PIC32 CAN EID TX RX Code Example

Extended ID CAN Message Communication

****
* Note: the path "../h" must be added to the GCC compiler search path
* Jumpers JP1 and JP2 on the chipKit Pro MX7 board must be in the “CAN” position.
****

Equipment list:

1. chipKIT Pro MX7
2. CAN node connection cable to connect CAN1 to CAN2 on a single chipKIT Pro MX7.
3. MPLAB X version 2.10
4. XC32 version 1.30 or 1.31.
5. Digilent Analog Explorer

Notes : About the Code Example:

This code example demonstrates the use of the PIC32MX CAN Peripheral Library and PIC32MX CAN Module to send and receive Standard ID CAN messages. The general functionality is as follows:

CAN1 is setup for an SID to receive a message that will be sent by CAN2.
CAN2 is setup for an SID to receive a message that will be sent by CAN1.
LED1 and LEDA are toggled each time CAN1 is called upon to send a message.
LED2 and LEDB are toggled each time CAN2 is called upon to send a message.
LED3 and LEDC are set to the state sent in the CAN2 received message.
LED4 and LEDD are set to the state sent in the CAN1 received message.

A process in initiated where LED1 is toggled CAN1 sends out a message that contains the state of LED1. This message is intended to be received by CAN2. When CAN2 receives a message, it will set LED4 to the state contained in the message from CAN1. The CAN2 receive function calls the CAN2 transmit function. The CAN2 transmit function toggles LED2 and sends a message to CAN1 that contains the state of LED2. When CAN1 receives the message, LED3 is set to the state contained in the message.

CAN2 uses Channel 0 to transmit and Channel 1 to receive messages. The size of both the channels is set to 8 messages. Filter 0 on CAN2 is configured to accept Standard ID messages with ID 0x8004002.
The accepted message is stored in Channel 1.

If the code example runs successfully, it will be observed that LED1 through LED4 on the chipKIT Pro MX7 processor board will toggle together every one second as illustrated in the two following plots.

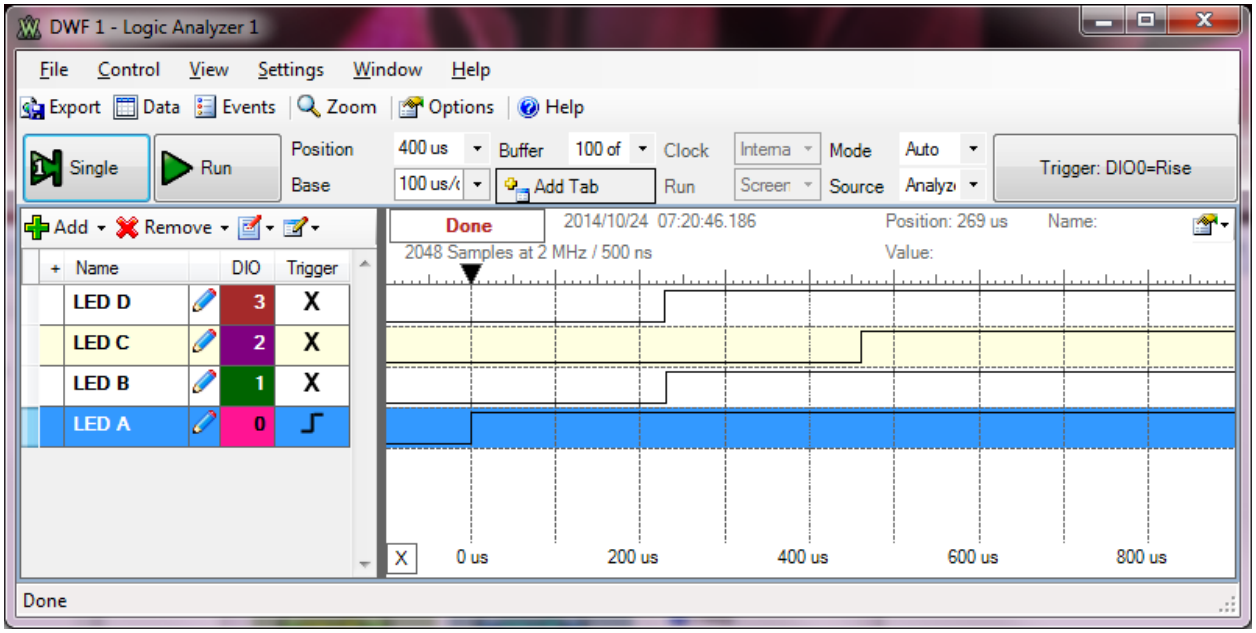

Figure 1. Toggling LEDs to the on state

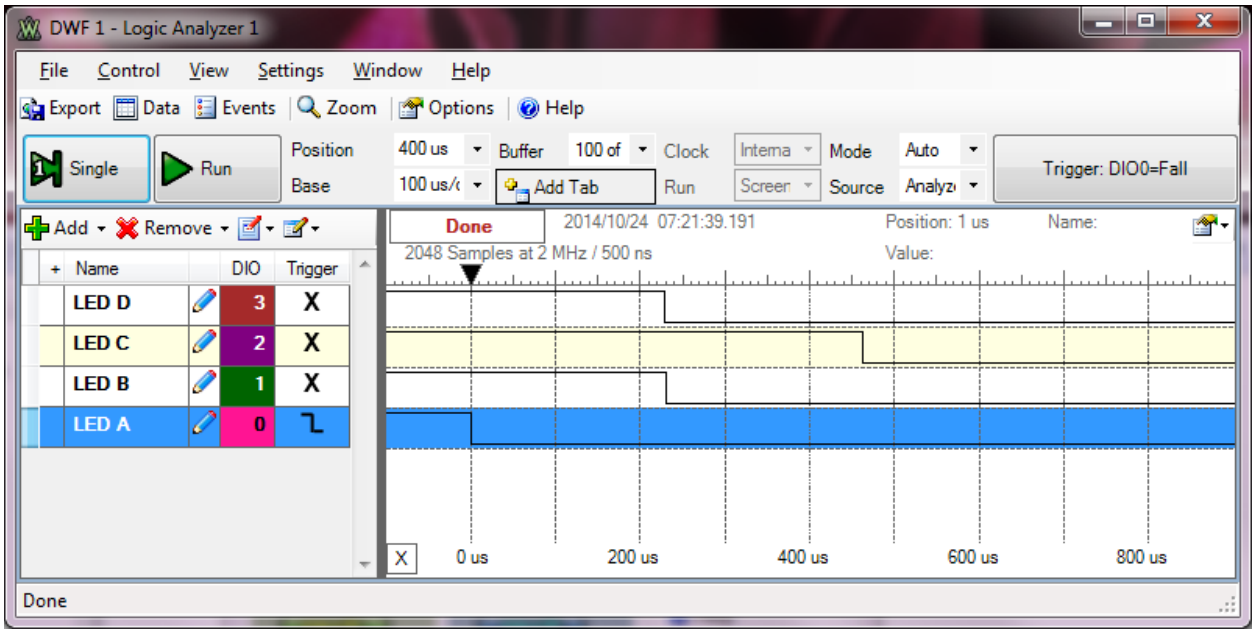

Figure 2. Toggling LEDs to the off state

Note: LEDA through LEDD correspond to LED1 through LED4. Based upon the description above, what are your observations and conclusions concerning the performance of CAN communications?