Jennic

TECHNOLOGY FOR A CHANGING WORLD

Time-of-Flight API User Guide

JN-UG-3068

Revision 1.0

6-Oct-2009

Contents

About this Manual

This manual introduces the Time-of-Flight (ToF) mechanism of the Jennic JN5148 wireless microcontroller and describes the supplied ToF Application Programming Interface (API) which allows access to this functionality.

Organisation

This manual consists of five chapters, as follows:

- Chapter 1 introduces the JN5148 Time-of-Flight (ToF) mechanism.
- Chapter 2 outlines the usage of the ToF API.
- Chapter 3 deals with design considerations of developing a location system.
- Chapter 4 details the ToF API functions.
- Chapter 5 details the data structures used by the ToF API.

Conventions

Files, folders, functions and parameter types are represented in **bold** type.

Function parameters are represented in *italics* type.

Code fragments are represented in the Courier typeface.

Acronyms and Abbreviations

API	Application Programming Interface
RSSI	Received Signal Strength Indication
SQI	Signal Quality Indication
ToF	Time of Flight
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance

Related Documents

[R1] Jennic Time of Flight Ranging Demo [JN-AN-1143]

Feedback Address

If you wish to comment on this manual, or any other Jennic user documentation, please provide your feedback by writing to us (quoting the manual reference number and version) at the following postal address or e-mail address:

Applications
Jennic Ltd
Furnival Street
Sheffield S1 4QT
United Kingdom

doc@jennic.com

1 Introduction

The Jennic JN5148 wireless microcontroller includes a hardware Time-of-Flight (ToF) engine that allows measurement of the time taken for a 2.4-GHz radio signal to travel between two nodes. As this time is proportional to the distance traversed by the signal, it can be used to estimate the range from one node to another. This, together with signal strength measurements, can be used to develop location-aware systems.

1.1 ToF Mechanism

To perform a ToF measurement between two nodes, the local node sends a packet to the remote node. The remote node automatically sends an acknowledgement (ACK) in response to this packet.

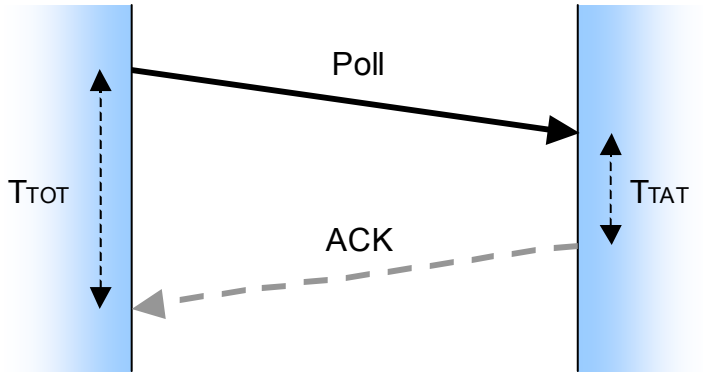

Figure 1: Time-of-Flight Measurement

The local node is able to measure the time from sending the poll to receiving the acknowledgement, which gives the total time T_{TOT}. In addition, the remote node records how long it took to respond to the poll, or the turnaround time, T_{TAT}. Subtracting this from the total time then gives the time both packets spent in flight or the round trip time, T_{RTT}. It can be assumed that each direction took an equal amount of time and thus the time of flight, T_{TOF}, is equal to half the round trip time, as indicated in Equation 1 below.

$$T_{ToF} = \frac{T_{RTT}}{2} = \frac{T_{TOT} - T_{TAT}}{2}$$

Equation 1: Time-of-Flight Calculation

As the ToF measurement system relies on the measurement of time at both local and remote node, it can be affected by clock frequency offsets at each end. To reduce the impact of any such errors, the API provides a facility to take measurements in reverse, whereby the remote node sends the poll and the local node acknowledges. By averaging the results obtained in the forward and reverse directions, it is possible to cancel the effects of any frequency offsets.

There are additional delays introduced while the packets are passing through the radio. However, these are largely constant and are removed by the API. Retries are disabled during each ToF poll and although CSMA/CA back-offs are still used, they occur before the measurement and are not included in the timing, although they do affect the total duration of each measurement.

1.2 ToF Data Transfers

The ToF calculation relies on co-operation between the two nodes to measure and transfer the timing data. A burst of one or more ToF measurements is co-ordinated by the ToF API through additional packet transfers, as illustrated in Figure 2.

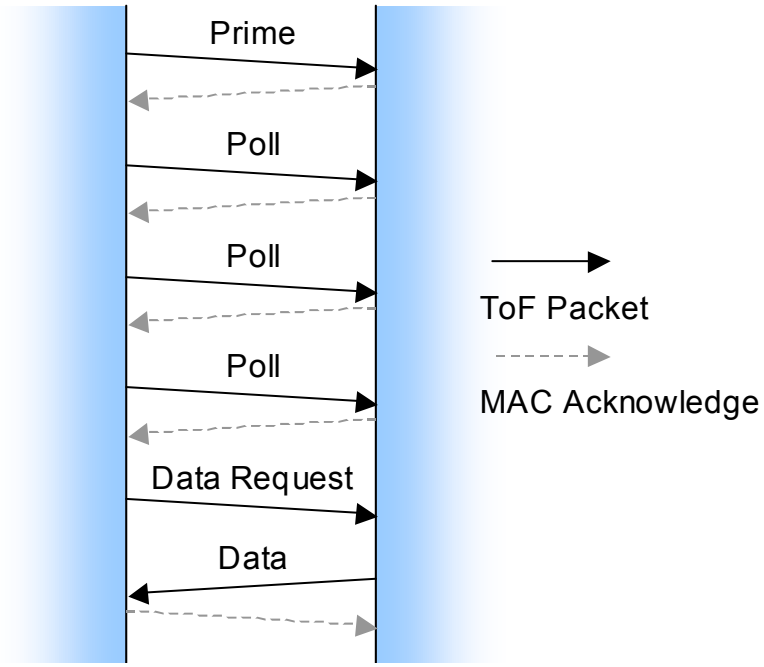

Figure 2: Forward ToF Packet Transfers

The ToF burst is initiated by transmitting a Prime command to the remote node. This is followed by a number of ToF measurements consisting of a Poll command and Acknowledgement, as detailed in Section 1.1. If a reverse direction burst has been requested, the Poll/Acknowledge pairs are reversed.

After the ToF readings have been completed, the local node requests and receives the timing and signal data from the remote node. The final Data packet can contain results for up to 11 ToF readings. When more than 11 readings are taken, extra Data packets are used to transfer the results.

A single Poll/Acknowledge pair within the burst is approximately 4 ms in duration. The Prime, Data Request and Data packets collectively add an extra 6 ms to the overall duration of the burst.

2 Using the ToF API

The ToF API provides functions to configure and use the Time-of-Flight engine in the JN5148 device. Fully working example code can be found in the Application Note JN-AN-1143, *Time-of-Flight Ranging Demo*.

2.1 Initialising the ToF API

Before the ToF hardware and API can be used, they must be initialised by calling **vAppApiTofInit()**. This must be done on all nodes that are intended to be used in taking ToF readings.

```
PUBLIC void AppColdStart(void)
{

    ...


    /* Perform general system initialisation */
    vInitSystem();

    /* Enable TOF ranging. */
    vAppApiTofInit(TRUE);

    ...

}
```

2.2 Configuring the ToF API

The two functions **s32AppApiTofGetCalloffset()** and **vAppApiTofSetCalloffset()** allow the configuration of the calibration offset within the ToF API. This offset is measured in picoseconds and is applied to the ToF measured in all readings taken by the API. In this way, it is possible to compensate for extra signal delays introduced into the system.

It is important to note that the offset provided is applied to the round-trip time and so should be the sum of any delays in the transmit and receive paths. For example, in a system with the antenna mounted away from the device at the end of a one-meter cable, it would be necessary to set the offset to twice the delay present in the cable.

2.3 Taking ToF Readings

A single function call to **bAppApiGetTof()** initiates a burst of up to 255 ToF measurements to another node. An array of `tsAppApiTof_Data` structures must be allocated and a pointer to it passed to the API. This array will be filled with the data collected.

The following example code illustrates initialising the appropriate variables and then requesting a burst of 255 readings in the forward direction.

```
/* Flag to indicate measurement active */
volatile bool_t bTofInProgress = FALSE;
/* Array of result structures */
tsAppApiTof_Data asTofData[255];
MAC_Addr_s sAddr;

/* Set eTofStatus to invalid value.
   Will be updated in ToF callback */
eTofReturn eTofStatus = -1;

/* Fill structure with address of target node */
sAddr.u8AddrMode = 2;
sAddr.u16PanId = PAN_ID;
sAddr.uAddr.u16Short = COORDINATOR_ADR;

/* Test if a measurement is already in progress */
if(bTofInProgress == FALSE)
{
    if (bAppApiGetTof(
          asTofData,
          &sAddr,
          255,
          API_TOF_FORWARDS,
          vTofCallback))
    {
      /* Set flag to indicate measurement underway */
      bTofInProgress = TRUE;
      bTofDirection = API_TOF_FORWARDS;
    } else {
      /* Failed to start ToF measurement */
    }
}
```

2.4 Processing the ToF Data

When a ToF burst has been started, the API co-ordinates the necessary packet transfers. The callback function that is passed when calling **bAppApiGetTof()** can be used to indicate when the burst has completed and the ToF data is ready for processing:

```
void vTofCallback(eTofReturn eStatus)
{
    eTofStatus = eStatus;
}
```

Once the data has been collected, it can be processed. The code fragment below demonstrates processing the data to find the mean ToF to a node from 255 measurements. To ensure only valid results are processed, it is necessary to check both the return code provided in the callback function and the individual measurement status codes.

```
/* Has callback indicated completion of burst */
if(eTofStatus != -1)
{
    /* Perform processing if burst sucessfull */
    if (eTofStatus == TOF_SUCCESS)
    {
        u8NumErrors = 0;

        for(n = 0; n < 255; n++)
        {
            if(asTofData[n].u8Status == MAC_TOF_STATUS_SUCCESS)
            {
                s32Sum += asTofData[n].s32Tof;
            }
            else
            {
                u8NumErrors++;
            }
        }

        if(u8NumErrors != 255)
        {
            dMean = s32Sum / (255 - u8NumErrors);
        }
        else
        {
            dMean = 0;
        }
    }

    /* Clear flag to allow next measurement */
    bTofInProgress = FALSE;
    eTofStatus = -1;

}
```

2.5 Interpreting ToF Data

The ToF API does not return range data directly. Instead, it is necessary to interpret the ToF and RSSI data to estimate the range to a node and judge the quality of the estimate.

2.5.1 Time-of-Flight Data

At a basic level, the 2.4-GHz radio signals can be assumed to travel at the speed of light (299,792,458 m/s) and so distance is directly proportional to the measured ToF. As the ToF reading returned by the ToF API is measured in picoseconds, it can be converted to a distance in meters by multiplying by 0.0003. At short ranges, the ToF measurement may result in negative values.

Note: The Time-of-Flight mechanism estimates the time taken by the radio signal to travel along the path between two nodes. This means that the ToF-based distance only equates to the range between two devices if the signal follows only a direct path. If the radio signal is instead received after reflecting off an intervening object, the measured distance will be greater than the actual range. Generally, the received signal will be a combination from a direct path and several indirect paths (multi-path). Strong multi-path interferers may adversely affect the estimated ToF. Small changes in node position or radio environment may also affect the ToF and so it is important to consider taking multiple readings per position.

2.5.2 Received Signal Strength Indication (RSSI) Data

The Received Signal Strength Indication (RSSI) represents the power of the received signal in the radio with 1dB resolution as shown in Figure 3. The maximum obtainable RSSI value is 108. The intrinsic noise floor of the radio receiver limits the minimum reported RSSI value to approximately 20.

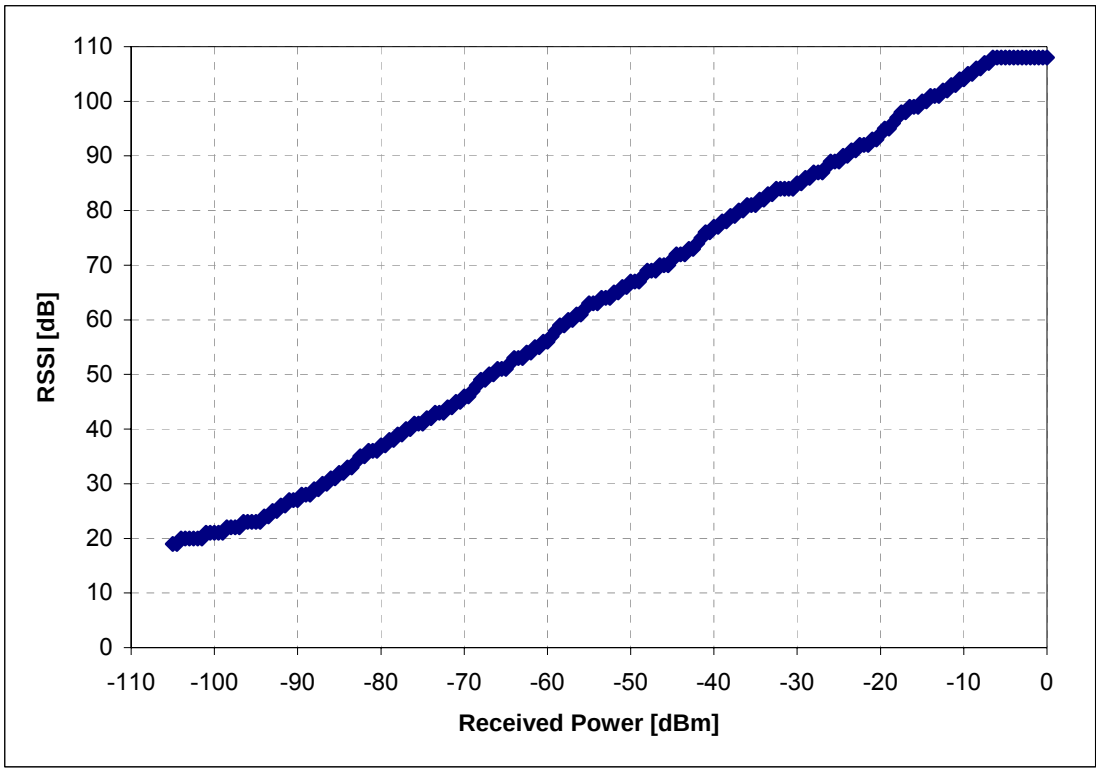

Figure 3: RSSI versus Received Signal Power

The power of the received signal, and hence RSSI, decreases with increasing distance from the source of transmission. Therefore, the RSSI may be used to estimate the distance between a receiver and a transmitter with a known output power.

However, the received signal power typically has an inverse-square law relationship with distance. Therefore, as distance increases, the absolute rate of change in RSSI (with respect to distance) decreases. As a result, the RSSI will become an increasingly inaccurate indicator of range. In general, RSSI has been found to provide greater accuracy than Time-of-Flight at ranges less than 10 m.

It is important to consider the selection of a model for the RSSI to distance conversion appropriate to the environment in which the system will be used. There are several IEEE propagation models developed for 2.4-GHz wireless communications that are relevant to IEEE 802.15.4-based location.

For example, the *Time-of-Flight Ranging Demo* Application Note (JN-AN-1143) uses the inverse-square law based model shown in Equation 2 and plotted in Figure 4. This model is only valid when using standard-power modules - high-power modules will cause ranges to be under-estimated.

$$Range = 0.02 \times 10^{\left(\frac{108\text{-}RSSI}{20}\right)}$$

Equation 2: Example RSSI to Distance Model

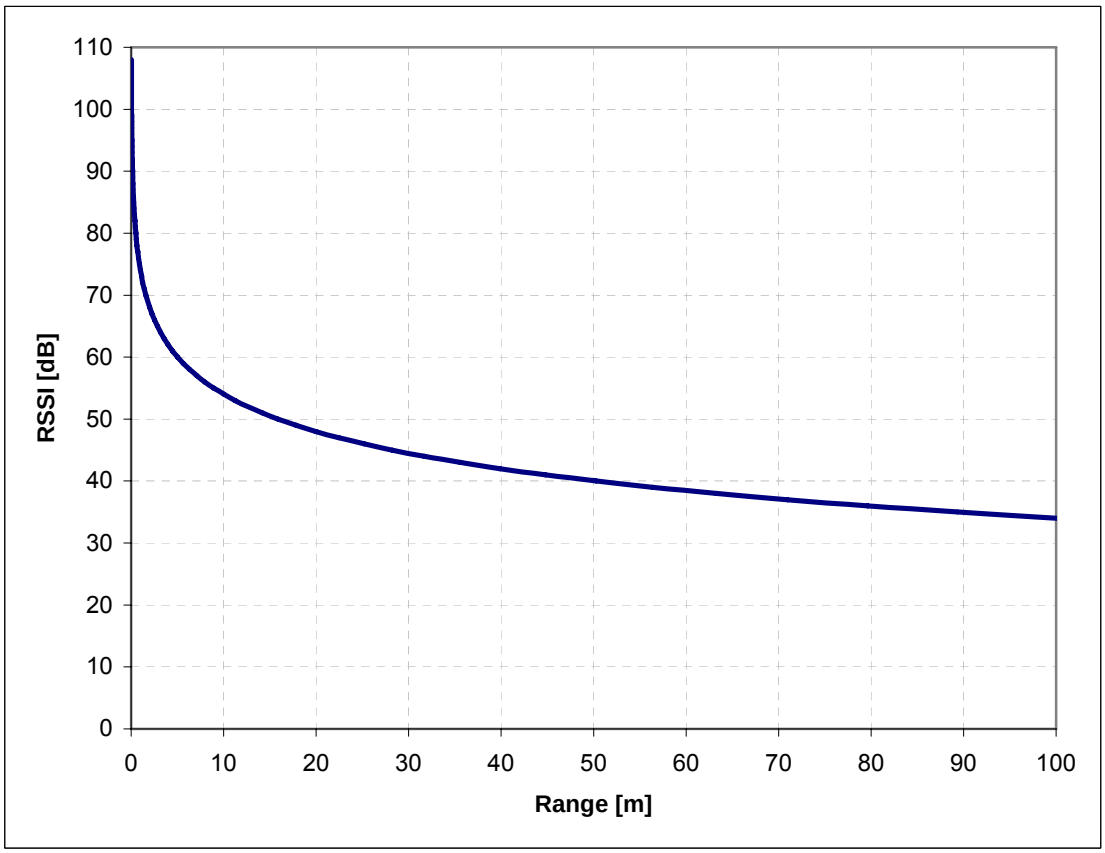

Figure 4: Example RSSI to Distance Model

Note: The signal strength measured at either device is dependent on many factors in addition to distance. Irregularities in the antenna's radiation pattern can cause orientation to alter the signal strength. Any obstructions will also attenuate the signal by varying degrees.

The IEEE 802.15.4 standard Link Quality Indication (LQI) is derived from the RSSI using the following pseudo-code:

```
LQI = (RSSI – 20) x 3;

if (LQI < 0)
  LQI = 0;
else if (LQI > 255)
  LQI = 255;
```

This allows LQI values to be directly compared with RSSI values as supplied by the ToF API.

2.5.3 Signal Quality Indication (SQI) Data

The Signal Quality Indication (SQI) is a value in the range 0 to 255 that represents the correlation quality of the received signal. An SQI value above 200 generally indicates a good signal. The SQI of each ToF value can be used to evaluate the reliability of the measurement.

2.5.4 Timestamp Information

The timestamp data provides a time indication for each measurement with a 16-µs resolution. The timestamp is produced by the node that generates the ToF poll packets. The local and remote nodes are not synchronised and will therefore produce different timestamps.

3 Location System Development

It is possible to develop location-aware systems using the ToF API and JN5148 device. This can be done by taking range measurements from a tag node to reference nodes with a known location. It is then possible to combine these range measurements to estimate the location of the node. However, several factors must be taken into account in order to develop a successful system.

3.1 Finding Reference Nodes

The first challenge in developing a location system is finding the reference nodes that will be ranged to by the tag node which is to be located. The methods that could be used to do this vary between networking stacks, as indicated below.

3.1.1 IEEE 802.15.4

If no networking layer is being used (just IEEE 802.15.4), a data packet can be broadcast by the tag node to all nodes within range. The application in the reference nodes can respond to this packet, supplying the tag node with their addresses and any other appropriate data. These responses can also provide the tag node with signal strength data, allowing the tag to more intelligently select the reference nodes to use.

3.1.2 JenNet

JenNet is not currently supported on the JN5148 device.

3.1.3 6LoWPAN

6LoWPAN is not currently supported on the JN5148 device.

3.1.4 ZigBee PRO

Sleeping End Devices in a ZigBee PRO network are only able to communicate directly with their parent node, so directly discovering nodes in range is not possible. Sleeping End Device nodes will therefore need to obtain a list of potential nodes to range against from their parent, which could be done in one of the following ways:

- For small networks, a complete list of ToF nodes in the network could be supplied.
- Parent nodes could simply supply the list of all ToF nodes that are in range of the parent. However, this could exclude ToF nodes that are in range of the End Device but not its parent.
- Parent nodes could hold a list of neighbouring reference nodes. In addition, for each neighbouring node, a further list of neighbouring reference nodes may be obtained. This information may be used to provide a list of all reference nodes with which an End Device could potentially communicate.

3.2 Performing Range Measurements

Once the tag has a list of reference node addresses, it is able to perform ranging. Again, there are options relating to how this is performed.

- The tag node could work down the list, attempting a range measurement to each of these nodes.
- If the tag is given the reference node locations as well as their addresses then it may be possible to process the reference node data intelligently. For example, if no response is received from a certain reference node, it may be assumed that it is out-of-range. Therefore, it would be better to attempt to range with a reference node that is not very close to the one that did not respond.
- A record of reference nodes that previously responded could be maintained on the tag node. Then when a new location operation begins, these fixed nodes could be addressed first.

To perform range estimation to one reference node, a ToF API function is used. With this function, it is possible to perform multiple range estimates in either the forward or the reverse direction.

3.3 Packaging Results for Transmission

The ranging measurements generate a significant amount of results data. It is therefore important to consider how the data will be packaged and transmitted if the processing is to be performed centrally.

A location system may not need to make use of all the available data and some data may be discarded before transmission. It may also be possible to perform some or all of the data processing, including averaging or filtering the results, on the tag node in order to reduce the volume of data to be transmitted.

3.4 Configuring the Location System

Any location system is likely to need a certain amount of configuration capability to adapt it to the installation environment and system goals. This may include parameters such as:

- Frequency of location measurements
- Number of reference nodes required for location
- Number of range readings for location
- Direction of readings

4 ToF API Functions

The Time-of-Flight (ToF) API comprises the following functions, which are described on the referenced pages:

vAppApiTofInit

void vAppApiTofInit(bool_t *bEnable***);**

Description

This function can be used to enable or disable the Time-of-Flight mechanism.

Time-of-Flight must be enabled on both devices that are to be involved in a measurement, before starting the measurement.

Parameters

bEnable	Enables/disable Time-of-Flight mechanism: TRUE to enable ToF FALSE to disable ToF

Returns

None

bAppApiGetTof

```
bool_t bAppApiGetTof(
                    tsAppApiTof_Data *pTofData,
                    MAC_Addr_s *pAddr,
                    uint8 u8NumReadings,
                    bool_t bDirection,
                    PR_TOF_APPCALLBACK prTofCallback);
```

Description

This function initiates a set of ToF readings between the node on which it is called and a remote node.

Parameters

pTofData	Pointer to array of structures to hold ToF data. There must be sufficient space for the number of readings requested (see below).
pAddr	Pointer to address structure containing the address of the device to which the ToF will be performed.
u8NumReadings	The number of ToF readings to take (1 to 255).
bDirection	Direction of ToF experiment: API_TOF_FORWARDS - from local to remote node API_TOF_REVERSE - from remote to local node
prTofCallback	Pointer to callback function to be called when ToF readings are complete.

Returns

TRUE - ToF started

FALSE - ToF failed to start

s32AppApiTofGetCalloffset

```
int32 s32AppApiTofGetCalloffset(void);
```

Description

This function retrieves the calibration offset that is applied to all ToF measurements.

Parameters

None

Returns

Returns the calibration offset in picoseconds.

vAppApiTofSetCalloffset

void vAppApiTofSetCalloffset(int32 *s32pSecs***);**

Description

This function sets the calibration offset that is applied to all ToF measurements.

Note: The offset supplied is subtracted from the round-trip time in ToF measurements involving this node. It should therefore be the sum of any delays present in the transmit and receive paths of this node.

Parameters

s32pSecs	The calibration offset, in picoseconds.

Returns

None

5 ToF API Type Definitions

5.1 tsAppApiTof_Data Structure

The `tsAppApiTof_Data` structure is used to return the ToF and signal data gathered by the ToF API during a single reading.

```
typedef struct
{
    int32   s32Tof;
    int8    s8LocalRSSI;
    uint8   u8LocalSQI;
    int8    s8RemoteRSSI;
    uint8   u8RemoteSQI;
    uint32  u32Timestamp;
    uint8   u8Status;
} tsAppApiTof_Data;
```

where:

- `s32Tof` is the measured time of flight, in picoseconds.
- `s8LocalRSSI` is the RSSI measured at the local node.
- `u8LocalSQI` is the SQI measured at the local node.
- `s8RemoteRSSI` is the RSSI measured at the remote node.
- `u8RemoteSQI` is the SQI measured at the remote node.
- `u32Timestamp` is a timestamp of when the measurement was taken
- `u8Status` is a bitmap containing:

Status	Bit	Reason
MAC_TOF_STATUS_SUCCESS	0	Reading completed successfully
MAC_TOF_STATUS_RTE	1	Remote time value invalid
MAC_TOF_STATUS_LTE	2	Local time value invalid
MAC_TOF_STATUS_NOACK	3	Failed to receive acknowledgement
MAC_TOF_STATUS_DATAERROR	4	Failed to receive data from remote node

5.2 PR_TOF_APPCALLBACK

PR_TOF_APPCALLBACK defines the prototype to be used for a callback function passed to **bAppApiGetTof()**. This callback function will then be called when the ToF burst has completed, with a status code indicating success or failure.

```
typedef void (*PR_TOF_APPCALLBACK)(eTofReturn eStatus);
```

5.3 eTofReturn Enumerated Type

The `eTofReturn` type enumerates the possible status codes for a completed ToF burst.

```
typedef enum
{
    TOF_SUCCESS,
    TOF_FAIL,
    TOF_NO_VALID_SEQUENCES,
    TOF_NOT_STARTED,
    TOF_TX_ERROR
} eTofReturn;
```

Revision History

Version	Date	Description
1.0	6-Oct-2009	First release

Important Notice

Jennic reserves the right to make corrections, modifications, enhancements, improvements and other changes to its products and services at any time, and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders, and should verify that such information is current and complete. All products are sold subject to Jennic's terms and conditions of sale, supplied at the time of order acknowledgment. Information relating to device applications, and the like, is intended as suggestion only and may be superseded by updates. It is the customer's responsibility to ensure that their application meets their own specifications. Jennic makes no representation and gives no warranty relating to advice, support or customer product design.

Jennic assumes no responsibility or liability for the use of any of its products, conveys no license or title under any patent, copyright or mask work rights to these products, and makes no representations or warranties that these products are free from patent, copyright or mask work infringement, unless otherwise specified.

Jennic products are not intended for use in life support systems/appliances or any systems where product malfunction can reasonably be expected to result in personal injury, death, severe property damage or environmental damage. Jennic customers using or selling Jennic products for use in such applications do so at their own risk and agree to fully indemnify Jennic for any damages resulting from such use.

All trademarks are the property of their respective owners.

Jennic Ltd
Furnival Street
Sheffield
S1 4QT
United Kingdom

Tel: +44 (0)114 281 2655
Fax: +44 (0)114 281 2951
E-mail: info@jennic.com

For the contact details of your local Jennic office or distributor, refer to the Jennic web site:

www.Jennic.com
TECHNOLOGY FOR A CHANGING WORLD