Application Note

Time Domain Reflectometry with DP83867 and DP83869

Melissa Chang

ABSTRACT

This application note outlines how Time Domain Reflectometry (TDR) helps solving various kind of cable fault challenges of Ethernet based communication systems. The application note describes the procedure to use TDR feature of the DP83867 and the DP83869 for implementing cable diagnostics feature in system.

Table of Contents

List of Figures

List of Tables

Trademarks

1 Time Domain Reflectometry

TDR only works for twisted pair connections. TDR involves sending a pulse on TX and RX pair and observing results on either pair. By measuring voltage amplitude, polarity, and the time interval, the PHY can determine the nature and position of the fault. The DP83867/DP83869 TDR generator sends pulse on the TX and RX channel, then monitors both channels to observe reflections. It sends a pulse one channel at a time, and if reflections are observed on the other channel, then the PHY TDR realizes that the wires have been crossed. The DP83867/DP83869 can detect one peak for each transmit and receive channel. TDR can be used for the following:

- Cable Open
- Cable Short
- TX/RX pair cross-wired
- Impedance discontinuity

TDR can be used only when the Link is down.

1.1 Example Connections

Following are the example connections where TDR can be used.

1.1.1 Open Circuit Cable

Open cable is easy to diagnose since it generates a strong reflection. No reflection is observed on the other channel. The reflection due to the open circuit is in-phase with the transmitted pulse (positive polarity). Any kind of inductive impedance discontinuity generates in-phase reflection and the amplitude depends on the amount of impedance discontinuity.

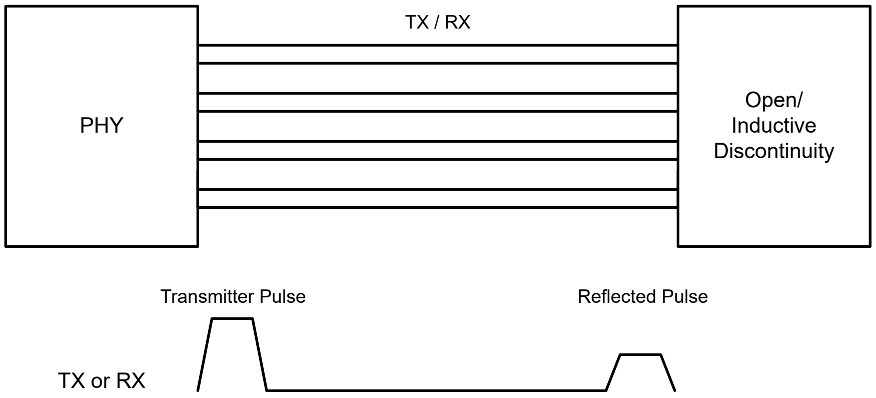

Figure 1-1. Open Circuit Cable

1.1.2 Short Circuit Cable

Short-circuited cables also generate a strong reflection, but this reflection is out-of-phase with the original pulse (negative polarity). Any kind of capacitive impedance discontinuity generates out-of-phase reflection. The amplitude depends on the amount of impedance discontinuity.

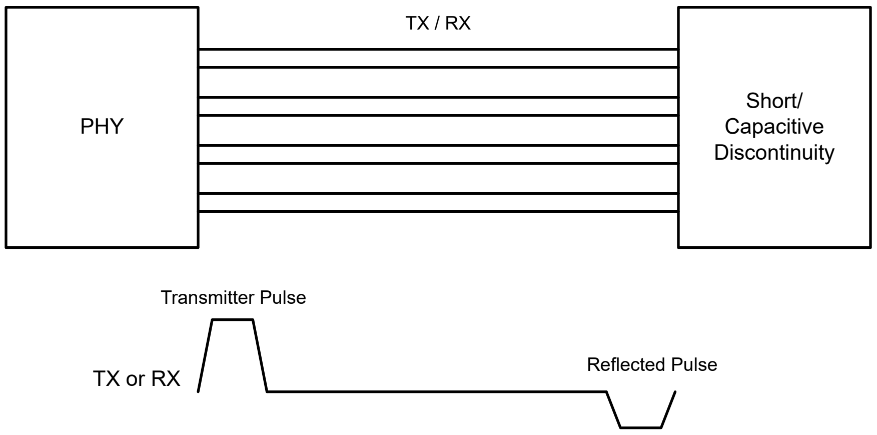

Figure 1-2. Short Circuit Cable

2 DP83867 and DP83869 TDR Implementation

For the purpose of this document, TDR implementation in DP83867 and DP83869 will be discussed.

2.1 TDR Configuration

To run TDR manually, link must be down *before* register configuration. To use the TDR feature, follow these steps:

1. Configure the registers shown in the Table 2-1. To see how to configure the extended registers, view the DP83867 or DP83869 data sheets.

Table 2-1. TDR Register Writes

Register Address	Value
0x0189	0x0000
0x0186	0x294A
0x0187	0x0A9B
0x001E	0x0003

2. After running the previous configuration, check that 0x001E[1] = 1 to confirm TDR is successfully completed. Then, read the following *extended* registers:

Table 2-2. TDR Register Results

Registers
0x0190
0x0191
0x0192
0x019a
0x019b
0x019c
0x01a5

3. See the next section, TDR Algorithm, for how to process the data collected from the registers to determine the cable fault type and location.

2.2 TDR Algorithm

This section describes how to process the TDR register data. All of the following steps must be carried out in code.

1. Define 5x3 matrix as shown in Table 2-3. Then, define each column as an array and name them peak_index, peak_value, and peak_sign.

Table 2-3. 5x3 Matrix Format

peak_index	peak_value	peak_sign
0190[7:0]	019a[7:0]	01a5[0]
0190[15:8]	019a[15:8]	01a5[1]
0191[7:0]	019b[7:0]	01a5[2]
0191[5:8]	019b[15:8]	01a5[3]
0192[7:0]	019c[7:0]	01a5[4]

2. Initialize the following variables:

Table 2-4. Initial Variables

Variable	Type	Initial Value	Description
i	int	5	Current row of the matrix
threshold	int	10	The threshold that peak_indx must exceed for a fault to be detected
threshold2	int	24 or 17	The threshold that peak_indx must exceed for a fault to be detected for the first two iterations. DP83867: threshold2 = 24 DP83869: threshold2 = 17
fault_detected	bool	FALSE	Defines whether a fault has been detected
fault_location	float	0	Fault location
prop_dly	float		Propogation delay variable, depends on the cable type and EVM. See the Prop_dly Values table for more information.
offset	int	16	Variable used in calculating the fault location
fault	int	1	Defines whether a fault is a SHORT (1) or an OPEN (0) fault

Table 2-5. Prop_dly Values

Cable type	Prop_dly (DP83867)	Prop_dly (DP83869)
Cat5/5e	5.35	5.1
Cat6	5	5
Cat7	4.6	4.6

3. Write a program with the following algorithm. See the MATLAB example code for reference.

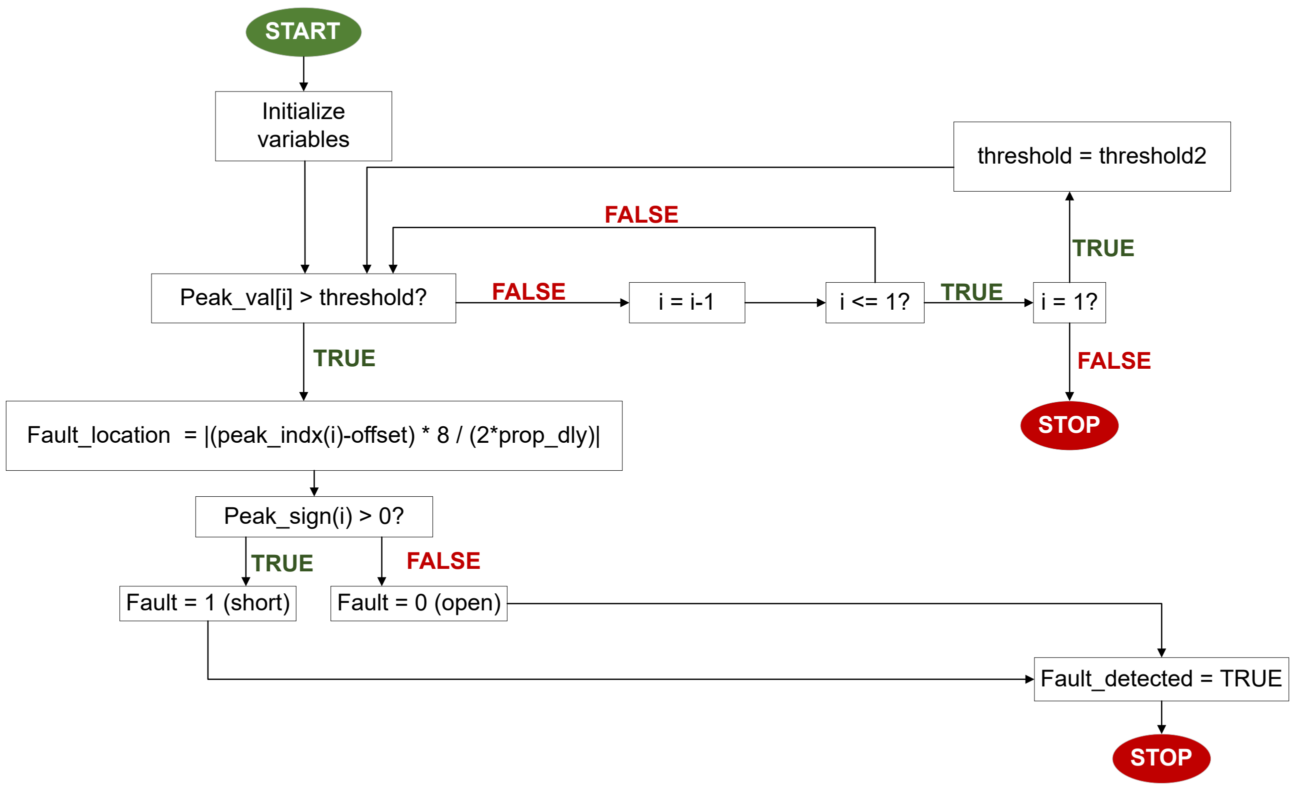

Figure 2-1. DP83867 and DP83869 TDR Algorithm

4. The fault type is in the *Fault* variable and the fault location is stored in fault_location.

2.2.1 TDR Algorithm Example Flow

Table 2-6 shows the TDR results from a 100m CAT5e cable.

Table 2-6. TDR Data Table Example

peak_index (0x180)	peak_value (0x185)	peak_sign (0x18A)
F	2D	0
93	11	0
0	0	0
0	0	0
0	0	0

The algorithm starts by checking to see if the last element the array *peak_val* is greater than the current threshold, 10. If it is not greater than 10, it moves onto the next row. If it is greater, the algorithm will look for the corresponding row in the array peak_indx and use that value to calculate the location of the fault. Then, the algorithm will look at the corresponding row of peak_sign to determine the fault type. A *0* means the fault is open, a nonzero value means that the fault is a short.

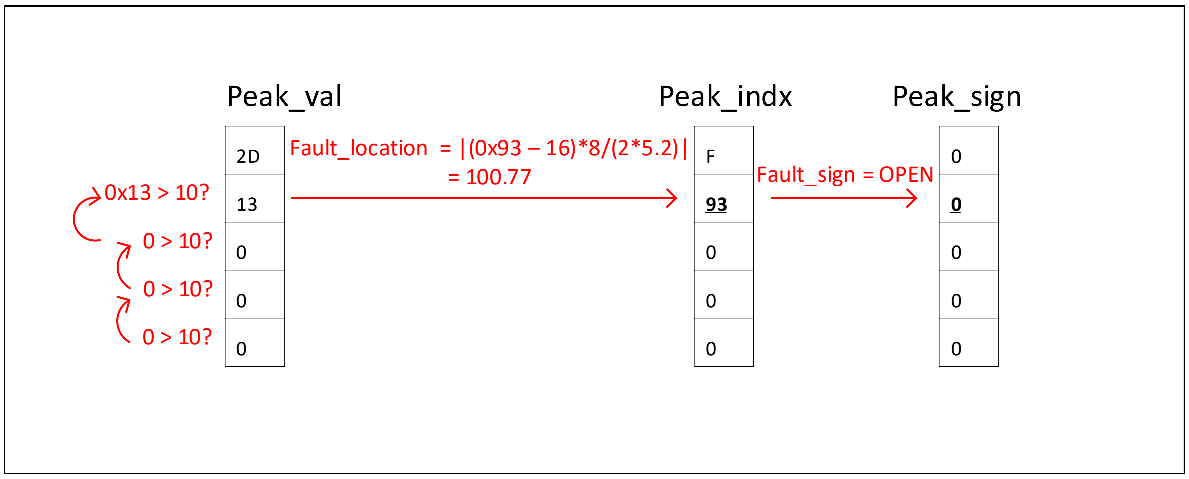

Figure 2-2. TDR Algorithm Example Flowchart

2.2.2 TDR Algorithm Matlab Example

The following code is an example of how to implement the TDR algorithm in Matlab. The input to the program is the 5x3 matrix.

```
function [tdr_results] = tdr_869(input_matrix)

tmp = input_matrix

%iteration = tmp(1:4:end);
peak_indx = tmp(1:3:end);
peak_indx %first column in 5x3 matrix
peak_val = tmp(2:3:end);
peak_val %second column
peak_sign = tmp(3:3:end);
peak_sign %third column

thr = 10;
prop_dly = 4.6; % ns perm %propogation delay of the cable type
offset = 16;

flt_found = 0;
flt_loc = 0;
flt_sign = 0;

%% Process the TDR data from Iteration 5 to Iteration 2
for jj = 1:4   %% 1,2,3,4,5 => 5,4,3,2,1
   jj=5-jj;
   jj+1
   peak_val(jj+1)
   if peak_val(jj+1) > thr
     flt_loc = abs(((peak_indx(jj+1) - offset)*8)/(2*prop_dly));
     if peak_sign(jj+1) > 0
       flt_sign = 1;
     else
       flt_sign = 0;
     end
     flt_found = 1;
     break;
   end
end

%% Process the TDR data for Iteration 1..
%% 1st for the offset seting of 0xC..
threshold2 = 17;

if flt_found == 0
    fprintf('Peak not found in higher iterations\n')
    peak_val(1)
    if peak_val(1) > thr_seg1_2
      fprintf('peak val : %d\n', peak_val(1));
      flt_loc = abs(((peak_indx(1) - offset)*8)/(2*prop_dly));
        if peak_sign(1) > 0
          flt_sign = 1;
        else
          flt_sign = 0;
        end
        flt_found = 1;
    end
end


%% Print the Results..
if flt_found == 1
  fprintf('\n');
  if flt_sign == 0
    fprintf('Fault location = %6.2f; Fault = Open\n',flt_loc);
  else
    fprintf('Fault location = %6.2f; Fault = Short\n',flt_loc);
  end
else
  fprintf('\n');
  fprintf('No Fault found\n');

end
tdr_results.flt_loc = flt_loc;
tdr_results.flt_sign = flt_sign;
```

```
return
end
```

3 Summary

This application note explains the basics of TDR and how to use the TDR functionality of the DP83867 and DP83869 Industrial Ethernet PHYs.

4 References

- Texas Instruments, *DP83867IR/CR Robust, High Immunity 10/100/1000 Ethernet Physical Layer Transceiver*, data sheet.
- Texas Instruments, *DP83869HM High Immunity 10/100/1000 Ethernet Physical Layer Transceiver With Copper and Fiber Interface*, data sheet.
- Texas Instruments, *How to use the TDR Feature of DP83822*, application note.
- Texas Instruments, *Solving Cable Faults Challenges with TI Ethernet PHYs*, application note.