AN-7: The Ant18e Frequency and Event Counter

1.Summary

This application note introduces the 500MHz, 18-channel Ant18e Frequency and Event Counter - extra no-cost functionality included with the Ant18e 1GHz Logic Analyzer.

For each of the 18 input signals, this functionality enables the Ant18e to

- measure the signal's frequency
- measure the signal's period
- count signal events

2.Introduction

The Ant18e implements 18 frequency counters - one per channel. Each counter can measure incoming frequencies over a range which is better than 500MHz down to 0.01Hz:

- the high frequency limit is set by the input characteristics of the Ant18e, rather than by the speed of the internal logic.
- the low frequency limit is of the order of one event every few centuries - the internal counter registers are 64 bits each and the reference clock is 100MHz.

These are the frequency counter modes:

- *Frequency counting* mode is applicable to reasonably fast incoming signals - over 100Hz.
- *Period measurement* mode, sometimes described as inverse frequency counting mode, applies to lower frequencies - they are measured by determining the exact time between two edges of an incoming signal.
- *Event counting* mode is a generalization of frequency counting mode - the user defines a high/low pattern on some of the Ant18e signals. When the pattern is matched, the Ant18e counts rising edges. Rising edges are ignored when the pattern is not matched.

The variable threshold capability of the Ant18e applies in all modes. The input threshold can be set between 0.8V and 2.5V, in steps of 0.1V.

This is how the frequency counter operates internally:

- in frequency counter mode each channel starts off in an *idle* condition. When commanded to run, the internal gate signal is generated and the channel immediately transitions to a *busy* condition. Finally the channel moves to a *finished* condition when the gate period times out.
- in period counter mode each channel starts off in an *idle* condition. When commanded to run, the channel waits until it sees the first edge and then transitions to a *busy*

condition. Finally the channel moves to a *finished* condition when it sees the second edge

- in event counter mode each channel starts off in an *idle* condition. When commanded to run, the internal gate signal is generated and the channel immediately transitions to a *busy* condition. Finally the channel moves to a *finished* condition if the gate period times out. If the gate period is infinite, the counter remains in the *busy* condition until commanded to stop.

An application program can signal the frequency counter to stop, which causes any busy channels to enter the *forcedFinished* condition.

After a run each channel remains in a *finished* condition, with the count registers preserved, until the next run is started. When the next run is signaled each channel clears its count register, moves to *idle* and starts a new count.

Readback from the frequency counter is possible at any time and returns the following:

1. the status of the overall frequency counter
2. the status of each channel
3. the 64-bit counter value for each channel. As readback starts for each channel, a snapshot is taken of the status and the counter value. This means that the status agrees with the counter value and all the digits of the counter are valid.

2.1.Frequency Counting

This is the simplest case. The user chooses a *gate length*, which can be set at 0.1s, 1s, or 10s. The gate is common to all channels. The Ant18e then counts incoming rising edges when the gate is active. Typically, signals above 10kHz can use a 0.1s gate, signals above 1kHz can use 1s, and slower signals need to use the longer 10s gate.

The maximum frequency of the incoming signal for frequency counting is at least 500MHz.

WindowsXP can be quite sluggish - although measurements taken with a 0.1s gate will complete in 0.1s, there can be a delay of up to 5s before Windows schedules the task which is displaying the result.

In frequency counter mode, readback returns the number of input rising edges as follows:

- *busy*: the count so far
- *finished*: the count during the gate time
- *forcedFinished*: the count from the gate start to the forced stop

2.2.Period Measurement

In period measurement mode, the internal 100MHz clock measures the time between edges of an incoming signal. The measurement is not gated. With 64-bit registers, the time can be up to around 500 years, in steps of 10ns, though RockyLogic products are not guaranteed for this duration. The maximum frequency of the incoming signal for period measurement is 25MHz.

Rising and falling edges are both recognized, and the edge measurements can be between any two. Rising to rising, rising to falling, falling to falling, and falling to rising are all possible.

In period measurement mode, readback returns the number of 10ns periods between the defined edges as follows:

- *idle*: the first edge has not been seen. The readback is always zero.
- *busy*: the first edge has been seen but not the second. The readback is the time since the first edge, in units of 10ns.
- *finished*: both edges have been seen and the readback is the time from from the first edge to the second, in units of 10ns.
- *forcedFinished*: if non-zero, the time between the first edge and the forced stop, in units of 10ns.

2.3.Event Counting

In event counting mode, the user first sets a pattern on some or all of the channels. For instance, the pattern could be *high* on channels 0 to 3, *low* on channels 4 to 7, and *don't care* on the other channels. The default pattern is *don't care* on all channels.

Rising edges will be counted on all channels when the pattern is matched.

In addition the event counter can be *gated*. As with the frequency counter, the gate time can be 0.1s, 1s, and 10s. It can also be set to *infinite* to disable gating.

The maximum frequency of the incoming signal for event counting is at least 500MHz.

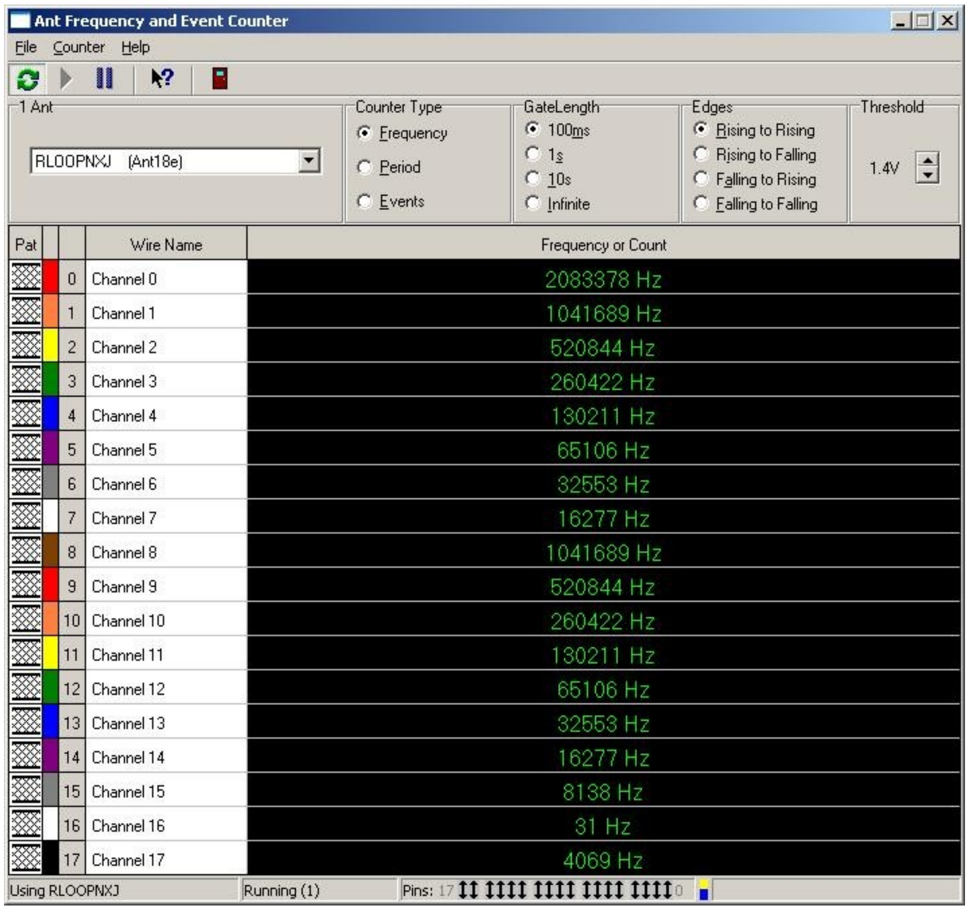
The maximum skew between the input signals and the pattern is +/-50ns; the signals which define the pattern should overlap the event signals by at least this amount.

In event counting mode, readback returns the number of input rising edges which occurred when the pattern was matched and the gate was enabled as follows:

- *busy*: the count so far
- *finished*: the count during the (potentially infinite) gate time
- *forcedFinished*: the count from the gate start to the forced stop

3.The antfc Program

The figure below shows the antfc.exe frequency counter application.



1 Ant: RLOOPNXJ (Ant18e)

Counter Type: [✓ Frequency] Period Events

GateLength: [✓ 100ms] 1s 10s Infinite

Edges: [✓ Rising to Rising] Rising to Falling Falling to Rising Falling to Falling

Threshold: 1.4V

Pat			Wire Name	Frequency or Count
▨	🟥	0	Channel 0	2083378 Hz
▨	🟧	1	Channel 1	1041689 Hz
▨	🟨	2	Channel 2	520844 Hz
▨	🟩	3	Channel 3	260422 Hz
▨	🟦	4	Channel 4	130211 Hz
▨	🟪	5	Channel 5	65106 Hz
▨	⬜	6	Channel 6	32553 Hz
▨	⬜	7	Channel 7	16277 Hz
▨	🟫	8	Channel 8	1041689 Hz
▨	🟥	9	Channel 9	520844 Hz
▨	🟧	10	Channel 10	260422 Hz
▨	🟨	11	Channel 11	130211 Hz
▨	🟩	12	Channel 12	65106 Hz
▨	🟦	13	Channel 13	32553 Hz
▨	🟪	14	Channel 14	16277 Hz
▨	⬜	15	Channel 15	8138 Hz
▨	⬜	16	Channel 16	31 Hz
▨	⬛	17	Channel 17	4069 Hz

Using RLOOPNXJ | Running (1) | Pins: 17 0

The help file for the program contains more detailed operating instructions. They key features depend on the counter mode.

- in *frequency counter* mode, set the threshold and the gate length. The active edges and the pattern are *don't care* in this mode.
- in *period measurement* mode, set the threshold and the active edges. The gate length and the pattern are *don't care* in this mode.
- in *event counter* mode, set the gate length, and the pattern (Pat). The active edges are *don't care* in this mode.

When the parameters are set, the counter can be run either single-shot or continuously.

4.Software Library Functions

The RockyLogic software support library is described in application note AN-6.

The frequency counting functionality is supported by these software library functions:

- RL_FCRun sets the parameters and starts or stops the frequency counter.

- RL_FCGetStatus returns the frequency counter status.
- RL_FCGetCount returns the per-channel status and the per-channel counts.

```
RL_RESULT RL_FCRun(RL_HANDLE h, int AOp, int AEdges, int AGateLength,
char *APat, bool ARun)
```

Parameters

h	handle returned from a previous RL_Initialize.
AOp	operation type. FC_OP_FREQ or FC_OP_PERIOD or FC_OP_EVENT.
AEdges	define the start and stop edges in period mode.
AGateLength	define the gate length in frequency counter mode.
Apat	pointer to the pattern for event counting.
ARun	true (1) to start, false (0) to stop.

Remarks

The parameter values for AOp, AEdges, and AGateLength are specified in antif.h

The pattern is a string of high ('1'), low ('0'), and don't care ('-') characters. Where necessary it is padded with don't care characters. The first character in the string applies to channel 0, the second to channel 1, and so on.

To force a counter stop, set ARun to 0 - the other parameters are *don't care* in this case.

In period mode, the counter measures from the A edge to the B edge. The constants for rising and falling A and B edges are OR'd together to form the AEdge value.

Example

```
RL_RESULT retval = RL_FCRun(h, FC_OP_PERIOD,
                   FC_EDGEA_RISING | FC_EDGEB_RISING,
                   0, "--------", true);
// counter now running in period measurement mode
```

```
RL_DLL RL_FCGetStatus(RL_HANDLE h, int *pStat, int Alen)
```

Parameters

h	handle returned from a previous RL_Initialize.
pStat	pointer to the buffer to receive the Frequency Counter status.
Alen	number of channels to report, typically 19 for the Ant18e.

Remarks

The first value (pStat[0]) has bit zero set if the frequency counter is busy, clear if idle. Other bits are reserved.

The per-channel status is encoded as follows (other bits are reserved) in the remaining ints – pStat[1] to pStat[18]:

```
bits 2..0   0      channel is idle
            1      channel is counting
            2      channel is finished
            3      reserved
            4      channel was forced to stop by host command (forcedFinished)
            5 to 7 reserved
bit 3       set if the A edge has been seen
bit 4       set if the B edge has been seen
```

Example

```
            int status[1+18];
            RL_RESULT retval = RL_FCGetStatus(h, status, 19);
            bool CounterIsBusy = ((status[0] & 1) == 1);
```

```
RL_RESULT RL_FCGetCounts(RL_HANDLE h, __int64 *pVal, int AChans);
```

Parameters

h	handle returned from a previous RL_Initialize.
pVal	pointer to the buffer to receive the per-channel counter values.
AChans	number of channels to read, starting from channel 0. Usually 18.

Remarks

The per-channel counter values are 64-bit unsigned integers. The current count value can be read back wile the counter is running.

Example

```
            __int64 val[18];
            RL_RESULT retval = RL_FCGetCount(h, val, 18);
```

5. Revision History

Date	Revision
14 June 2006	Initial Version
21 February 2007	Library functions updated

For more information see www.rockylogic.com