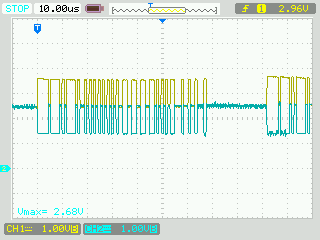
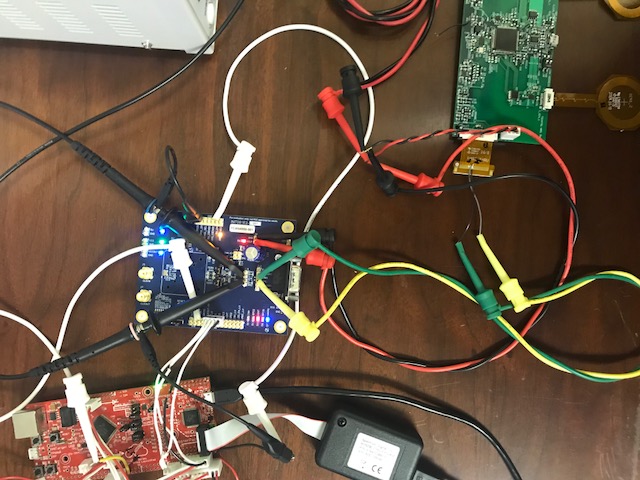
I’m setting up two TCAN4550 modules to send messages back and forth but I’m having a problem with the Transmit operation. I was expecting to see a single transmission. After the transmission, I was expecting to see the CAN Lines go Idle. What I’m seeing is an abbreviated message being transmitted every 91.6us (As shown in the picture below).

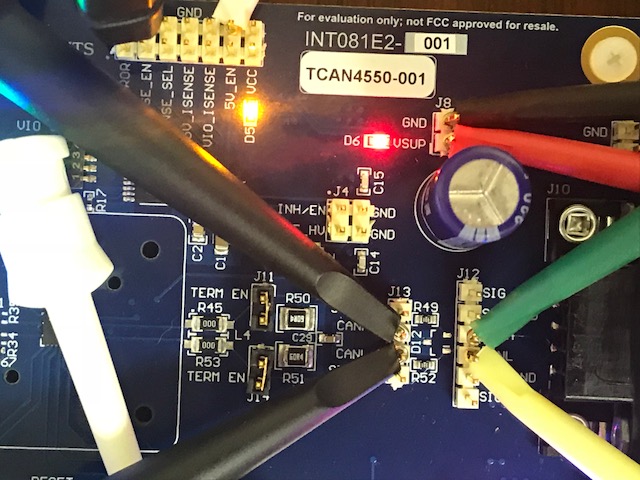
The only way I have been able to stop the transmissions is either Power cycling the transmitting device or by canceling the transmission by writing to the TX Buffer Cancellation Request.

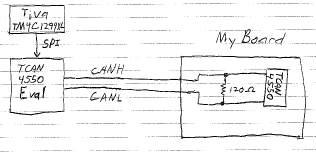


The message I’m trying to send is a 32 Byte message out of the Transmit Buffer. I’m using the STD 11 bit ID (Not the Extended). The Bit timing of the message looks to be correct at 1.11MHz.

Here is a picture of my Physical HW setup.







CODE USED TO SETUP THE TCAN4550

//-- RESET the chip

CAN\_RESET();

//--Test the CAN Transceiver

CAN\_Trans\_Commands(CAN\_PORT, CAN\_DEVICE\_ID);

/\*-- Run a test to ensure

\* device compatibility and communication

\*/

**if**(!dwError)

{

//-- Setup the Transceiver for Initialization --

CAN\_Write(dwCommPort,&TCAN4550\_W\_CCCR\_INIT\_ENABLE[0]); // 0x1018

//Control Register Write to being

//the initialization process and make registers

//write accessible.

CAN\_Write(dwCommPort,&TCAN4550\_W\_STANBY\_NOTCONFIG[0]); // 0x0800

//Standby Mode, Pin Config

//& MCAN\_CONFIG = 0

//-- Setup the Interrupts --

CAN\_Write(dwCommPort,&TCAN4550\_W\_nINT\_CLEAR[0]); // 0x0820

//Clear Interrupt/Diagnostic Flags

CAN\_Write(dwCommPort,&TCAN4550\_W\_nINT\_INTERRUPT\_ENABLE[0]); // 0x0830

//Enable interrupts for nINT

CAN\_Write(dwCommPort,&TCAN4550\_W\_INTERRUPT\_REGISTER\_SETTINGS[0]); // 0x1054

//Program the Interrupt Register

CAN\_Write(dwCommPort,&TCAN4550\_W\_ILS\_AND\_ILE[0]); // 0x1058 & 0x105C

//Assign Interrupts to M\_CAN\_INT1

//Enable EINT0 & EINT1

//-- Setup the BIT Timing --

CAN\_Write(dwCommPort,&TCAN4550\_W\_NOMINALBIT\_TIMING\_REG[0]); //0x101C

// 40MHz/4, Bit Sample settings

// Bit Time = 1.111MHz

//-- Setup the TXBuffer & FIFOs --

CAN\_Write(dwCommPort,&TCAN4550\_W\_TXBUFFER\_CONFIG[0]); // 0x10C0

// 5 Transmit Buffers

CAN\_Write(dwCommPort,&TCAN4550\_W\_TXEVENTFIFO\_CONFIG[0]); // 0x10F0

// 3 Transmit FIFOs

CAN\_Write(dwCommPort,&TCAN4550\_W\_TXELEMENTSIZE\_CONFIG[0]); // 0x10F0

//-- Setup the RX Buffers and FIFOS

CAN\_Write(dwCommPort,&TCAN4550\_W\_RXFIFO0\_CONFIG[0]); // 0x10A0

// 4 RX FIFO 0 Buffers

CAN\_Write(dwCommPort,&TCAN4550\_W\_RXFIFO1\_CONFIG[0]); // 0x10B0

// 4 RX FIFO 1 Buffers

CAN\_Write(dwCommPort,&TCAN4550\_W\_RXELEMENTSIZE\_CONFIG[0]); // 0x10BC

// RX FIFO & BUFFER Elements

//-- Setup the SID Filters --

CAN\_Write(dwCommPort,&TCAN4550\_W\_SID\_FILTER[0]); // 0x1084

// SID FILTERS

CAN\_Write(dwCommPort,&SID\_FILTER\_IDS[0]); // 0x8000

// SID IDS

//-- Put the Chip into Normal Mode

CAN\_Write(dwCommPort,&TCAN4550\_W\_CCCR\_NORMALMODE[0]); // 0x1018

// Control Register

// Clear the CCE and INT bits

CAN\_Write(dwCommPort,&TCAN4550\_W\_NORMAL[0]); // 0x0800

// Mode of Operation - Normal

}

}

CODE USED TO TRANSMIT A DUMMY MESSAGE.

//-- Read the Transmit FIFO Status Register

dwError = CAN\_Read(dwCommPort,&TCAN4550\_R\_DEVICEID[0]);

dwTXBuffMemory = 0x00;

dwError = CAN\_Read(dwCommPort,&TCAN4550\_R\_TXFIFO\_STATUS[0]);

//-- Determine if a Transmit Buffer is Available

dwNumAvailTXBuffers = CAN\_RX\_Buffer[3] & 0x1F;

**if**(dwNumAvailTXBuffers == 0)

{

dwError = 0x01; //No Buffer Available

}

**if**(!dwError)

{

//Get the Buffer Index

dwTXBufferIndex = CAN\_RX\_Buffer[2] & 0x1F;

dwTXBuffMemory = CAN\_TXBUFFERS[dwTXBufferIndex];

//Format the CAN\_TX\_Buffer for new message

CAN\_TX\_Buffer[1] = CAN\_WRITE\_COMMAND | (dwTXBuffMemory << 8) | CAN\_TXBUFF\_MSG\_CNT;

CAN\_TX\_Buffer[0] = 0x00000001; //One unique Address

//Format Header#1

CAN\_TX\_Buffer[2] = 0x00140000;

//Format Header#2

CAN\_TX\_Buffer[3] = 0x01850000;

//Move Message into the Local Transmit Buffer

CAN\_TX\_Buffer[4] = 0xCCCCAAAA;

CAN\_TX\_Buffer[5] = 0xCCCCAAAA;

CAN\_TX\_Buffer[6] = 0xCCCCAAAA;

CAN\_TX\_Buffer[7] = 0xCCCCAAAA;

CAN\_TX\_Buffer[8] = 0xCCCCAAAA;

CAN\_TX\_Buffer[9] = 0xCCCCAAAA;

CAN\_TX\_Buffer[10] = 0xCCCCAAAA;

CAN\_TX\_Buffer[11] = 0xCCCCAAAA;

//Move Local Transmit Buffer into the TCAN4550 Transmit Buffer

dwError = CAN\_Write(dwCommPort,&CAN\_TX\_Buffer[0]);

dwError = CAN\_Read(dwCommPort,&TCAN\_R\_TXBUFFER0[0]);

//-- Request that the Loaded TXBuffer Transmit the data

CAN\_TX\_Buffer[2] = (dwTXFromBuffer << dwTXBufferIndex);

CAN\_TX\_Buffer[0] = 0x00000001;

CAN\_TX\_Buffer[1] = CAN\_WRITE\_COMMAND | TCAN\_TXBAR\_REG;

dwError = CAN\_Write(dwCommPort,&CAN\_TX\_Buffer[0]);

//Determine if the Transmission is complete.

dwError = 1;

for(i=0;i<3;i++)

{

CAN\_delay(0x00000FF);

dwError = CAN\_Read(dwCommPort,&TCAN4550\_R\_TXBUFF\_PENDING[0]);

dwTXBufferIndex = CAN\_RX\_Buffer[3] & 0x1F;

if(!dwTXBufferIndex)

{

i = 3;

dwError = NO\_ERROR;

}

}

//Cancel if not complete.

**if**(dwError)

{

dwError = CAN\_Write(dwCommPort,&TCAN4550\_W\_TXBUFFER\_CANCEL[0]);

}

}

}