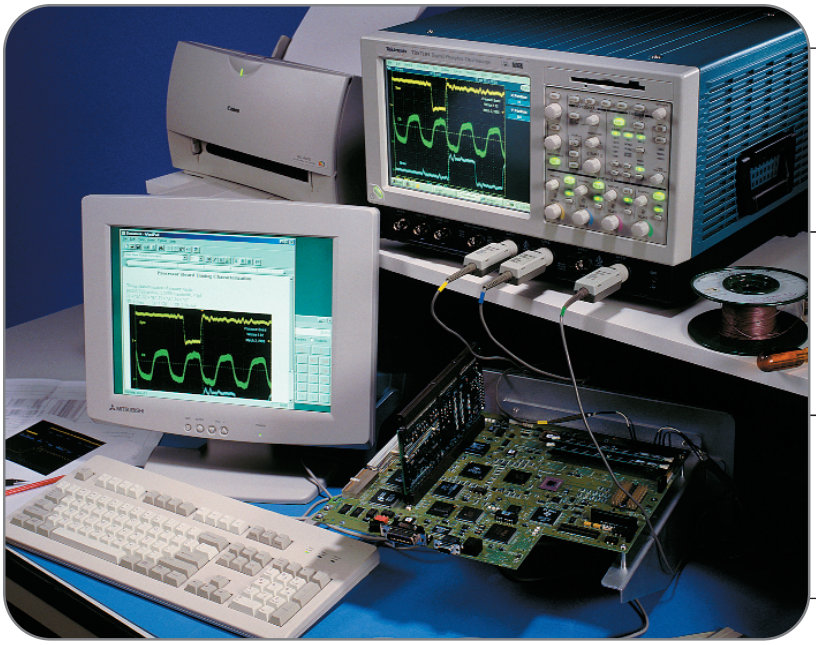
클럭 지터를 MATLAB 로 분석하기



COMPUTING
COMMUNICATIONS
VIDEO

▶ 간편한 데이터 상호통신능력을 갖춘 TDS7000 시리즈 제품

범위가 100ps 또는 100µs 이든 관계 없이, 지터의 특성화는 파형 데이터를 상세하게 분석해야 하기 때문에 많은 시간이 필요한 작업입니다. 이러한 분석을 자동화함으로써 수작업 보다 효율성과 테스트 반복성을 현저하게 향상 시키고, 정밀도를 더욱 증대 시킬 수 있습니다.

개방형 Windows® 를 사용하고 있는, TDS7000 시리즈 디지털 포스퍼 오실로스코프 (DPO) 는 Excel, Mathcad® 및 MATLAB® 과 같은 데이터 베이스 응용 프로그램을 계기 내에 상주 시킬 수 있을 뿐 아니라, 업계-표준 분석을 실행할 수 있는 타의 추종을 불허하는 우수한 장점을 갖춘 제품입니다.[1] 오실로스코프 획득 메모리의 파형 데이터를 송출한 뒤, 분석 애플리케이션으로 다시 가져오고, 동일 플랫폼상에서 모두 처리하고 디스플레이할 수 있습니다.

본 자료는 신호 데이터를 포착한 뒤 NRZ (영점 비 복귀) 클럭 신호 지터 분석을 간단하게 실행할 수 있는 TDS7000 시리즈 DPO 및 MATLAB 사용법에 관한 내용입니다. 다음과 같은 내용으로 구성되어 있습니다:

- ▶ 정밀 결과 도출에 필요한 샘플 해상도 결정 방법
- ▶ TDS7000 으로부터 MATLAB 로의 파형 데이터 이동 방법
- ▶ 간단한 지터 분석 워크시트 작성법
- ▶ 결과 그래프화에 필요한 MATLAB 사용법

[1] TDS7000 은 분석 시스템처럼 오실로스코프의 기능성을 극대화 할 수 있도록 또 다른 Windows 모니터로 출력할 수 있습니다. 두 번째 모니터는 별도의 애플리케이션을 동시에 하위 모니터 상에서 실행하면서 스코프를 효과적으로 사용할 수 있지만 반드시 필요한 것은 아닙니다.

Tektronix®

지터 개요

지터에 대해

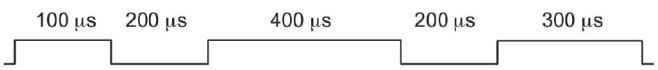
지터는 다음과 같은 2 가지의 중복된 뜻을 갖고 있습니다:

1. 시간 축 최적의 위치로부터의 신호 트랜지션 편차, 또는...
2. 트랜지션 간 타이밍 편차

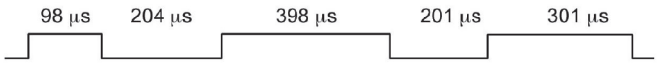
지터 시간간격은 대체로 수십 ps 에서 수백 ps 까지의 범위를 갖습니다. 클럭 주파수가 1GHz 범위에 도달한 경우, 외형적으로 적은 양의 지터 오류는 "타이밍 허용범위"의 핵심 부분이 될 수 있습니다. 즉, 일련의 논리 작동을 위해 시간이 할당됩니다. 예를 들어, 2.5Gbits/s 표준 SONET/SDH 비트 속도에서, 1 단위 시간간격 (1 데이터 비트) 은 400ps 일 뿐입니다. 송신기 및 수신기 컴포넌트가 이러한 범위의 대부분을 소모합니다. 지터는 너무 많은 잔여 시간 때문에 꼭 필요한 작동을 불가능하게 (또는 적어도 부정확하게) 만듭니다.

지터 특성

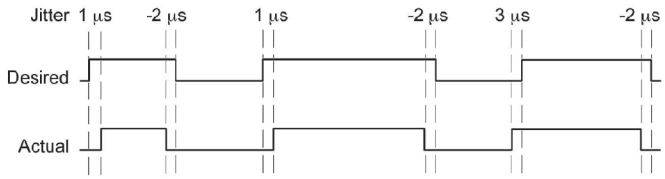
도해로 표시된 지터를 주의 깊게 보십시오. 다음과 같은 파형을 염두에 두십시오:



파형이 지정 임계값 위 + 일 때마다, 데이터는 논리 1 이 됩니다. 파형이 임계값 아래에 있으면, 데이터는 논리 0 이 됩니다. 이러한 파형을 송수신 할 때, 해당 타이밍이 아래 그림과 같이 된다고 가정하면:

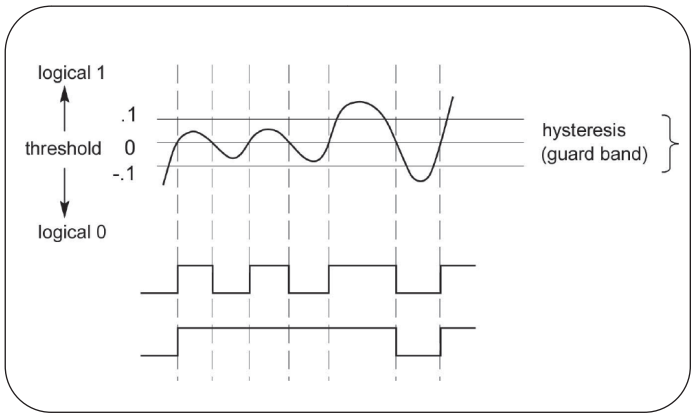


이 경우 신호 트랜지션은 시간 축 최적의 위치로부터 편차가 발생되고 그것은 트랜지션 사이에서 변화합니다 (지터의 정의). 잡음 및 기타 오류 소스와 같은 팩터로 인해 에지 위치가 변경되는 지점에서, 변경되는 크기는 지터의 절대량에 따라 µs, ns, 또는 ps 단위로 표시되는 지터 값입니다.



히스테리시스 효과 및 보호 대역

지터를 검출하고 평가할 때, 임계 교차 및 히스테리시스를 고려해야 합니다. "보호 대역"을 사용하는 것이 일반적인 방법입니다 (본질적으로 최적의 임계값을 넓히는 허용한계). 그림 1 은 교차하게 될 임계값 및 주변의 보호 대역을 나타냅니다. 논리 레벨에서 실제 변화와 같은 잡음이 표시되는 것을 피하기 위해, 파형이 먼저 보호 대역 밖으로 가지 않는다면 임계 교차는 유효 트랜지션을 고려하지 않아도 됩니다. 이러한 규칙이 강화됨으로써 참 에지 (그림 1 의 하단 추적)만의 검출이 보장됩니다.



▶ ***그림 1:*** *클럭 지터 분석시 임계값 및 히스테리시스*

지터 및 설계

고속 회로를 설계할 때 지터가 안정성에 영향을 준다는 것은 이미 모두가 다 알고 있는 사실입니다. 점차 이러한 인식은 표준, 사양 및 컴플라이언스 지침에 반영되고 있습니다.

그 실례로서 USB 업체 포럼에서 공표된 USB 2.0 사양이 있습니다. 본 자료는 여러 제조 업체, 즉 다른 표준의 USB-기반 제품들 간의 정보처리 상호 운용성을 보장할 수 있도록 하기 위한 것이고, 이는 향상되고 있는 플랫폼의 품질 및 결과 유지에 필수 불가결한 조건입니다. 계속 관심이 증대되고 있는 즉시 액세스 가능 멀티미디어 컨텐츠 분야의 성장과 같이 고객들의 USB 2.0 성능에 대한 필요성은 지속적으로 증대 될 것입니다.

USB 2.0 은 최고 480Mbits/s 의 데이터 속도와, 비디오 및 기타 대역폭-집중 컨텐츠를 제공하기에 충분한 성능을 갖추고 있습니다. 또한 속도는 지터 문제를 충분히 해결할 수 있을 정도로 고속입니다. 따라서 USB 2.0 인증에 필요한 Compliance Suite 에는 지터 사양이 포함되어 있습니다. USB 2.0 (계속 증가하고 있는 일반 소비자용 및 업무용 제품)을 포함하고 있는 모든 플랫폼 신제품은 인증 자격을 갖추기 위해 반드시 이러한 지침에 부합되어야 합니다.

USB 2.0 사양은 기술적 측면의 기능성 뿐 아니라, 몇 ps 타이밍 부정확도가 신제품으로서의 시장 점유 생존 가능성에 영향을 주는지 알려 주는 하나의 기준입니다.

알고리즘

다음과 같은 알고리즘을 사용합니다:

- 에지 타이밍 측정
- 에지 및 심볼 속도를 사용한 클럭 구동
- 측정된 평균 심볼 속도 결정
- 측정된 평균 심볼 속도의 오류 계산
- 구동 클럭 및 측정된 평균 심볼 속도를 사용한 에지 타이밍 재구성
- 측정된 에지 타이밍 및 재구성된 타이밍으로부터 지터 계산
- 지터 플롯화

용어

클럭 지터 예에서 사용되는 용어는 다음과 같습니다:

표 1: 클럭 지터 문제에서 사용하는 용어

용어	의미
심볼 속도	통신 시스템이 데이터를 전송하고 있을 때의 주파수. (RS232 에서의 보드속도)
샘플링 속도	오실로스코프가 데이터를 샘플링하고 있을 때의 주파수 (샘플/초 단위로 측정)
샘플링 시간간격	샘플간 시간 차이 (초/샘플 단위로 측정). 오실로스코프 사용자 인터페이스는 이것을 해상도라고 부릅니다. 수학적 *샘플링 시간간격 = 1/샘플링 속도.*
임계값	전압 값이 논리 영 (0) 또는 논리 일 (1)인지 결정하기 위해 사용하는 전압.
에지	파형이 임계값을 교차하는 지점. 교차될 때의 시간이 더욱 중요함. 에지는 항상 샘플 사이에서 발생하기 때문에 에지를 정확하게 찾기 위해 선형 보간법이라고 하는 방법을 사용해야 합니다.
히스테리시스	잡음을 줄이기 위해 알고리즘을 보다 덜 민감하게 하려고 사용하는 임계값 주위의 보호 대역. 각 에지에 대해 보호 대역 외부에 적어도 한 개의 지점이 있어야 합니다.

최적의 샘플링 시간간격 결정

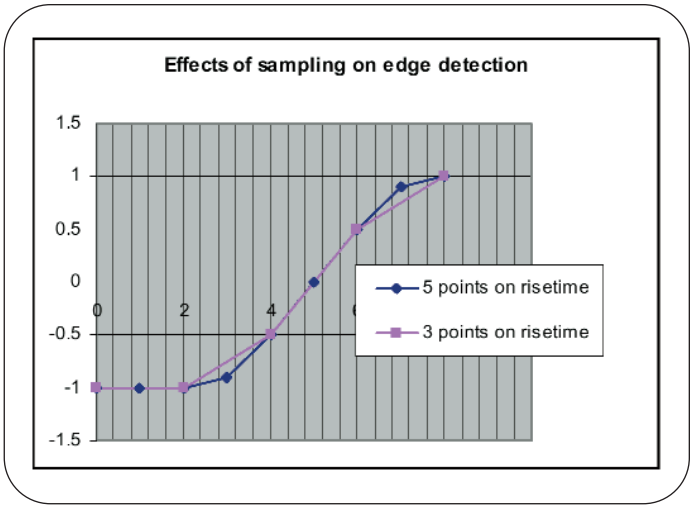
파형 데이터를 획득하기 전에, 사용하게 될 최적의 샘플링 시간간격을 결정해야 합니다. 지터를 측정하는 동안, 두 가지 사항을 고려해야 합니다:

- 가능한 한 많은 에지를 포착 (또는 원하는 에지)
- 가능한 한 각 에지의 위치를 정확하게 표시

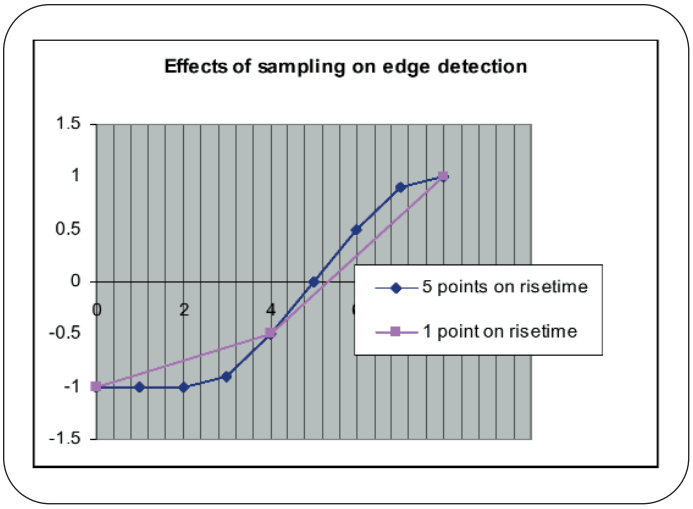
이것을 실행하기 위해 무리 없이 포착할 수 있는 에지 수를 제한하기에는 충분하지 않지만, 에지의 위치를 표시하기에 충분한 해상도를 갖는 데이터를 샘플링 해야 합니다. 여기에는 다음과 같은 두 개의 관련성이 있습니다:

- 데이터 처리 소요 시간과 레코드 길이와의 관련성
- 에지 발견 정밀도와 주어진 레코드 길이에서 포착할 수 있는 에지 수와의 관련성

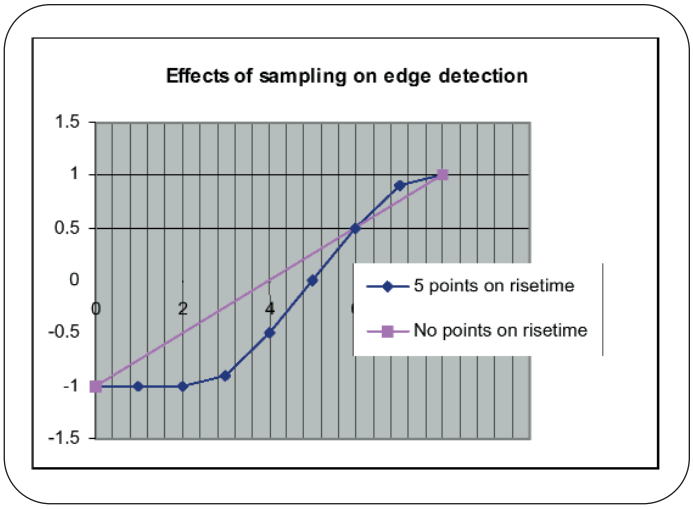
두 번째 관련성에 대한 내용은 아래와 같습니다. 다음 그래프는 상승 시간 3 개 지점 및 상승 시간 5 개 지점 모두에서 샘플링 된 에지를 나타냅니다. 두 가지 모두 수평 값 5 에서 0 을 통과합니다. 따라서 이것을 5 에서 발생한 에지라고 부릅니다.



다음 그래프는 상승 시간 단일 지점과 상승 시간 5 개 지점이 관련된 경우를 보여 주고 있습니다. 이 경우, 상승 시간에서 단일 지점을 갖는 선은 수평 값 5.3 에서 0 을 통과합니다. 데이터는 언더샘플링 되고 이러한 언더샘플링은 오류가 0.3 정도 나타나게 합니다. 결과는 지터처럼 보이지만, 이것은 단지 측정 오류일 뿐입니다.



마지막 그래프는 상승 시간중에 해당 지점을 갖지 않는 경우입니다. 이 경우, 상승 시간 중에 지점이 없는 선은 수평 값 4 에서 0 을 통과합니다. 데이터는 심하게 언더샘플링 되고 이 경우에는 오류가 -1 이나 됩니다. 다시 언급하지만, 결과는 지터처럼 보이지만 이것은 데이터가 언더샘플링 됨으로써 측정 중에 발생되는 오류일 뿐입니다.



위의 경우에서, 파형 상승 시간 중 3 개 및 5 개 샘플 간에서 제공되고, 에지를 정확하게 발견할 수 있도록 해주는 샘플링 시간간격 (수평 해상도)을 확인할 수 있습니다. 보다 높은 해상도는 선택한 레코드 길이에 관계 없이 에지 수를 반드시 제한합니다.

이러한 첫 번째 과정은 언더샘플링 됨으로써 발생한 오류가 신호에 존재하는 지터와 구별되지 않기 때문에, 파형 획득 및 차후 분석의 정확 한 결과를 도출하도록 결정 짓는 매우 중요한 과정입니다.

샘플링 시간간격 (획득용으로 설정된 샘플링 속도)을 결정함으로써, 일반적인 오실로스코프 방법론을 사용하여 파형을 포착합니다.

MATLAB 를 사용한 클럭 지터 솔루션

다음은 시작하기 전에 MATLAB 에 대해 몇 가지 확인해야 할 사항입니다:

- MATLAB 은 차원 구성이 필요 없는 어레이인 기본 데이터 성분의 대화형 시스템입니다. 따라서 C 언어와 같은 비-대화형 스칼라 언어로 프로그램을 작성해야 하는 경우보다 훨씬 적은 시간으로 여러 가지 기술적인 문제들을 해결할 수 있다는 것을 의미합니다.
- 대부분의 MATLAB Command Window 명령어 라인은 세미콜론 (;)으로 끝난다는 것을 항상 기억하십시오. 이는 MATLAB 이 특정 출력을 디스플레이하지 않고 즉시 계산을 실행할 수 있도록 하기 위한 것입니다.
- MATLAB 이 새로운 변수 명을 만나면, 변수를 자동으로 생성하고 해당 저장 용량을 할당합니다.

주: 본 자료는 MATLAB 자체의 상세 지침 역할을 할 수 없기 때문에 차후의 모든 설명은 주요 과정의 개요일 뿐, 세부적인 단계별 내용은 아닙니다.

파형을 MATLAB 에 적합한 파일로 송출

TDS7000 오실로스코프에 저장된 데이터를 송출하려면, 메뉴 바 모드를 선택한 뒤, File> Export Setup 를 선택하십시오. Export Setup 대화 상자가 나타납니다. Data destination 은 *MATLAB* 로, source 는 *channel1* 로 선택한 뒤, Include waveform scale factors 옆에 있는 상자에 표시하십시오. 2 개의 파일이 생성될 것입니다. 본 예에서 사용하기 위해 파일 이름은 *jitter5k.dat* 와 *jitter5k.hdr* 이라고 하겠습니다. 헤더 파일 (.hdr) 은 4 개의 필드로 구성되어 있습니다: 파형 레코드 길이, 샘플 시간간격, 트리거 위치 및 트리거 시간 오프셋.

Jitter1 함수 생성

다음은 클럭 지터 문제를 해결하기 위한 일부 하부-함수를 호출하는 함수를 생성하는 단계입니다. MATLAB 를 시작하고, Path Browser 를 선택한 뒤 *jitter5k.dat* 및 *jitter5k.hdr* 파일을 포함하고 있는 폴더 경로를 정의하십시오. 그리고 나서 Editor/Debugger 를 시작하고 다음과 같은 명령어를 입력하십시오:

```
function rmsJitter = jitter1(file,symbolRate,threshold,hysteresis)

waveform = dlmread(strcat(file,'.dat'));

header = dlmread(strcat(file,'.hdr'));

sampleInterval = header(2);
```

첫 번째 입력 인수 *(파일)* 는 귀하의 오실로스코프에서 송출한 파형 파일명입니다. 이 함수의 첫 번째 두 할당 문은 *파형* 및 헤더 어레이를 송출된 2 개의 파일로 해석합니다. 그리고 프로그램은 데이터를 수집하기 위해 사용한 (및 *헤더* 어레이의 두 번째 성분으로 가져온) 샘플 시간간격에 값을 할당합니다.

나중에 파형 파일 *(jitter5k)* 명을 갖는 jitter1 함수를 호출하고 심볼 속도, 임계값 및 히스테리시스 보호 대역에 필요한 값을 입력할 것입니다.

에지 타이밍 측정

우선 문제를 해결하기 위해 필요한 모든 함수를 정의해야 합니다. MATLAB 이 처리하는 방식 때문에 (모든 파일명은 포함하는 함수와 부합해야 함), 각 함수는 자체 파일명 하에 저장될 것입니다.

거친 입력 파형의 에지 타이밍을 측정하려면 별도의 ".m" 파일로 함수를 생성한 뒤, *jitter1* 함수에서 그 함수를 호출하면 됩니다.

measureEdgeTiming 함수 생성

여기서 4 개의 입력 인수들을 채택하고 1 개의 출력 인수 (시간)를 귀선 시키는 measureEdgeTiming 이라 불리는 함수를 정의할 것입니다. 이 함수를 만들려면, Editor/Debugger 의 File > New > M-file 명령어를 사용하여 새로운 파일을 시작합니다. 다음과 같은 함수 정의를 입력하십시오:

function time = measureEdgeTiming(waveform,threshold, hysteresis,sampleInterval)

이 알고리즘의 핵심은 우리가 에지를 찾기 전에 보호대역 (히스테리시스) 밖에 있는 것을 보장하는 것입니다. *thresholdTest* 변수가 이 작업을 수행합니다. *thresholdTest* 는 0 에서 4 까지의 "경우"에 따른 범위를 갖는 상태 변수입니다.

샘플링 된 전체 파형을 통해 증대되는 FOR 루프를 시작해서 보호 대역 위 또는 아래에 있는 최초의 전압 값을 발견해 냄으로써 시작하게 됩니다.

초기 상태를 처리하는 Case 0:

- 샘플 값이 임계값 + 히스테리시스 값보다 큰 경우, 네거티브 교차를 찾기 위해 *thresholdTest* 를 2 로 설정합니다.
- 샘플 값이 임계값 (히스테리시스 값 보다 적을 경우, 포지티브 교차를 찾기 위해 *thresholdTest* 를 1 로 설정합니다.
- 2 가지 조건에 부합되지 않는 경우, *thresholdTest* 를 0 으로 유지하고 보호 대역 외부에서의 샘플을 계속 찾습니다.

Case 1 은 포지티브 에지 임계값과 같거나 큰 경우의 샘플을 테스트합니다. 조건에 부합하는 경우, 에지 위치지정을 위해 2 개의 인접 샘플 사이에 보간 합니다. 그러면 두 번째 샘플이 보호 대역 위에 있는지 확인하기 위해 테스트를 실행합니다:

- 있다면, 네거티브 교차를 찾기 위해 *thresholdTest* 를 2 로 설정해야 합니다.
- 없다면, 네거티브 교차를 찾기 전 보호 대역 위에서 샘플을 찾기 위해 *thresholdTest* 를 3 으로 설정해야 합니다.

Case 2 는 네거티브 에지 임계값과 같거나 적은 샘플을 찾습니다. 조건에 부합하는 경우, 에지 위치지정을 위해 2 개의 인접 샘플 사이에 보간 합니다. 그러면 두 번째 샘플이 보호 대역 아래에 있는지 확인하기 위해 테스트를 실행합니다 .

- 있다면, 포지티브 교차를 찾기 위해 *thresholdTest* 를 1 로 설정해야 합니다.
- 없다면, 포지티브 교차를 찾기전 보호 대역 아래에서 샘플을 찾기 위해 *thresholdTest* 를 4 로 설정해야 합니다.

Case 3 은 네거티브 에지를 찾기전 보호 대역 위에 재차 있는지 확인하기 위해 검사합니다. 이러한 조건은 포지티브 에지를 정의하는 두 번째 샘플이 보호 대역보다 적은 경우에 발생합니다.

Case 4 는 포지티브 에지를 찾기전 보호 대역 아래에 재차 있는지 확인하기 위해 검사합니다. 이러한 조건은 네거티브 에지를 정의하는 두 번째 샘플이 보호 대역보다 큰 경우에 발생합니다.

FOR 루프 실행이 완료되면, 모든 파형 에지가 발견되고 교차 시간으로 변환됩니다 (그림 2).

```
function time = measureEdgeTiming(waveform,threshold,hysteresis,sampleInterval)
% This function finds the edges in a waveform.
% An edge is defined as the point where the waveform crosses the threshold
% The function includes a provision to define a guard band which can be used
% to reduce the sensitivity to noise.  The guard band is defined by the hysteresis
% values.
% The function will ensure that the waveform moves outside the guard band between
% successive edges.
% The edges are located by time.
% The first sample is considered as time zero.

edgeNbr = 1;
thresholdTest = 0;
% threshold test is a state variable, its values have the following meanings
% (remember, it takes two samples to define an edge)
%    0 = looking for a value outside the guard band
%    1 = looking for a positive edge. a positive edge occurs when we find a sample
%        at or above the threshold
%    2 = looking for a negative edge. a negative edge occurs when we find a sample
%        at or below the threshold
%    3 = looking for a sample above the guard band before we look for a negative
%        edge. it takes two samples to define an edge.  in this case the second
%        sample that defined a positive edge was in the guard band.
%    4 = looking for a sample below the guard band before we look for a positive
%        edge. it takes two samples to define an edge.  in this case the second
%        sample that defined a negative edge was in the guard band.

for index = 1:length(waveform)-1; % iterate through the entire waveform
   switch thresholdTest;
      case 0 % looking for a value outside the guard band
         if waveform(index) > threshold + hysteresis;
            thresholdTest = 2;
         elseif waveform(index) < threshold - hysteresis;
            thresholdTest = 1;
         end

      case 1 % below threshold, looking for positive threshold crossing
         if waveform(index + 1) >= threshold;
            fractional = (threshold - waveform(index))/(waveform(index + 1) - waveform(index));
            time(edgeNbr) = (index - 1 + fractional)*sampleInterval;
            edgeNbr = edgeNbr + 1;
            if waveform(index + 1) > threshold + hysteresis;
               thresholdTest = 2;
            else
               thresholdTest = 3;
            end
         end

      case 2 % above threshold looking for negative crossing
         if waveform(index + 1) <= threshold;
            fractional = (threshold - waveform(index))/(waveform(index + 1) - waveform(index));
            time(edgeNbr) = (index - 1 + fractional)*sampleInterval;
            edgeNbr = edgeNbr + 1;
            if waveform(index + 1) < threshold - hysteresis;
               thresholdTest = 1;
            else
               thresholdTest = 4;
            end
         end

      case 3 % looking for a sample above the threshold + hysteresis
         if waveform(index) > threshold + hysteresis;
            thresholdTest = 2;
         end

      case 4 % looking for a sample below the threshold - hysteresis
         if waveform(index) < threshold - hysteresis;
            thresholdTest = 1;
```

▶ ***그림 2:*** *measureEdgeTiming 함수*

measureEdgeTiming 함수 호출

이제 함수에 호출한 것을 위치시키면 됩니다. Editor/Debugger 를 시작하고, *jitter1.m* 을 열어 파형의 에지를 발견하는 명령어를 만드는 다음과 같은 문을 추가하십시오:

```
measuredTime = measureEdgeTiming(waveform,threshold,
hysteresis, sampleInterval);
```

이 문은 4 개의 인수를 갖는 measureEdgeTiming 함수를 호출하고 귀선 결과를 measuredTime 어레이에 할당합니다.

에지간 클럭 구동

각 에지 쌍 사이에는 클럭 번호 n 이 항상 있습니다. 인접 에지 간의 측정된 에지 시간을 뺀 뒤, 그 값을 주어진 심볼 속도에 곱함으로써 에지 간 클럭 번호의 비-정수 값을 얻습니다. 그 값을 가장 가까운 정수로 반올림하면, 그때 클럭 어레이에 추가된 값을 얻을 수 있습니다. 이 어레이에 저장된 번호는 첫 번째 에지 이후의 전체 클럭 번호입니다. 번호를 계산하기 위해, *jitter1.m* 함수에 FOR 루프를 만드십시오.

측정된 평균 심볼 속도 계산

여기서 우리는 최적의 측정 에지 (measureTime 어레이) 및 직선의 구동 클럭 (클럭 어레이) 을 발견할 수 있습니다. 이것을 실행하기 위해, MATLAB 의 "polyfit" 함수를 사용한 선형 회귀를 사용할 것입니다. 직선에 필요한 공식은 다음과 같습니다:

$$y = a + bx$$

여기서:

a = 인터셉트 (선이 y 축을 가로지르는 지점)
b = 기울기 (선을 따라 변하는 속도)

MATLAB 의 "polyfit" 함수로 a 및 b 값을 생성할 수 있습니다. 이제 귀선 결과 *coef (1)* 및 *coef (2)*를 각각 호출할 것입니다. 클럭 번호를 알고 있는 경우 에지 구동 시간을 제공해 주는 공식이 필요합니다.
그래서 다음과 같은 것을 사용할 것입니다:

y = 에지 시간 (reconstructedTime 에 저장될 값)

x = 에지 클럭 번호 (클럭에서)

다음과 같은 라인을 jitter1.m 함수에 추가해서 측정된 평균 심볼 속도 [1] 을 계산하기 위해 기울기 공식을 사용하십시오:

```
coef = polyfit(clocks, measuredTime, 1);

a = coef(2);

b = coef(1);

measuredAverageSymbolRate = 1/b;
```

[1] 수치 해법 (Numerical Recipes) 참조: 여기서 사용된 방정식의 완전한 내용은 캠브리지대 출판국 발행 논문 'The Art of Scientific Computing' 14.2 장을 참조 바람.

심볼 클럭 오류 속도 계산

측정된 평균 심볼 속도가 주어지면 심볼 클럭 오류 속도를 계산할 수 있습니다. 측정된 평균 심볼 속도에서 주어진 심볼 속도를 빼고 그 값을 심볼 속도로 나눔으로써 원하는 결과를 얻을 수 있습니다. *jitter1.m* 함수에 다음을 추가하십시오:

```
measuredSymbolRateError = (measuredAverageSymbolRate
–symbolRate)/symbolRate;
```

파형 그래프화

지금까지 계산된 값을 사용하여 MATLAB 는 디스플레이된 심볼 속도 오류를 갖는 입력 파형을 그래프화 할 수 있습니다. 그래프에서 보다 상세한 것을 볼 수 있도록 하기 위해 파형의 한 부분만 (5000 개 중 2000 개의 샘플)을 그래프화 할 것입니다. *jitter1.m* 함수에, measuredSymbolRateError 을 플롯화 하는 명령어를 추가하십시오. MATLAB 플롯 함수는 자동으로 새로운 Figure Window 를 열거나 기존의 것을 사용합니다.

타이밍 재구축

첫 번째 에지 (reconstructedTime1) 시간은 y-축 최적의 라인 인터셉트입니다. 모든 에지 타이밍을 재구축하려면 (그리고 reconstructedTime 어레이에 에지 타이밍을 저장하려면), 기계산된 인터셉트 a 및 슬로프 b 를 사용한 공식 $y = a + bx$ 를 적용하십시오.

```
reconstructedTime(1) = coef(2);

for index = 1:length(clocks)

  reconstructedTime(index)=a + (b * clocks(index));

end
```

지터 계산

다음은 재구축된 에지 타이밍 및 측정된 에지 타이밍 간의 타이밍 차를 발견함으로써 계산된 지터를 해결할 것입니다. 이 필드를 계산하기 위해, *jitter1.m* 에 다음을 추가하십시오:

```
jitter = reconstructedTime - measuredTime;
```

RMS 지터 계산

RMS (자승 평균 평방근) 지터를 계산하려면, 공칭 (average 또는 mean) 지터 어레이 [2] 를 지터 어레이 길이의 자승 평방근으로 나누십시오:

```
rmsJitter = norm(jitter)/sqrt(length(jitter));
```

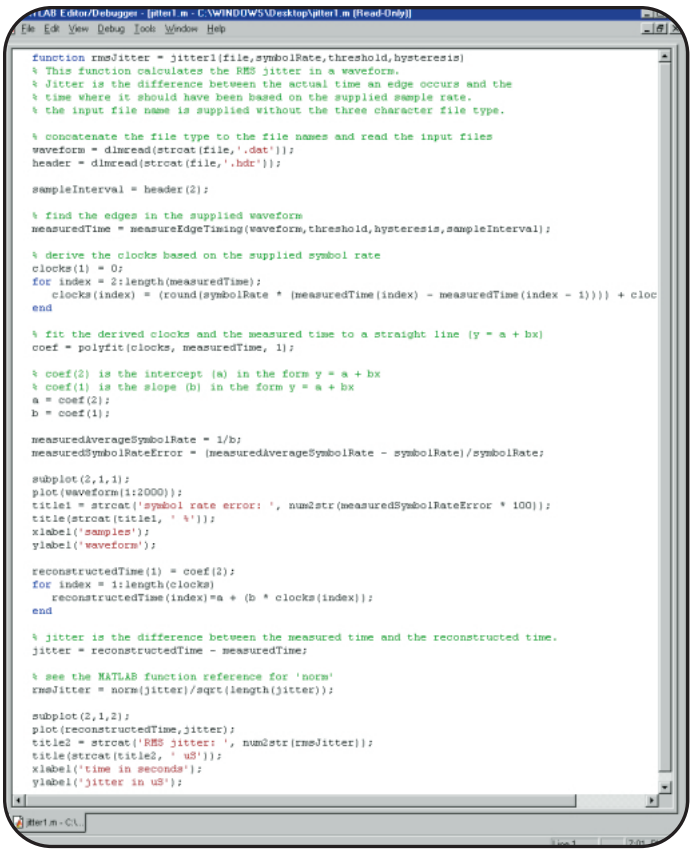
솔루션 그래프화

jitter1.m 에 다음과 같은 명령어를 추가함으로써 결과를 그래프화 하도록 MATLAB 을 실행할 수 있습니다:

```
subplot(2,1,2);

plot(reconstructedTime,jitter);

title2 = strcat('RMS jitter: ', num2str (rmsJitter));

title(strcat(title2, ' uS'));

xlabel('time in seconds');

ylabel('jitter in uS');
```

[2] 공칭 지터 어레이는 모든 평방근 지터 어레이 성분 합계의 자승 평방근입니다.

완성된 함수는 아래 그림과 같습니다. 이것을 호출하기에 앞서 *jitter1* 함수에 모든 변경 사항을 반드시 저장해야 합니다 (그림 3).



▶ ***그림 3:*** *MATLAB 의 완성된 jitter1 함수*

파형 및 샘플 시간간격 및 입력 값 가져오기

이제, 파형 데이터를 귀하의 오실로스코프에서 MATLAB 로 가져와서, 다음과 같은 입력 값과 헤더 파일 *(jitter5k.dat* 및 *jitter5k.hdr)* 및 파형 데이터의 이름을 포함한 *jitter1* 함수를 호출하겠습니다.

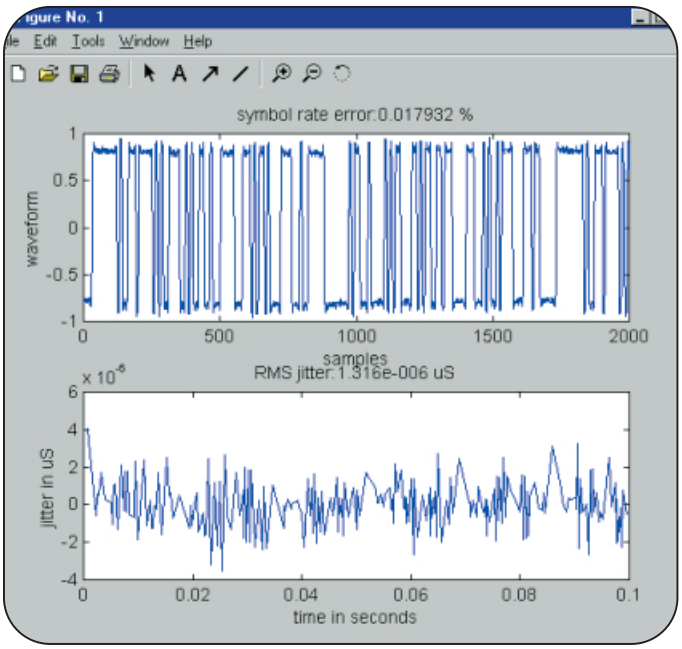
- ▶ 데이터를 발생시키기 위해 사용된 클럭 심볼 속도
- ▶ 파형이 논리 하이 (1)인지 로우 (0)인지를 결정하기 위해 사용된 임계값
- ▶ 획득 파형의 잡음을 막기 위하여 임계값 주변에 보호 대역을 구축한 히스테리시스

다음과 같은 구문을 사용하여 jitter1 함수 호출합니다:

jitter1 ('jitter5k', 5000, 0, .1)

MATLAB 은 *jitter1* 함수를 실행하고 *rmsjitter* 값으로 귀선 결과를 할당하며, 라인이 세미콜론 (;)으로 끝나지 않기 때문에, 즉시 Command Window 에 답을 디스플레이합니다.

또한 보다 많은 지점에서, MATLAB 은 아래 그림 (그림 4)에서 보는 바와 같이 2 개의 플롯화 한 그래프를 디스플레이합니다. 상위 그래프는 심볼 오류 속도인 반면에, 하위 라인은 RMS 지터입니다.



▶ ***그림 4:*** *MATLAB 의 파형 및 지터 그래프*

결론:

본 애플리케이션 노트에서는 파형 지터를 분석하기 위해 Tektronix TDS7000 제품 시리즈 DPO 와 MATLAB 을 함께 작동하는 방법을 살펴 보았습니다. 파형 획득 도구처럼 동일한 일체-완비 플랫폼 상에서 MATLAB 실행 기능을 포함하고 있는 TDS7000 제품 시리즈는 중요 지터 획득 및 특성화에 사용되는 최적의 도구입니다.

▶ *TDS7054*

▶ *TDS7104*

▶ *TDS7404*

상세 정보

텍트로닉스 (주)는 설계 엔지니어들이 최첨단 기술을 사용하여 작업할 수 있도록 애플리케이션 노트, 기술 요약집 및 기타 자료들을 광범위하고 지속적으로 유지 발전시키고 있습니다.

상세 정보는 당사 웹사이트 **www.tektronix.com** 의 "Resources For You" 부분을 방문해 주십시오.

www.tektronix.com

아시아 국가들 (65) 356-3900

호주, 뉴질랜드 61 (2) 9888-0100

오스트리아, 동유럽, 그리스, 터키, 몰타, 키프러스 +43 2236 8092 0

벨기에 +32 (2) 715 89 70

브라질, 남미 55 (11) 3741-8360

캐나다 1 (800) 661-5625

덴마크 +45 (44) 850 700

핀랜드 +358 (9) 4783 400

프랑스, 북아프리카 +33 1 69 86 81 81

독일 +49 (221) 94 77 400

홍콩 (852) 2585-6688

인도 (91) 80-2275577

이태리 +39 (2) 25086 501

일본 (소니/텍트로닉스 주식회사) 81 (3) 3448-3111

멕시코, 중앙 아메리카, 캐리비언 52 (5) 666-6333

네델랜드 +31 23 56 95555

노르웨이 +47 22 07 00

중국 86 (10) 6235 1230

폴란드 (48) 22 521 5340

한국 82 (2) 528-5299

남아프리카 (27 11) 651-5222

스페인, 포르투갈 +34 (91) 372 6000

스웨덴 +46 (8) 477 65 00

스위스 +41 (41) 729 36 40

대만 886 2722-9622

영국, 아이레 공화국 +44 (0) 1344 392000

미국 1 (800) 426-2200

기타 지역에서는, 1 (503) 627-1924 텍트로닉스 (주)로 연락하십시오.

ISO 9001

02/01 HMH/PG 55K-14593-0

Tektronix®