Toelichting bij het LIN simulatie-programma (pag. 1 wordt nog vervolgd)

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.

Een voorbeeld LIN-signaal van Pico is gebruikt voor de bestudering (reverse engineering) van het signaal

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 met de de lijn even laag te maken (startbit) waarna de informatie byte wordt verstuurd. Hierna wordt de lijn weer even laag gemaakt (stopbit) en de lijn wordt weer hoog en wacht op een volgende info-byte.

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.