Reading the Tx ID from the MS and HS Decoders
Application Note AN-00156

Introduction

The Linx MS and HS Series decoders have the ability to output a number that uniquely identifies the encoder that sent the transmission. This can be used to log access attempts or in any application where the originating transmitter needs to be known. This application note will discuss how to read the ID number and make use of it. Sample software is available for download from the Linx web site and will be described here.

Decoder Operation

In order for the MS and HS decoders to accept a transmission, they must first learn the encoder's Code Word. When the decoders receive a transmission they will compare the received Code Word with one that is saved in memory. If a match is found, then the decoders will reproduce the encoder's data line states on its data lines.

The decoders can store 40 Code Words and will output the number in which the Code Word was learned as the Tx ID. So, for example, the first Code Word that is learned will have an ID of '1' that will be output as a binary 1 (0000 0001). The second Code Word will be '2' and its ID will be output as a binary 2 (0000 0010) and so forth.

Once a transmission is received and the decoder determines that it is valid, it will output the number in which it was learned. This will happen once after reception of the first valid packet. The ID number will be output as an eight-bit byte at the baud rate that is set by the SEL_BAUD lines.

MS Series Decoder TX_ID Packet Timing

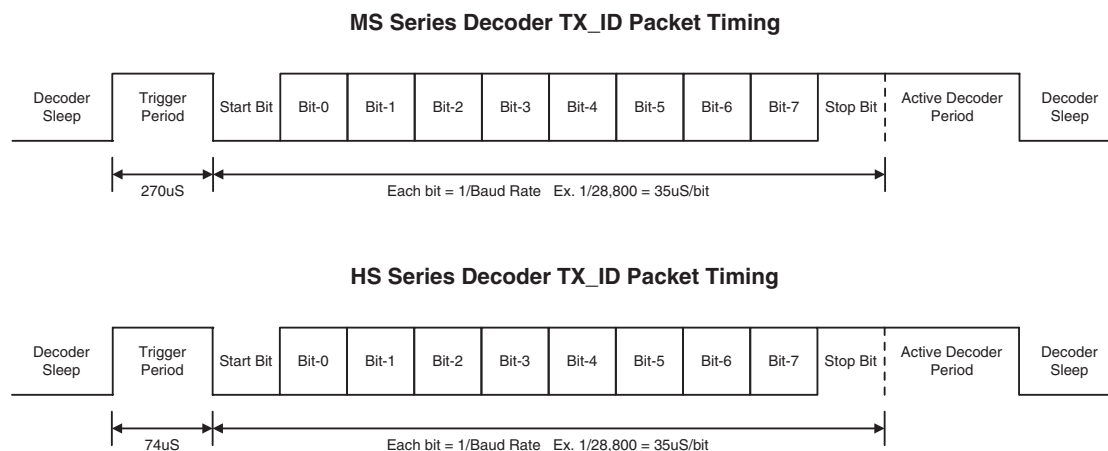**HS Series Decoder TX_ID Packet Timing**

Figure 1:TX_ID Packet Structure for the MS and HS Decoders

With this simple output, it is an easy task for a software routine to associate a number with a specific transmitter.

There are three common methods of reading the ID with an RS232 serial port, USB, or an embedded microcontroller. The following section will show examples of all three.

Examples

Send Over RS232

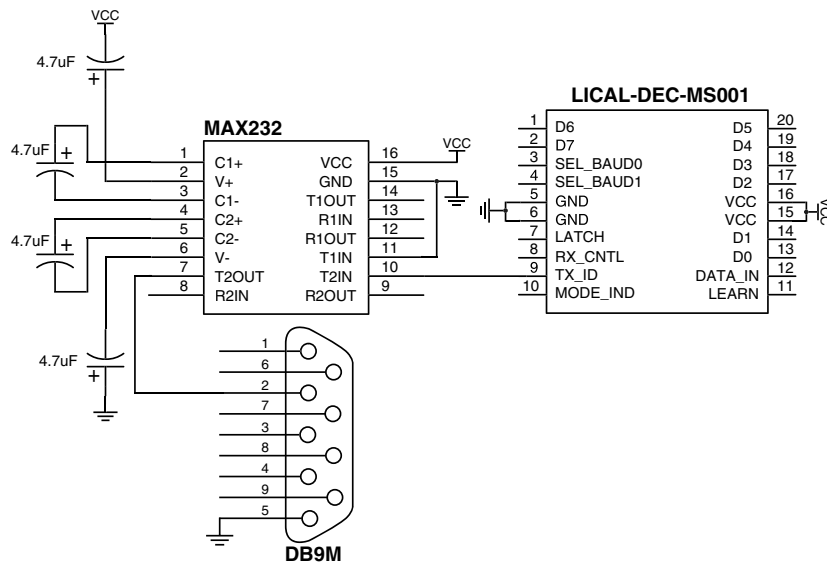Figure 2 shows the circuit schematic to send the Tx ID over RS232.

Figure 2: Schematic to Send Tx ID Over RS232

A Maxim Semiconductor MAX232 level converter chip is shown, but any RS232 level converter can be used. The TX_ID line on the decoder is connected to one of the RS232 driver inputs on the MAX232. The output is connected to a DB9 connector, which can then be connected to a PC.

Sample software to read the ID from the serial port can be downloaded from the Linx web site. This software is written in Visual Basic 6 and is well-commented so that it can be easily included into application software or ported into other formats.

Application Note AN-00156

Send Over USB
Figure 3 shows the circuit schematic to send the Tx ID over USB.
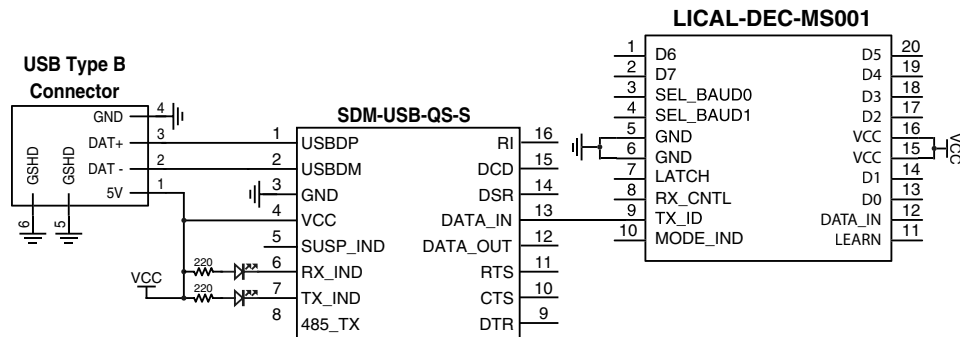

Figure 3: Schematic to Send Tx ID Over USB

The TX_ID line of the decoder is connected to the DATA_IN line of a Linx QS Series USB module, which is then connected to a USB type B jack. The QS module can then be used to communicate to a PC. This is useful since many current PCs do not have serial ports.

Sample software to read the ID from the PC can be downloaded from the Linx website. This software is written in Visual Basic 6 and is well-commented so that it can be easily included into application software or ported into other formats. It makes use of the Direct Drivers provided for the QS module. The Virtual COM Port drivers can be used with the software for use with a RS232 serial port that was described in the previous section. This offers the designer some flexibility when working on the system design. Please see the documentation for the QS module for details on the module and drivers.

Send to a Microcontroller
Figure 4 on the following page shows the circuit schematic to send the Tx ID to a microcontroller.

This example makes use of the PIC16F88 microcontroller from Microchip and the DMC-16202-LY LCD screen from Optrex. Both of these components are readily available from electronics distributors.
The PIC16F88 was chosen because it had more than enough I/O pins and a built-in UART. The UART is specifically made to handle serial data, making it perfect to read the ID from the decoder. The software to control it is also quite simple, just telling the UART to get data at a certain baud rate.

A UART is not required, though. The controller can be programmed to sample the state of a line and watch for the individual bits at the timing determined by the baud rate. This method works and could reduce costs by allowing a less expensive controller to be used, but the software gets much more complicated and will not be covered here.

DMC-16202-LY

D7 D6 D5 D4 D3 D2 D1 D0 EN R/W RS Vee Vcc GND
14 13 12 11 10 9 8 7 6 5 4 3 2 1

VCC

PIC16F88

1	RA2	RA1	1
2	RA3	RA0	2
3	RA4	RA7	3
4	RA5	RA6	4
5	Vss	V_{DD}	5
6	RB0	RB7	6
7	RB1	RB6	7
8	RB2	RB5	8
9	RB3	RB4	9

VCC

LICAL-DEC-MS001

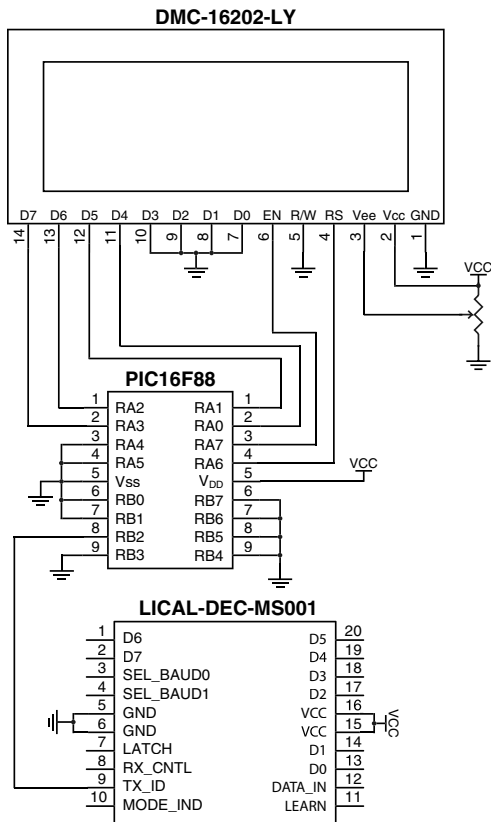|			
1	D6	D5	20
2	D7	D4	19
3	SEL_BAUD0	D3	18
4	SEL_BAUD1	D2	17
5	GND	VCC	16
6	GND	VCC	15
7	LATCH	D1	14
8	RX_CNTL	D0	13
9	TX_ID	DATA_IN	12
10	MODE_IND	LEARN	11

VCC

Figure 4: Schematic to Display the Tx ID to a LCD

Other microcontrollers with fewer pins, fewer features and a lower cost can be used with minor changes to the software. It is up to the designer to determine and make the changes.

The Optrex LCD screen has two lines of sixteen characters each. It takes data four bits at a time each time a control line is pulsed. Another control line tells the screen if the data is set-up information or for display. Other LCD screens can be used, but it is up to the designer to modify the software for operation with them.

The software in this example is written in C and was compiled using the PIC C compiler from CCS Info. The source code can be downloaded from the Linx Technologies web site and is well-commented, but some parts may need to be modified for different devices.

Application Note AN-00156

Linx