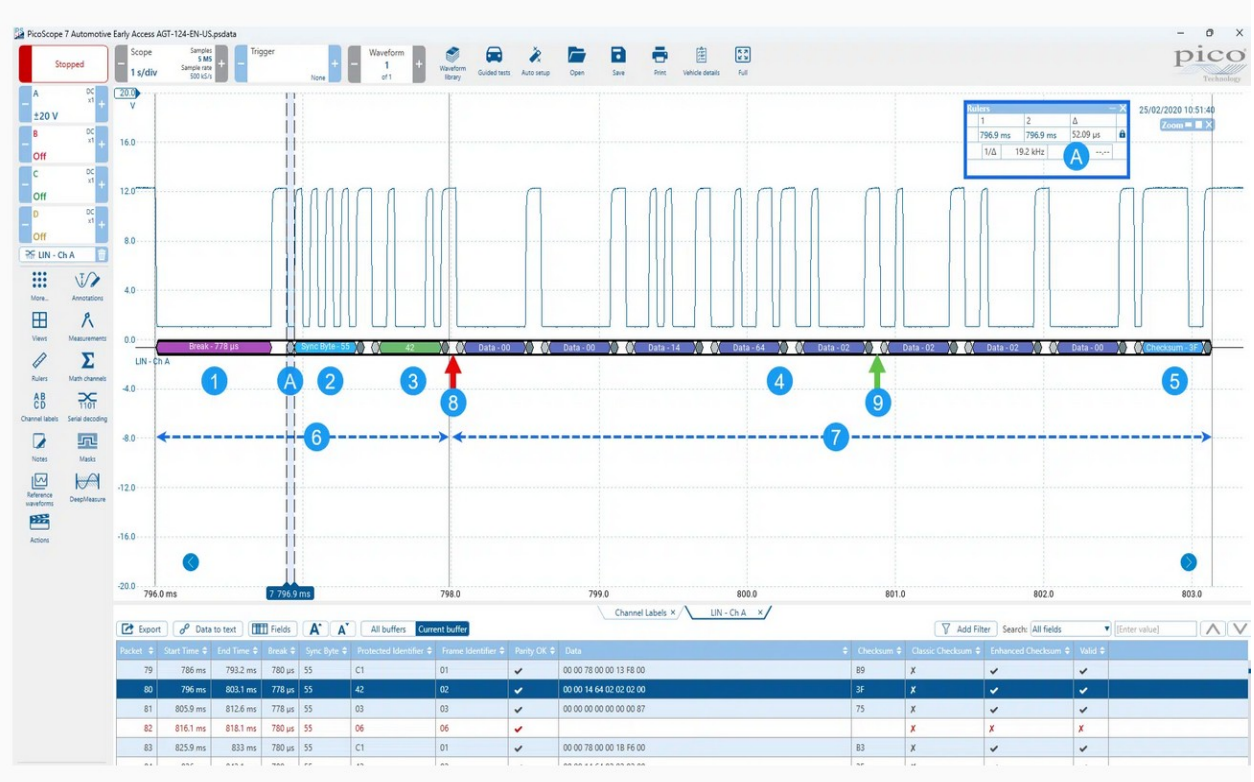

Fig. 1 Het pico-signaal dat is gebruikt voor de analyse van het LIN-bericht.

Bij de LINbus met een snelheid van 19,2 kbits/s duurt een bit dus 1000 000/19200 = 52,08 µs
Het eerste break-signaal duurt dan ongeveer zo'n 15 x 52 = 780 µs en daar ontstaat al het 1e programmeer-probleem. We kunnen bij deze hoge baudrate niet voldoende nullen uitsturen om de vereiste tijd te realiseren. Voor de oplossing kunnen we baudrate kortstondig verlagen of we kunnen de seriele-poort tijdelijk als OUT-poort instellen. Beide oplossingen zijn uitgeprobeerd en werken goed maar het leek ons fraaier om de baudrate te verlagen. Hier volgt het toegepaste programma met het nodige commentaar achter het // teken:

```
//Program name: SimulatieLINsignaal5.ino
//Arduino Mega TxD1, +LIN moduul TJA1021
//Dit programma werkt goed.
//Baudrate eerst lager om het breaksignaal te krijgen.
//Resultaat uit bestudering LINbus signaal PicoScope
//Signaal verder te bestuderen met PicoScope en Seriele decodering

#define analogInPin A0 // voor potmeter
#define TX1_PIN 18   // voor sleepmode /wake up

unsigned char data1;       // voor potmeterstand
unsigned char data2=0x3;   // data willekeurig gevuld
unsigned char data3=0x5;
unsigned char data4=0x06;
unsigned char data5=0x07;   //te versturen LIN-data
unsigned char data6=0x08;
unsigned char data7=0x09;
unsigned char data8=0x04;

int PotmeterValue=0;
unsigned char PotmeterStand=0;

void setup()
{
pinMode (2,OUTPUT);
digitalWrite(2,HIGH); // enable chip (wake-up)
}

void loop()
{
  PotmeterValue=analogRead(analogInPin); //lees potmeter in
  PotmeterStand=map(PotmeterValue,0,1023,0,255);
  data1=PotmeterStand;

  Serial1.end();
  Serial1.begin(9600, SERIAL_8N1);  // om de breaktijd voldoende lang te krijgen (1 stopbit)
  Serial1.write(0x00);
  Serial1.end();

  Serial1.begin(19200, SERIAL_8N2);  //Ppoort TX1 in de seriele mode, 2 stopbits
  Serial1.write (0x55);  //synchronisatie byte is altijd 0x55 (0b10101010)
  Serial1.write (0x42);  //protected identifier (willekeurig) bestaat uit 6 bits identifier + 2
pariteits bits

   // Nu het dataveld-array van (maximaal) 8 positieve bites
  unsigned char dataveld[8] = {data1, data2, data3, data4, data5, data6, data7, data8};
  Serial1.write (dataveld, sizeof(dataveld));

  //Veel gebruikte 'eenvoudige' berekening van de controle byte
  unsigned char checksum = 0x00; // defineer een variabele checksum voor de controle en zet
daar bijv. 0x00 in
```

```
  for (int i=0; i<8; i++)        // tel alle databytes bij elkaar op en breng deze terug naar 8 bits (1 byte)
    {
    checksum+=dataveld [i];
    }
  checksum = ~checksum;   // inverteer de byte

  Serial1.write (checksum);
}
```

Voeren we dit programma in en sluiten de picoscoop aan over de LIN-bus dan ontstaat het gesimuleerde signaal van fig. 2. Door de Pico-scope in te stellen op serieel decoderen zien we de LIN-boodschap inclusief de databytes weergegeven

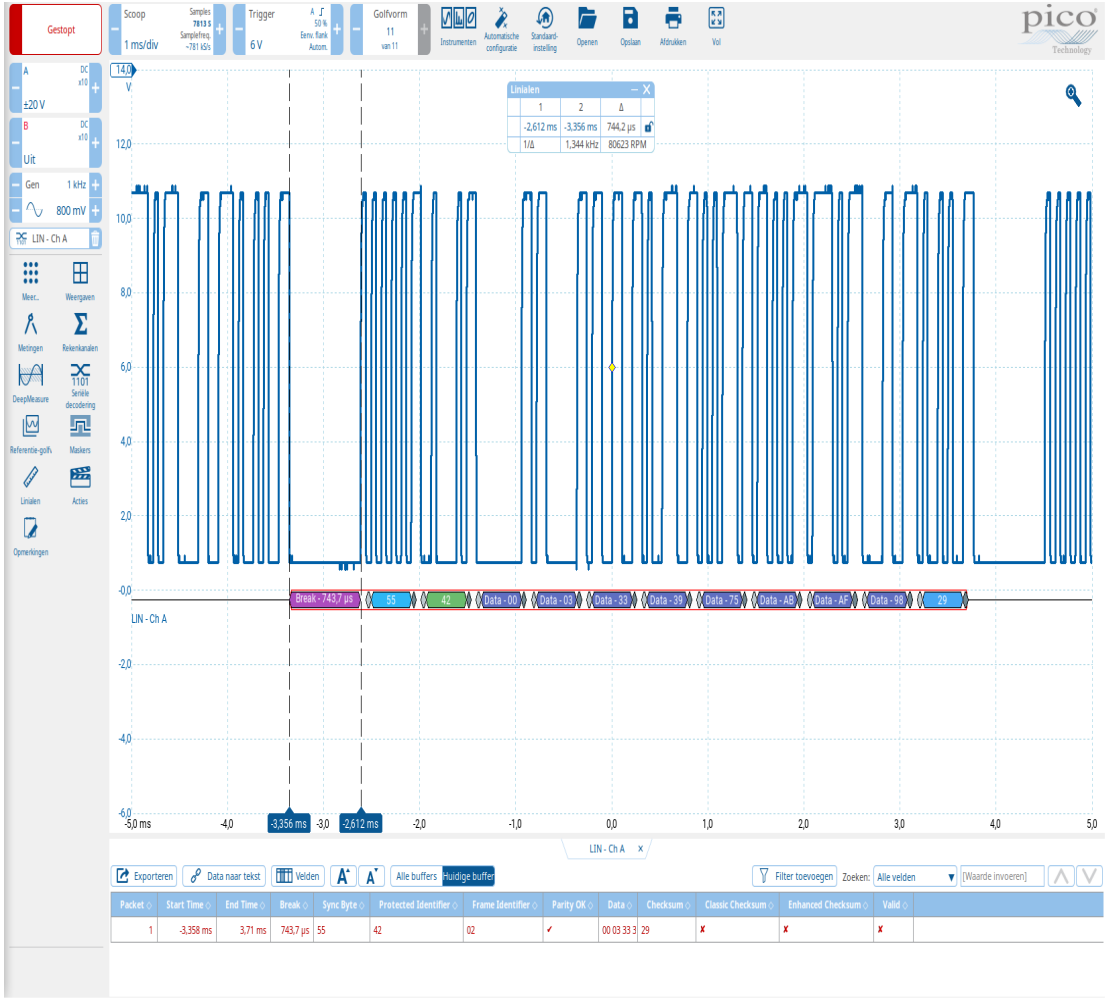

Fig. 2

In fig. 3 zien we de complele opstelling, de Arduino Mega met de TJS1021 moduul (die hier de 5V omzet naar 12V) en het Pico-scoopbeeld op de PC.

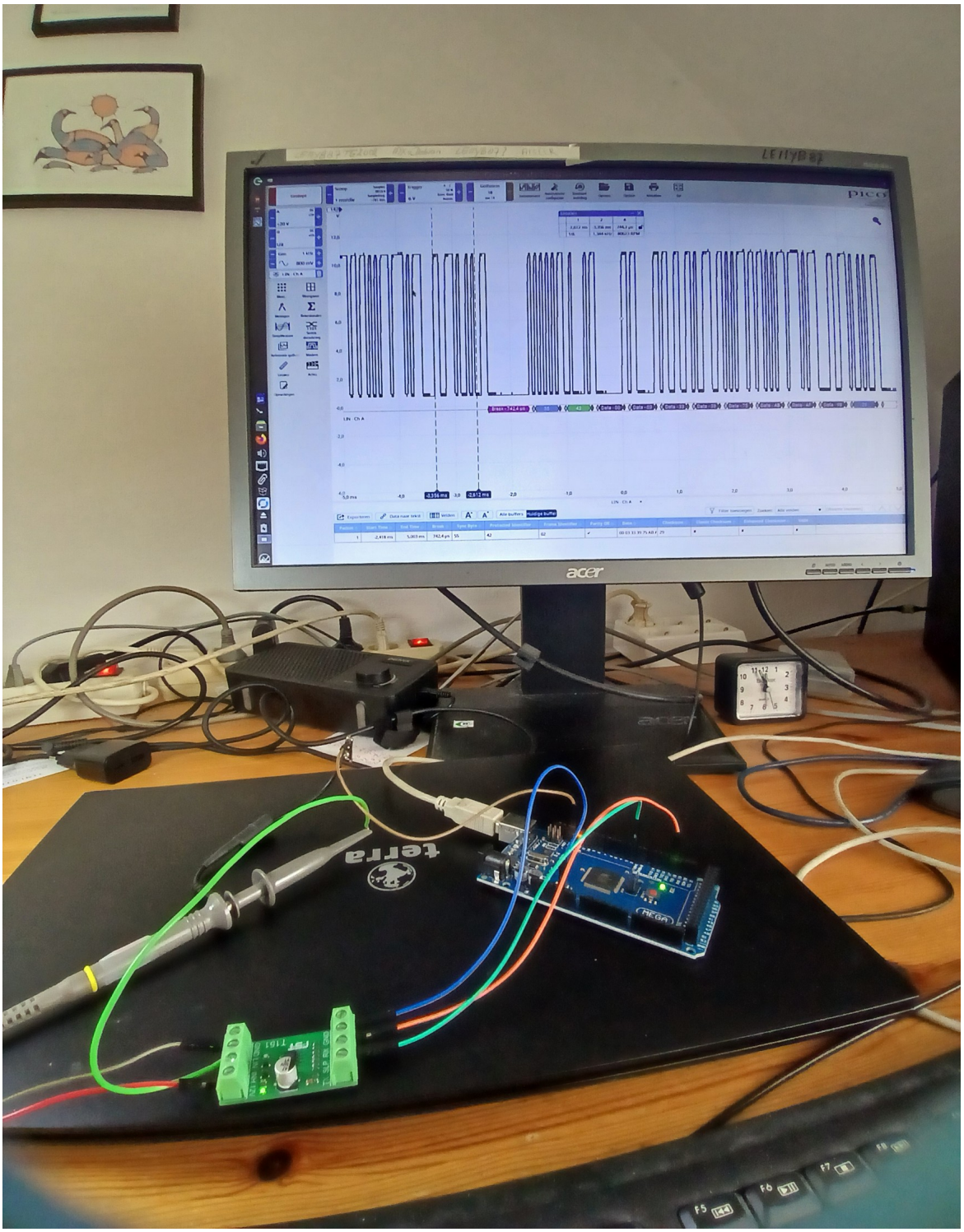

Fig. 3 Het gesimuleerde LIN-signaalweergegeven op de PC. De groene LIN-bus moduul wordt hiet alleen gebruikt voo de omzetting van 5 naar 12V.