**Real Time Location System介绍**

<http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/localization-index-cc13x2_26x2.html#sec-tof-rtls-ctrl>

1. 简介

RTLS（实时定位系统）可在TI CC26xx系列的标准蓝牙低功耗无线电中实现。这些技术提供了可用于开发定位的算法和限定与其他BLE节点安全范围的原始数据。 RTLS中包含的三种主要技术是：RSSI，蓝牙core spec 5.1规定的到达角【angle of arrival】和TI的飞行时间【time of flight】。

RSSI是指传入信号的接收信号强度指示，通常通过定位算法中的三边测量过程来利用RSSI得出接收器与发送器之间的距离。蓝牙低功耗协议栈使开发人员能够收到蓝牙数据包的RSSI。

到达角AOA是一种用于确定传入的蓝牙数据包来自何方的技术，从而为三角测量奠定了基础。该设备对传来的恒定音【constant tone】进行采样并作为I / Q数据。该原始I / Q数据表示信号的幅度和相位数据，并且该数据可用于得出设备发送恒定音调的角度。

TI Time of Flight是一项专有技术，用于通过测量RF数据包交换的往返延迟来保护设备的安全范围。通过采集多个样本，可以提取精度更高的结果，从而为开发人员提供数据以帮助对设备进行三边测量。

有关AOA例程的详细信息，请参见simplelink\_cc13x2\_26x2\_sdk\_x\_xx\_xx\_xx / examples / rtos / CC26X2R1\_LAUNCHXL / ble5stack / PROJECT文件夹中的相关README.html文件。

利用RTLS工具箱提供的原始数据，开发人员能够通过获得更多可用于三边测量和三角测量【trilateration and triangulation】的数据来改进基于蓝牙技术的定位算法。

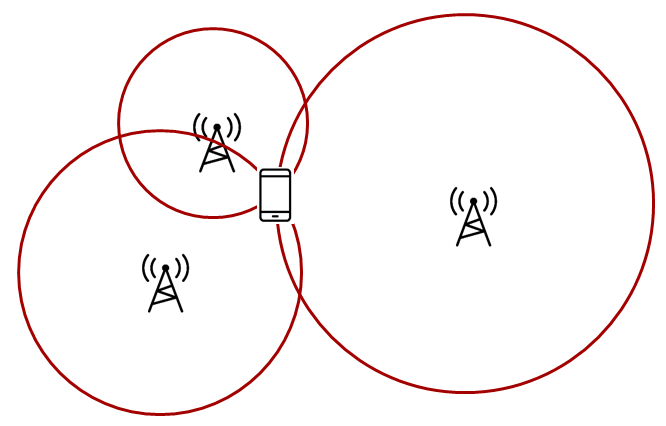
CC13x2/CC26x2 RF core的灵活性是实现功能扩展的原因。使用CC13x2或CC26x2的主要优势在于，客户可以以很少的额外成本，很少的额外能耗和不增加峰值功率的方式添加RTLS功能。

定位算法有如下两种：

1. 三边测量

三边测量适用于知道参考节点与目标节点之间的距离的应用。 这意味着一个定位器看到的可能位置是一个圆，因此通常需要三个定位器才能找到一个共同的相交点。 （假设是2D场景）

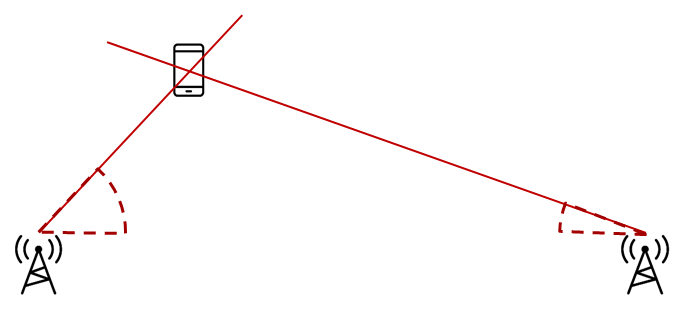
TOF提供了从接收器到发射器的距离。



1. 三角测量

三角测量适用于知道从参考节点到目标节点的方向的应用。 这意味着一个定位器看到的可能位置是一条直线，因此两个节点就足以确定单个相交点。 （假设是2D场景）

AOA提供了从接收器到发射器的角度。



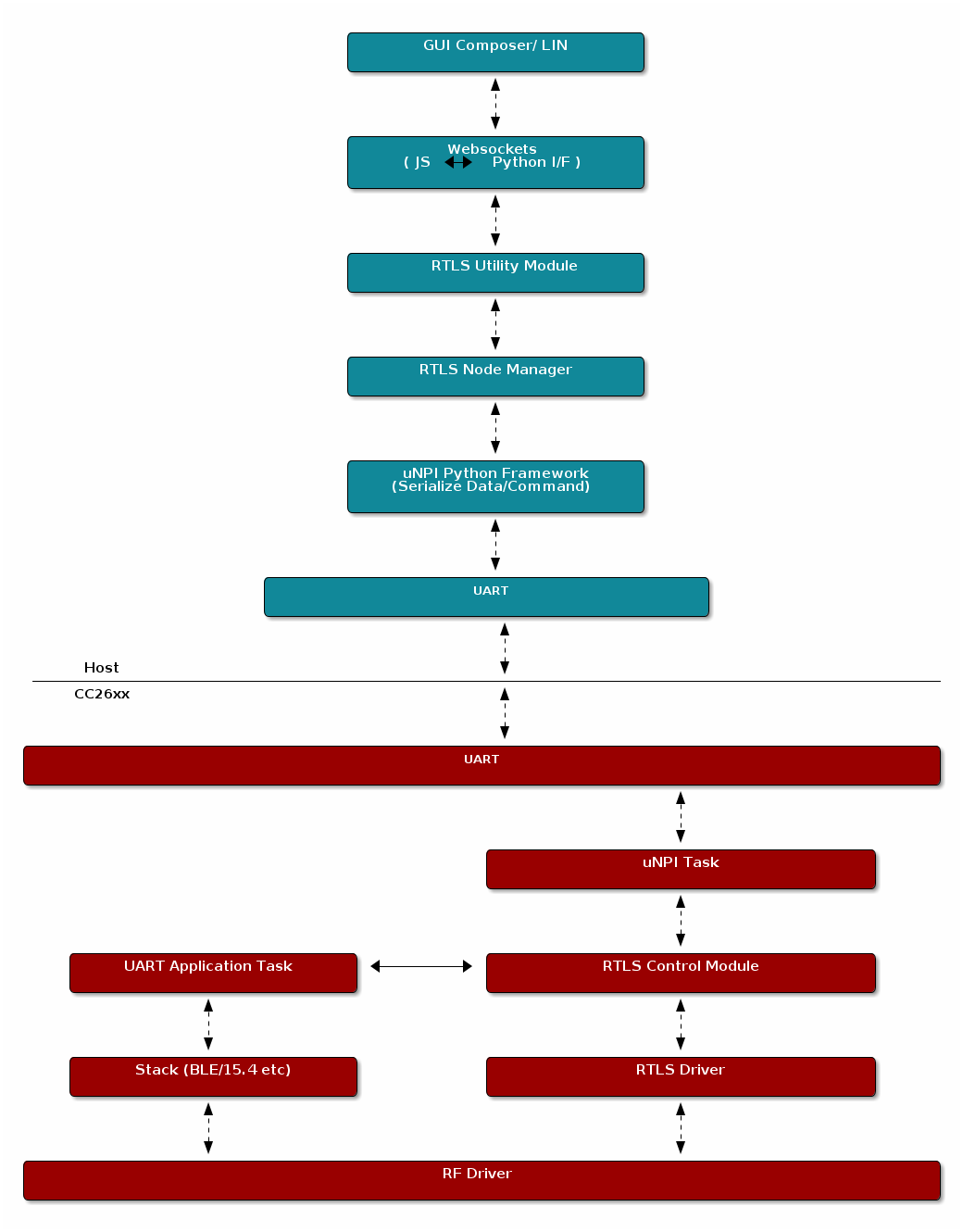
注意：没有外部控制接口的现有示例已停用。 现在，所有RTLS示例都具有一个外部控制接口。

二、常规RTLS软件架构

对于具有多个定位节点的应用，拥有支持这些节点控制体系的结构非常重要。必须可以自主地配置和触发定位，并且必须可以检索定位数据。我们通过重用网络处理器接口（NPI）来实现这一目标。这是一种通过串行接口支持远程过程调用（Remote Procedure Calls --RPC）的方法。注意，可以用您自己选择的串行协议替换NPI。

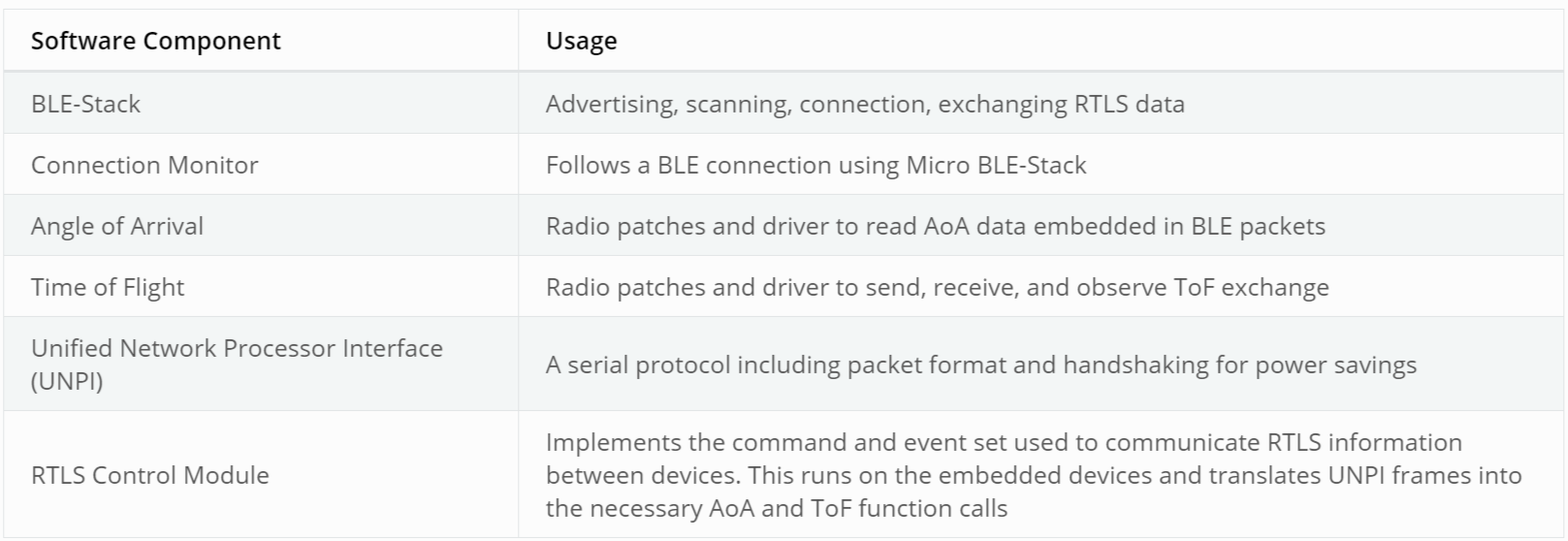
在我们的示例中，我们使用UART作为串行传输层。这是因为它很容易在主机PC上作为USB上的UART来使用，而我们的LaunchPads包含UART到USB的桥接器。除了例程之外，我们还提供PC软件作为主机，用来控制，检索和呈现定位数据。可以更快地进行用户性能展示，配置重要参数，且具备开发和测试更高级别的后处理算法的能力。

下图显示了RTLS软件体系架构。

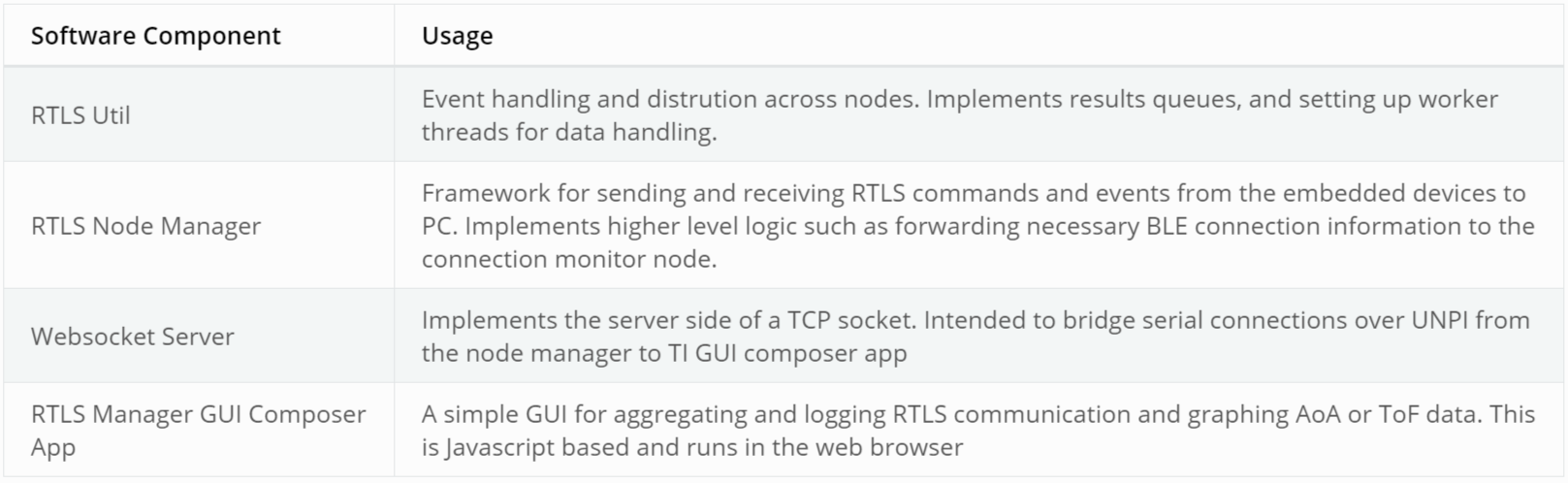


2.1、RTLS Toolbox

RTLS Toolbox是专用于定位的软件集合。下表总结了工具箱中的每个软件组件及其在定位上的应用。这些软件组件将在嵌入式节点上运行。



下表中的软件组件在PC上运行。 TI已建立基于PC的环境，以帮助评估和原型化各种RTLS算法。 一旦算法完成，即可将各种PC组件迁移到嵌入式设备。



RTLS Utility

RTIS Util模块在节点管理器的上层，根据RTLS网络的要求（例如种子分发）实施事件处理或阻塞。另外，RTLS UTil模块能够设置工作线程以对来自设备的结果进行数据处理，还可以创建队列来存储传入的数据。

RTLS Node Manager【节点管理器】，RTLS节点管理器提供以下功能：

GUI编写器与CC13x2或CC26x2之间的桥梁

保持每个设备的状态并负责配置设备

使RTLS控制模块和GUI 编写器变得简单

最小化GUI 编写器与系统其余部分之间的事务量

由于具有上述功能，节点管理器可以在用户的​​CPU上运行，并有助于处理慢速总线（LIN / CAN）。

RTLS控制模块

RTLS控制模块是作为RTOS任务运行的片上模块，它提供以下功能：

解析来自节点管理器的命令

RTLS驱动程序配置和操作

RF和NPI消息队列

2.2、RTLS Roles and Topology【拓扑】

RTLS网络中的每个节点都以不同的方式利用上面列出的软件组件来执行与定位有关的特定任务。 这些功能映射到RTLS网络中的不同角色，并最终由SDK中的示例应用程序实现。一共有三个示例：rtls\_master，rtls\_slave和rtls\_passive。 这些示例的功能如下所述。 所有嵌入式节点都实现UNPI并充当UNPI协议中的从设备。 此外，所有嵌入式节点都具备RTLS控制模块，用于处理来自UNPI接口的RTLS命令。

注意：以下小节旨在描述SDK中的示例应用程序实现了哪些定位角色。 这不是每个节点上可执行操作的完整列表，而是对示例程序实现的具体功能进行说明。

2.2.1、Master

RTLS主设备运行完整的BLE stack，并充当central设备。它将扫描并通过BLE连接到RTLS从设备。建立连接后，RTLS master将执行以下操作：

--与PC共享连接参数（访问地址，主睡眠时钟精度和CRC初始化）。

--通过BLE与从设备共享ToF和AoA参数。

--实现ToF master角色

--不发送AoA数据包，而是配置从设备发送AoA数据包。

2.2.2、Slave

RTLS从设备运行完整的BLE stack，并充当BLE从设备。这是需要被定位的设备。从设备将广播并与RTLS主设备建立连接。RTLS slave将执行以下操作：

--使用恒定音扩展（Constant Tone Extension --CTE）发送嵌入了AoA tone的数据包

--广播发送需由rtls\_master检测到的特殊字符串

--实现ToF slave角色

--BLE-Stack的从设备

--无线/电池供电，不连接到PC

2.2.3、Passive

RTLS被动设备不参与RTLS主机和从机之间的BLE连接。相反，它在连接监视模式下使用Micro BLE stack来跟踪连接。因此，一旦建立连接，被动设备就需要依靠主设备来传递连接参数。被动节点执行以下操作：

--实现ToF 被动角色

--接收具有CTE的数据包并执行同相和正交分量（IQ）采样

--实现Micro BLE-Stack连接监视应用层。有关连接监视器的更多信息，参考：<http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/u-stack/functional-description.html#sec-cm-app>

2.2.4、PC/Central Processing Node

PC节点负责通过发送命令和处理事件来控制嵌入式RTLS节点。在SDK中，这是通过将实现UNPI主角色的Python层和Websocket服务器（将UNPI命令转换为浏览器中运行的GUI Composer应用程序使用的套接字接口）的结合实现的。该软件作为一个框架，旨在从嵌入式节点提取数据并将其用作PC上高级RTLS算法的原型。

注意：在最终产品中，这些算法可以在嵌入式设备甚至RTLS master节点上实现。 TI提供了PC软件来帮助进行数据绘图和各种算法的原型设计。

PC在Python中实现以下角色：

--UNPI master设备

--COM端口接口

--RTLS UNPI子系统/命令集的实现

--Websocket服务器

PC在GUI Composer / Javascript中实现以下角色：

--本地主机上的Websocket客户端

--把websocket上的数据图形化并进行记录

--解析JSON对象以提取RTLS数据

--通过把WebSocket转化为UNTLS来向RTLS节点发出命令

--枚举设备

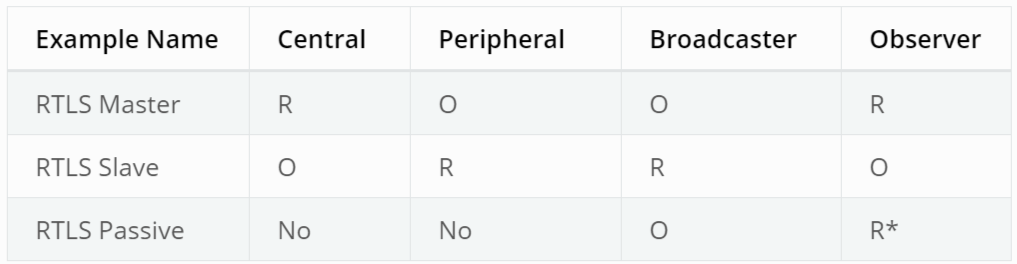
--将连接参数分发给passive设备

2.3、角色组合

注意：在以下所有表中（BLE，ToF，AoA），作为可选功能列出的功能，SDK例程并未实现，而是由用户自行选择的有效且可能的配置。例如rtls\_master可以是多角色设备，或者rtls\_passive可以用作ToF主机。

2.3.1、BLE-Stack

蓝牙spec 5.1允许设备以各种角色以及角色组合进行操作。 下表显示了每个示例的必需和可选功能。R: Required；O: Optional；No: Not supported



上面的R \*表示，尽管连接监视器可以扫描信标，而且AOA例程也要求连接监视器来跟随踪蓝牙连接。 但蓝牙规范并未正式定义“监视”角色，尽管它是rtls\_passive中的关键功能。

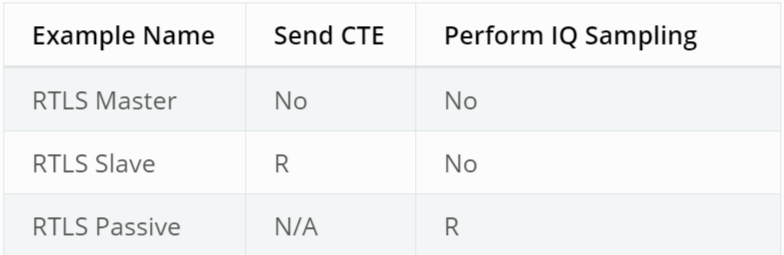
2.3.2、ToF

ToF角色可以在RTLS master和RTLS passive之间轮换。 这是一个理想的功能，因为它增加了空间分集，增加系统对多径衰落的稳健性。 下表列出了ToF的有效配置。



2.3.3、AOA

蓝牙spec 5.1为连接和无连接的AoA定义了多个角色。 SimpleLink CC13x2 / 26x2 SDK中的示例支持以下配置。



2.4、RTLS驱动

RTLS驱动程序的作用是实现RTLS模块的直接调用。更多详细信息，请查看“ ToF driver”部分和“ AoA driver”部分。

2.4.1、物理条件

在评估RTLS解决方案时，周围环境非常重要。所有无线电通信协议都容易受到多径衰落的影响，基于RTLS的系统也不例外。在评估多径衰落对结果的潜在影响时，控制环境非常重要。

建议在无线传输条件好的区域评估RTLS解决方案。包括：

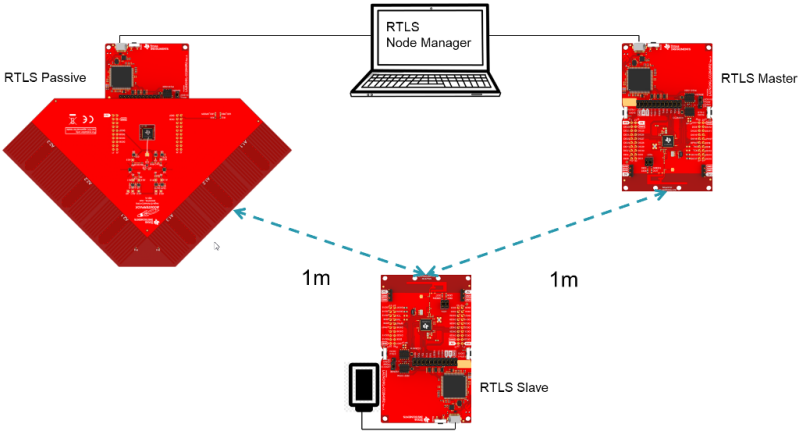
--没有大的金属或混凝土障碍物（柱子，电线杆等）的开放空间

--相对较少的干扰源（即Wi-Fi接入点等）

--距地面约1 m的用纸板制成的平台，可抬高节点。

--办公桌环境通常RF条件不太好，应避免使用。

有关评估的时候各节点的推荐布局，请参见下面的图像，这是2D图像，所有设备都放置在如图所示的平面上。实际测试时，应将每个节点放置在盒子上，以使其不直接接触地面。

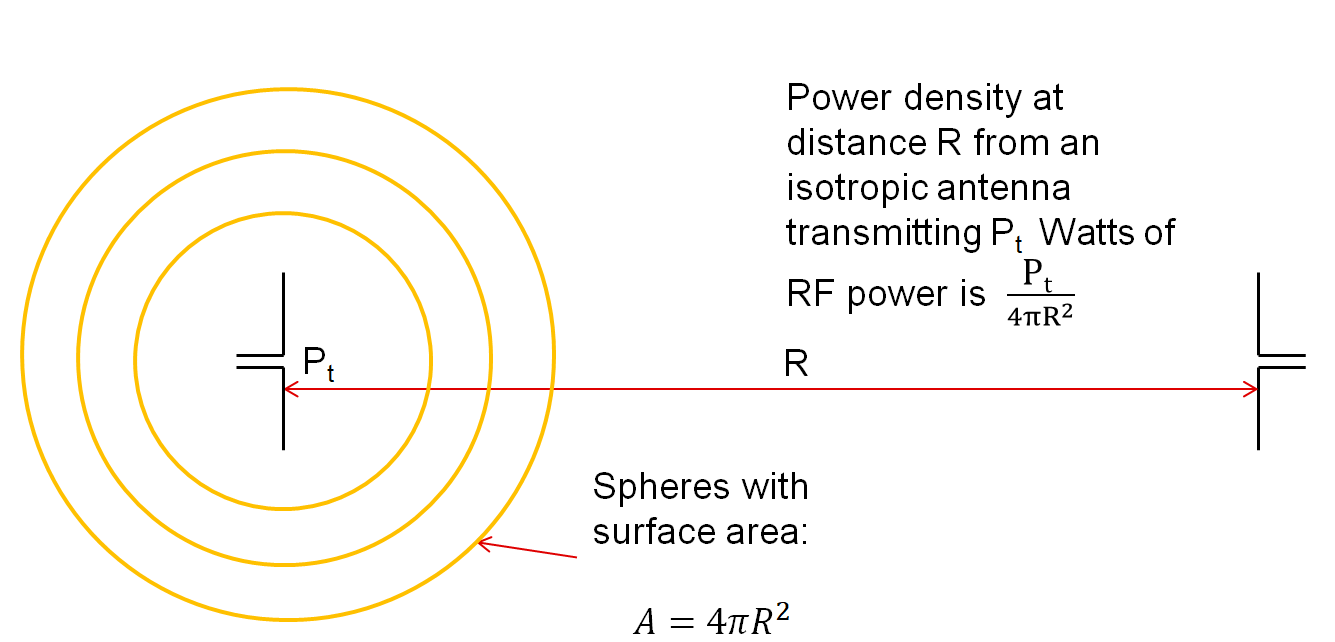


对于AOA的应用，我们还有蓝牙到达角天线设计文档【<http://www.ti.com/cn/lit/an/tida029/tida029.pdf>】，以进一步说明用户制作自己的AoA板时应关注的内容。

三、基于RSSI的定位

RSSI是指传入信号的接收信号强度指示，并且是蓝牙中最常用的三边测量定位方法。 TI蓝牙低功耗协议栈使开发人员能够收到传入的蓝牙数据包的RSSI，并可以用于RTLS算法。

RSSI定位基于Frii的传输方程式。 核心概念是接收信号强度与发射节点的距离成正比。 下图描述了RSSI如何用于估计距离。



与发射射频功率Pt瓦的各向同性天线距离R处的功率密度为Pt/（4ΠR2）；球体表面积为A=4ΠR2。

尽管基于RSSI的定位是当今RTLS系统中最常用的方法，但它也面临着需要克服的缺陷：

--准确性可能会受到反射和障碍物的影响

--无中继攻击保护

这些缺陷中的某些可以通过智能系统设计来克服。 TI的RTLS工具箱使开发人员能够监视主从设备之间的连接，并通过连接监视器从主从之间的数据包获得独立的RSSI测量。 这种方法提供了更多的数据和参考点，可用于帮助提高RTLS应用程序的分辨率。对于将来的RTLS应用，建议将RSSI与其他定位技术（例如AoA或ToF）结合使用，以帮助提高准确性和安全性。

3.1、读取RSSI

Micro BLE stack（连接监视器应用程序的基础）和完整的BLE5-Stack提供了API，以读取接收到的数据包的RSSI信息。以下各节将描述如何从接收到的数据包中提取RSSI信息。

3.2、Connection monitor【连接监视器】

连接监视器将为其跟踪的每个连接保留一组连接信息。包括上次扫描的主设备RSSI和从设备RSSI。这些字段可以在ubCM\_ConnInfo\_t结构体中找到。字段分别是rssiMaster和rssiSlave。RSSI信息在调用监视器完成回调之后（对于连接事件完成监视扫描时）有效。

rtls\_passive例程展示了一个在RTLSPassive\_monitorCompleteEvt中提取从机RSSI的示例。

3.3、BLE5协议栈

使用BLE5-Stack时，Gap\_RegisterConnEventCb（通用访问配置文件（GAP））将提供来自上一个连接事件的RSSI。它可以在主或从设备的配置中使用。连接事件回调是RTLSCtrl模块用于向PC /节点管理器报告数据的同步方法。有关如何从连接事件回调中提取RSSI的信息，请参见Gap\_ConnEventRpt\_t。

四、AOA

到达角（AoA）是一种用于查找传入的蓝牙数据包来自哪个方向，并为三角测量奠定基础的技术。使用具有明确定义属性的天线阵列，接收器将在各个天线之间快速切换，同时测量由于通向不同天线的路径长度的微小差异而导致的相移。这些路径长度差异取决于入射RF波相对于阵列中天线的方向。为了便于进行相位测量，数据包必须包含一个恒定音调（CT）的部分，其中没有调制引起的相移。

4.1、数据包格式

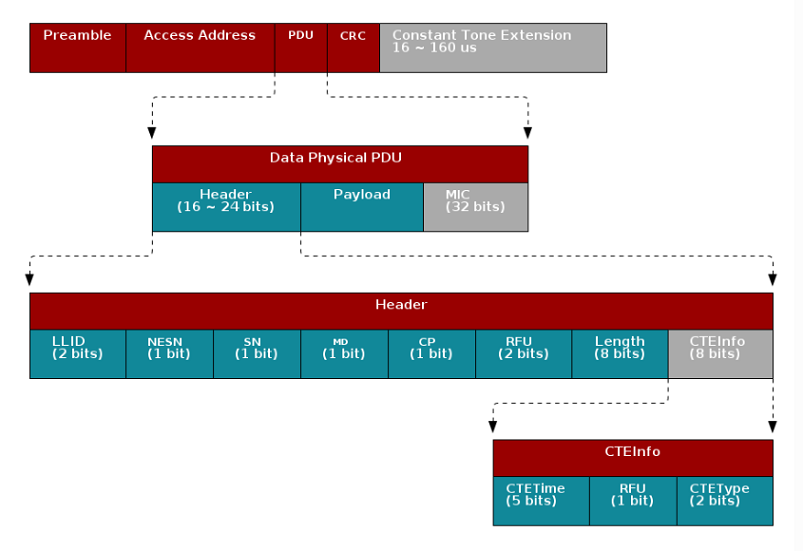
为了获得ϕ（相位）的良好估算，应消除信号中所有其他的相移。蓝牙spec 5.1引入了AoA / AoD的概念【在Direction Finding Using Bluetooth Low Energy Device一节】，并且还定义了以下几种支持发送定向数据包的状态：

--Periodic advertising; 也称为Connectionless CTE

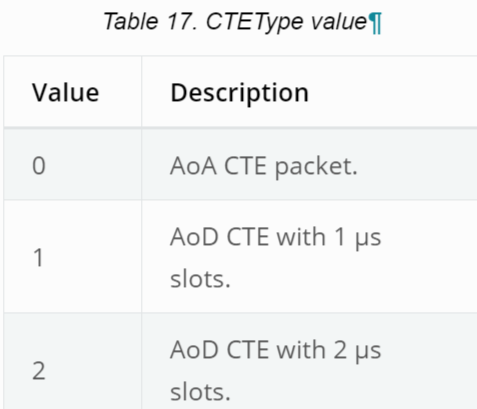
--Connection; 也称为Connection CTE

AoA / AoD和无连接/连接CTE（Constant Tone Extension）的原理是相同的，因此，我们仅关注BLE5-Stack中提供的Connection CTE AoA。

首先，让我们看一下有效负载。通过在连接数据包的末尾添加一段连续的1，可以有效地在载波频率+ 250kHz上传输单音信号。



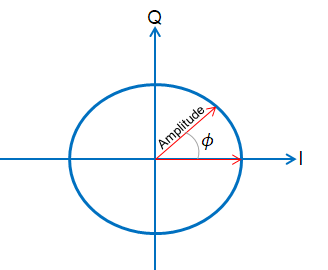
在数据包的header中，CP位【CTE Present】决定该header是否包含CTE info。CTEInfo的CTEType字段进一步指定这是哪种定位数据包类型，而CTETime字段指定CTE的持续时间。



CTETime的值应在2〜20之间，单位是8us。 这意味着当CTETime设置为20时，CTE的持续时间为8 \* 20 = 160（us）。这使接收机有时间先同步解调器，然后将来自单音调250kHz部分的I和Q样本存储到缓冲区中，最后可由AoA应用程序对该缓冲区进行后处理

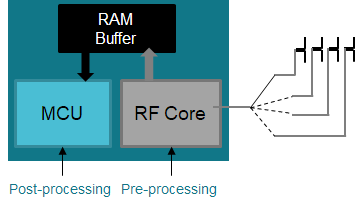
注意：

I / Q数据样本是在时间轴上的信号坐标，如下图所示。 实际上，I / Q数据只是幅度和相位数据从极坐标系到笛卡尔（X，Y）坐标系的转换，并且可以使用三角函数将极坐标正弦波信息转换为笛卡尔I / Q正弦波数据。



4.2、Integration【积分】

发射载波频率+250khz上的I/Q信号可以被RF core捕获、预处理、并且存储进buffer，而不会给主MCU造成任何负担。由于进行了预处理，因此应用程序可以确定相移，而无需先消除DC偏移或IF，从而大大简化了估算过程，并使应用程序MCU可以自由地做更多的事情。



对于rtls\_passive，I / Q采样速率目前不可配置，固定为4 MHz。 每个I / Q对 在RF RAM中占用32 bit的空间，RF RAM最多可以存储2KB（2048 byte）。因此可以计算，采样率为4MHz，每4us将有16个I / Q对，等于16 \* 4（一个I / Q对占用4个字节的空间）= 64个字节/ 4us。

这意味着即使CTE为160us长，采样率为4MHz，rtls\_passive的RF RAM也只能存储128us的I / Q数据。 除了缩短CTE长度，否则目前尚无其他解决方法。 对于rtls\_master而言，这不是问题。

注意：对于rtls\_passive，即使I和Q样本在RF Core RAM中分别占据16bit空间，也只有13位的分辨率。因此，作为有符号整数，它们的最大值和最小值为[4095，-4096]。

对于rtls\_master，I / Q 采样速率是可配置的，范围为1MHz至4MHz。每个I / Q对在RF RAM中占用32bit空间，并且RF RAM最多可以存储4KB（4096字节）。

注意：对于rtls\_master，I和Q样本分辨率是可配置的，请参阅后面章节的sample Size Setting。您可以选择13位分辨率的16位，或8位分辨率——Bluetooth Core Specification 5.1版标准。不管选择哪种分辨率，每个I / Q对将始终在RF RAM中占用32位空间。

应用层将天线切换表传递到RF Core，然后RF Core在收集I / Q样本的同时进行天线切换。在RTLS slave上，RF core确保CTE正确插入连接事件包的末端，而不会因白化滤波器失真；在passive和master中，RF Core分析数据包并在正确的时间开始捕获样本，同时同步天线切换。样本存储在RF Core RAM中，以供主MCU分析。

4.2、AOA driver

对于使用micro BLE协议栈的示例rtls\_passive，AoA驱动程序负责引脚初始化，AOA使能，数据提取和角度估计。rtls\_master中的AoA功能的实现符合蓝牙SPEC 5.1，因此，初始化，AOA使能和数据提取已移至主机来实现。但是，用户仍然可以使用RTLS控制模块来设置所需的参数。

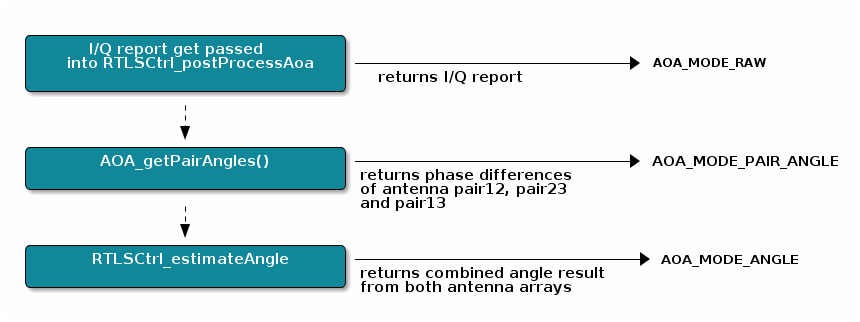
4.2.1、数据收集流程

当RF Core检测到AoA数据包时，它将开始采样音调上的I / Q信号，同时根据用户定义的周期切换天线。

对于rtls\_passive，将在RTLSPassive\_monitorCompleteEvt（）触发下一个同步事件后提取IQ样本。在同步事件处理期间，RTLSCtrl\_postProcessAoa将使能RF RAM，并使用AOA\_postProcess（）读出IQ样本；

对于rtls\_master，IQ样本将从事件类型为RTLSSRV\_CONNECTION\_CTE\_IQ\_REPORT\_EVT的RTLS Service host module发送到应用程序。

根据python设置的AoA结果模式，将调用不同的函数集，并将返回相应的数据集，如下所示：

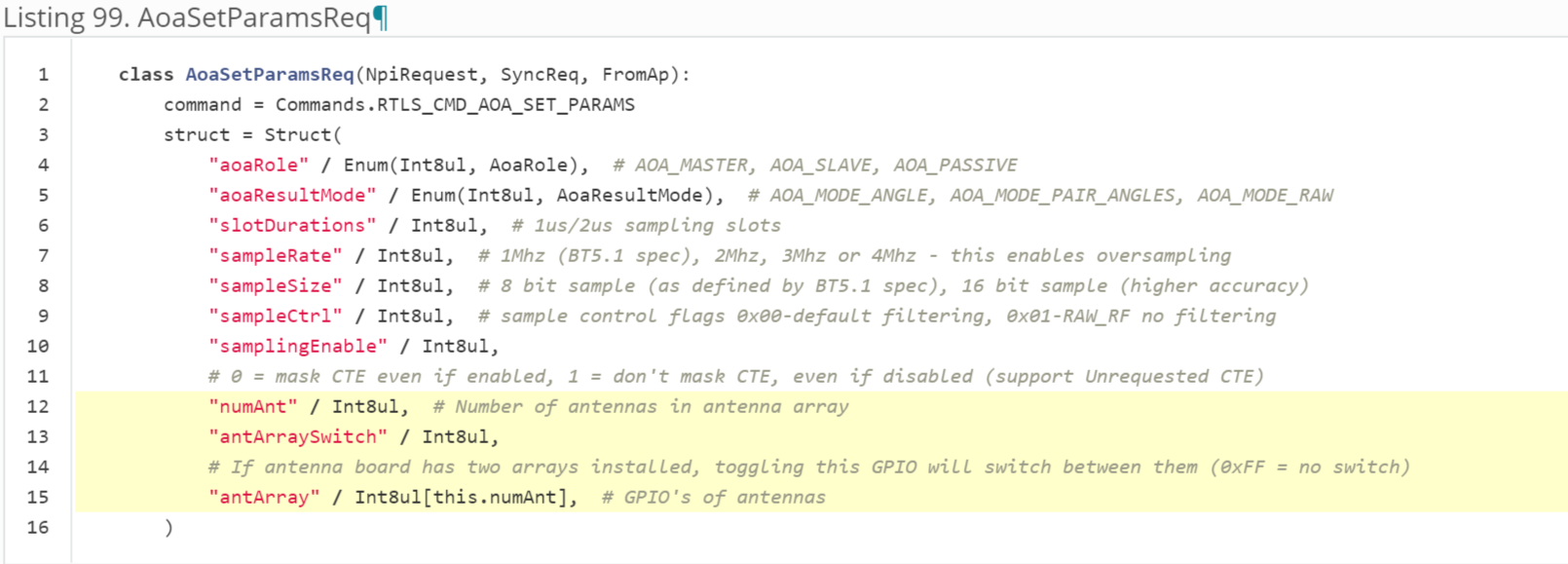


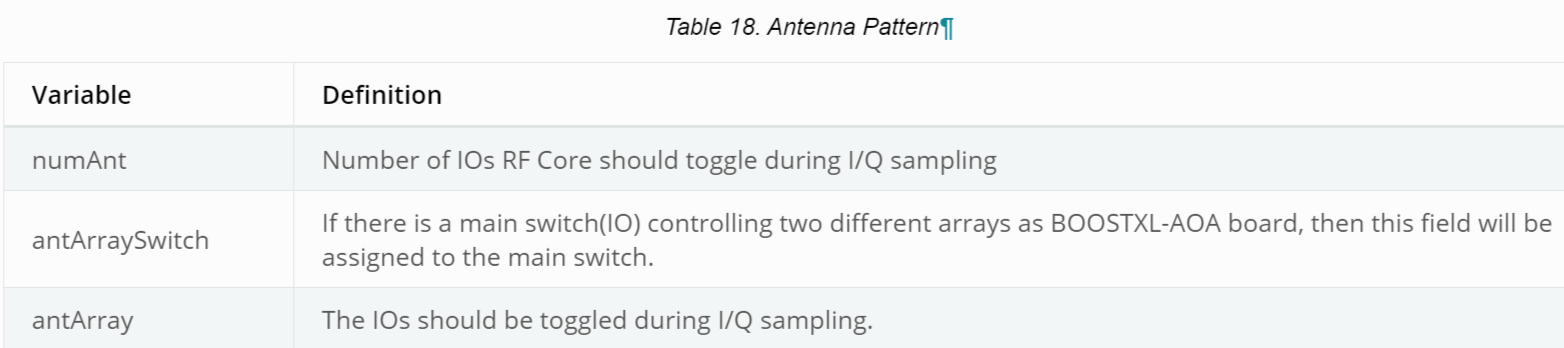
注意:对于rtls\_master，如果将sampleCtrl设置为0，则在使用AOA\_MODE\_RAW时，BLE5-Stack将从切换周期中过滤掉I / Q数据。要获取完整的I / Q数据集，你需要将sampleCtrl设置为1。 请参见4.2.6节sampleCtrl设置获取详细内容。

警告：对于rtls\_master，当sampleCtrl = 1时，不支持AOA\_MODE\_PAIR\_ANGLE和AOA\_MODE\_ANGLE。

4.2.2、天线切换

在RTLS\_CMD\_AOA\_SET\_PARAMS中设置了天线切换模式【python】。





以下是在rtls\_agent toolbox中提供的示例。

Listing 100. IO control pattern setup example[¶](http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/localization-index-cc13x2_26x2.html#id9)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18 | aoa\_params **=** {  "aoa\_run\_mode": "AOA\_MODE\_ANGLE", *## AOA\_MODE\_ANGLE, AOA\_MODE\_PAIR\_ANGLES, AOA\_MODE\_RAW*  "aoa\_cc2640r2": {  "aoa\_cte\_scan\_ovs": 4,  "aoa\_cte\_offset": 4,  "aoa\_cte\_length": 20  },  "aoa\_cc26x2": {  "aoa\_slot\_durations": 1,  "aoa\_sample\_rate": 1,  "aoa\_sample\_size": 1,  "aoa\_sampling\_control": 0,  "aoa\_sampling\_enable": 1,  "aoa\_num\_of\_ant": 3,  "aoa\_ant\_array\_switch": 27,  "aoa\_ant\_array": [28, 29, 30]  }  } |

该示例显示在I / Q采样期间切换了3个IO，并且切换顺序为IOID\_28，IOID\_29和IOID\_30。 然后，RF core将重复该模式，直到CTE结束。 之后，IO将恢复到其原始状态。

RF Core只负责切换，它不会初始化引脚状态。 应用程序应通过调用AOA\_initAntArray【rtls\_passive】或调用RTLSSrv\_initAntArray【rtls\_master】来初始化引脚状态。 主机发送RTLS\_CMD\_AOA\_SET\_PARAMS时要注意上述操作。

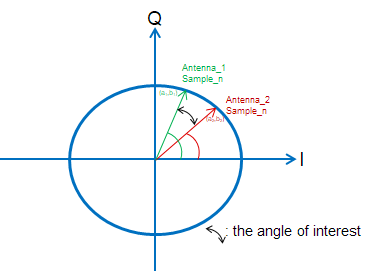
如果想要使用不同IO或模式，需要做的就是更改aoa\_set\_params的输入。

4.2.3、将I / Q数据转换为角度差

当射频波【radio frequency wave】入射到天线阵列（假设板上只有2个天线，4MHz采样率和2us 时隙（slot））并在不同时间到达不同天线时，天线之间将存在相位差。因此，我们对ant1\_sample [8至15]和ant2\_sample [8至15]之间的相位差进行提取。天线之间的切换会导致测量误差，因此在计算角度时会丢弃0至7的I / Q样本。

当使用自定义硬件和不同于TI在BOOSTXL-AoA板上的天线切换时，如果天线切换的时间短于1 us，则可能可以使用更多的I / Q样本。“用于角度计算的有效I / Q样本“中说明了确定可以使用多少个I / Q样本的方法。

可以将I / Q数据呈现为实数I和虚数Q（相差90度）的X-Y域。如前所述，对于每个250 kHz信号周期，我们采样16个I和Q数据。如果没有差异，则意味着I / Q数据相同，因此ant\_1 sample1和ant\_2 sample1之间的相位将相同。



如下是将I / Q数据2维呈现的代码，然后将I / Q对传递到函数AOA\_AngleComplexProductComp（）中以获得最终角度。

Listing 101. Get the phase differences.[¶](http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/localization-index-cc13x2_26x2.html#id10)

**for** (**uint16\_t** r **=** 1; r **<** (numIqSamples **-** (AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate))**/**(numAnt**\***sampleRate) ; **++**r) *// Sample Slot*

{

**for** (**uint16\_t** i **=** 0; i **<** sampleRate; **++**i) *// Sample inside Sample Slot*

{

*// Loop through antenna pairs and calculate phase difference*

**for** (**uint8\_t** pair **=** 0; pair **<** numPairs; **++**pair)

{

**const** AoA\_AntennaPair **\***p **=** **&**gAoaReport.antConfig**->**pairs[pair];

**uint8\_t** a **=** p**->**a; *// First antenna in pair*

**uint8\_t** b **=** p**->**b; *// Second antenna in pair*

*// Calculate the phase drift across one antenna repetition (X \* complex conjugate (Y))*

**int16\_t** Paa\_rel **=** AOA\_AngleComplexProductComp(pI[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** r**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i],

pQ[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** r**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i],

pI[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** (r**-**1)**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i],

pQ[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** (r**-**1)**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i]);

*// Calculate phase difference between antenna a vs. antenna b*

**int16\_t** Pab\_rel **=** AOA\_AngleComplexProductComp(pI[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** r**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i],

pQ[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** r**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i],

pI[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** r**\***numAnt**\***sampleRate **+** b**\***sampleRate **+** i],

pQ[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** r**\***numAnt**\***sampleRate **+** b**\***sampleRate **+** i]);

*// Add to averages*

*// v-- Correct for angle drift / ADC sampling frequency error*

antenna\_versus\_avg[a][b] **+=** Pab\_rel **+** ((Paa\_rel **\*** abs(a**-**b)) **/** numAnt);

antenna\_versus\_cnt[a][b] **++**;

}

}

}

**int32\_t** AOA\_AngleComplexProductComp(**int32\_t** Xre, **int32\_t** Xim, **int32\_t** Yre, **int32\_t** Yim)

{

**int32\_t** Zre, Zim;

**int16\_t** angle;

*// X\*conj(Y)*

Zre **=** Xre**\***Yre **+** Xim**\***Yim;

Zim **=** Xim**\***Yre **-** Xre**\***Yim;

*// Angle. The angle is returned in 256/2\*pi format [-128,127] values*

angle **=** AOA\_iatan2sc((**int32\_t**) Zim, (**int32\_t**) Zre);

**return** (angle **\*** angleconst);

}

需要强调的一点是，实际上250kHz可能并不准确（例如，可能为255kHz或245kHZ），因此，ant\_1 sample\_n与ant\_1 sample\_（n + 16 \* 1）之间存在微小的相位差。 所以代码里应用了运行时间补偿：

Listing 102. Compensation method.[¶](http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/localization-index-cc13x2_26x2.html" \l "id11" \o "Permalink to this code)

*// Calculate the phase drift across one antenna repetition (X \* complex conjugate (Y))*

**int16\_t** Paa\_rel **=** AOA\_AngleComplexProductComp(pI[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** r**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i],

pQ[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** r**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i],

pI[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** (r**-**1)**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i],

pQ[AOA\_OFFSET\_FIRST\_VALID\_SAMPLE**\***sampleRate **+** (r**-**1)**\***numAnt**\***sampleRate **+** a**\***sampleRate **+** i]);

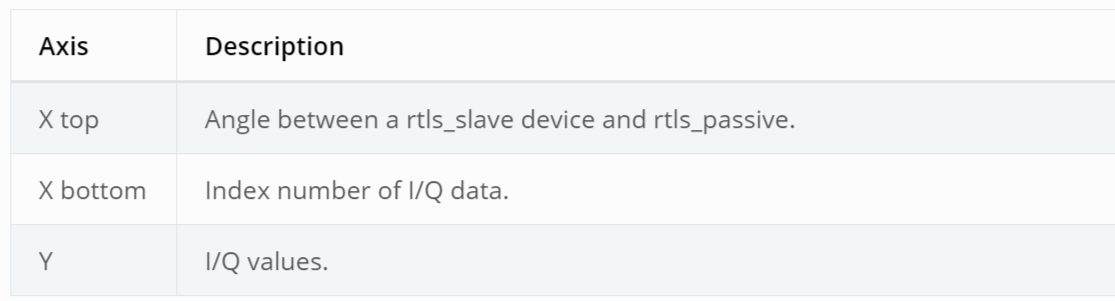
由于非完美的250kHz tone，会累积相位差。 假设每个周期都会有45度的延迟。 然后，在比较ant\_1 sample\_n和ant\_1 sample\_（n + 16 \* 1）时，合计相位差为90度。 但是每个周期之间的实际相位差只有45度。 因此，在这种情况下，必须将计算出的相位差除以所用天线的数量。

最终的相位差：

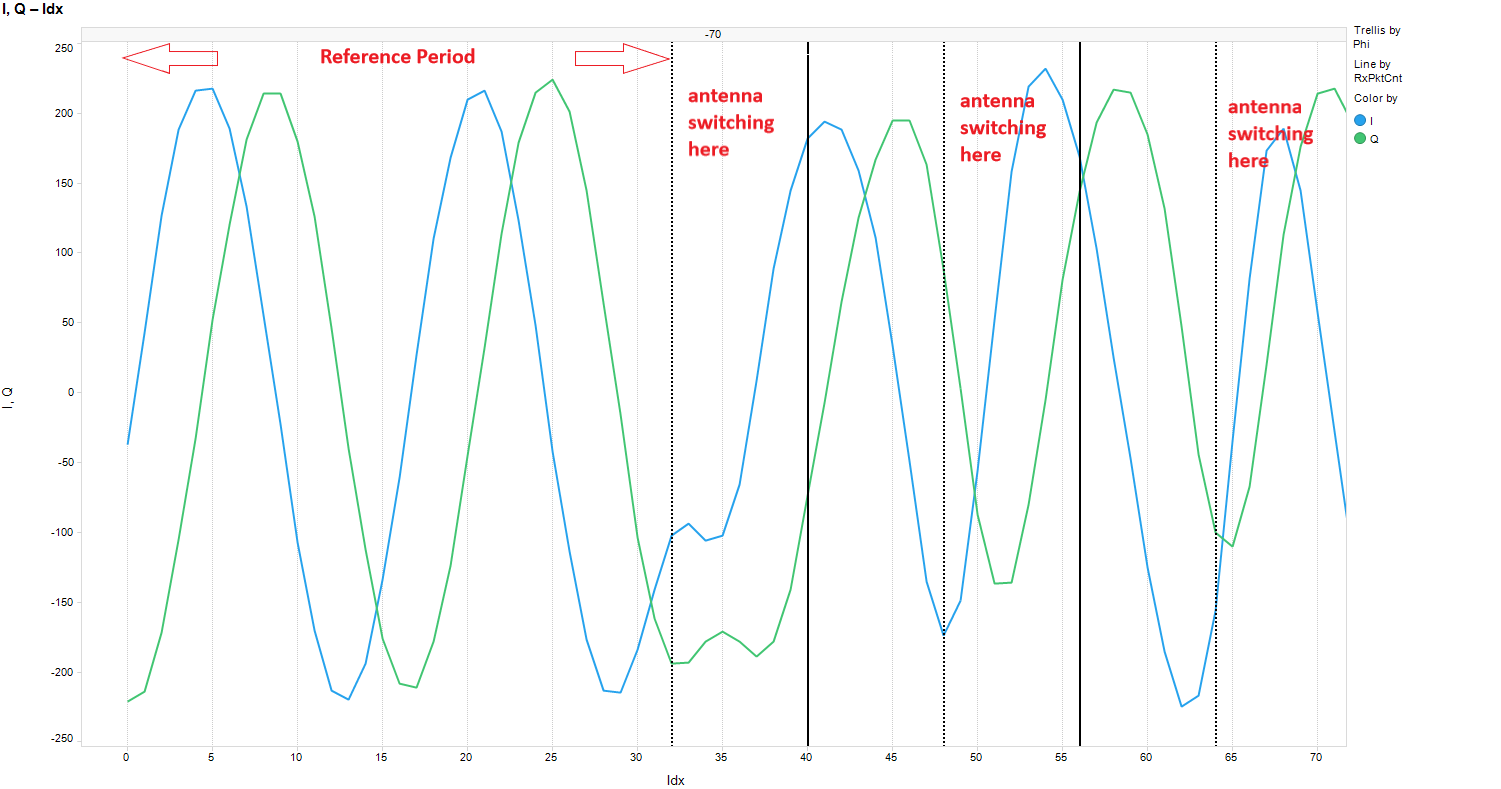
|  |  |
| --- | --- |
| 1 | antenna\_versus\_avg[a][b] **+=** Pab\_rel **+** ((Paa\_rel **\*** abs(a**-**b)) **/** numAnt); |

4.2.4、用于角度计算的有效I / Q样本

为了确定要使用的I / Q样本。 最简单的分辨方法是绘制所有I / Q样本。下图显示了rtls\_passive/ rtls\_master/rtls\_slave例程使用4MHz采样率和2us时隙的收集的I / Q样本。



前32个（数值0〜31）样本是在没有天线切换的参考周期内获取的。 因此，I / Q图看起来像正弦波。之后，在数值32处，您可以看到切换发生时I / Q样本的不连续。



注意：当存在相位差时，很容易看到相位不连续。 因此，在收集I / Q数据之前，请确保rtls\_passive和rtls\_slave之间的角度不为0。

4.2.5、角度补偿

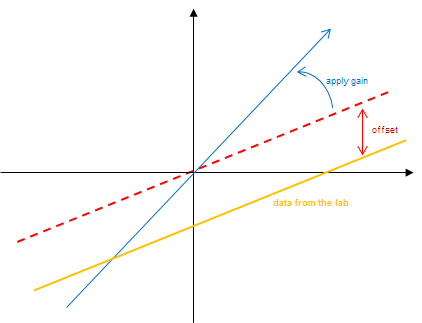
在AoA\_getPairAngles（）下，获取基于I和Q数据的相位差。 之后，添加角度补偿。 请参见下面的代码。 这是因为角度估计受天线对和频率影响。 值p-> gain，p-> offset，channelOffset\_A1和channelOffset\_A2基于实验室测量值。 不同的天线板设计和频率将有不同的p-> gain，p-> offset，channelOffset\_A1和channelOffset\_A2。

可以在AOA.c AoA\_getPairAngles（）中找到以下代码，这是天线对补偿。

Listing 104. Antenna pair compensation[¶](http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/localization-index-cc13x2_26x2.html#id14)

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | *// Write back result for antenna pairs*  **for** (**int** pair **=** 0; pair **<** numPairs; **++**pair)  {  **const** AoA\_AntennaPair **\***p **=** **&**gAoaReport.antConfig**->**pairs[pair];  gAoaReport.antResult**->**pairAngle[pair] **=** (**int**)((p**->**sign **\*** antenna\_versus\_avg[p**->**a][p**->**b] **+** p**->**offset) **\*** p**->**gain);  } |

从下图可以看到，偏移量是为了确保在0度时接收到的数据在计算后得出0度，然后更改斜率以使其更适合所有其他角度。



天线阵列1的补偿值可在ant\_array1\_config\_boostxl\_rev1v1.c的 AoA\_AntennaPair pair\_A1[]中找到。

Listing 105. Values used for compensation[¶](http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/localization-index-cc13x2_26x2.html#id15)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | AoA\_AntennaPair pair\_A1[] **=**  {  {*// v12*  .a **=** 0, *// First antenna in pair*  .b **=** 1, *// Second antenna in pair*  .sign **=** 1, *// Sign for the result*  .offset **=** **-**5, *// Measurement offset compensation*  .gain **=** 0.95, *// Measurement gain compensation*  },  {*// v23*  .a **=** 1,  .b **=** 2,  .sign **=** 1,  .offset **=** **-**20,  .gain **=** 0.9,  },  {*// v13*  .a **=** 0,  .b **=** 2,  .sign **=** 1,  .offset **=** **-**20,  .gain **=** 0.50,  },  }; |

除了天线补偿，我们还增加了频率补偿。 对于天线阵列1，可在ant\_array1\_config\_boostxl\_rev1v1.c的 int8\_t channelOffset\_A1 [40]中找到用于频率补偿的值。

Listing 106. Values used for compensation[¶](http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/localization-index-cc13x2_26x2.html#id16)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41 | **int8\_t** channelOffset\_A1[40] **=** {2, *// Channel 0*  2, *// Channel 1*  1, *// Channel 2*  1, *// Channel 3*  1, *// Channel 4*  1, *// Channel 5*  1, *// Channel 6*  1, *// Channel 7*  0, *// Channel 8*  0, *// Channel 9*  0, *// Channel 10*  3, *// Channel 11*  3, *// Channel 12*  2, *// Channel 13*  3, *// Channel 14*  3, *// Channel 15*  3, *// Channel 16*  3, *// Channel 17*  3, *// Channel 17*  3, *// Channel 18*  3, *// Channel 20*  3, *// Channel 21*  2, *// Channel 22*  3, *// Channel 23*  3, *// Channel 24*  3, *// Channel 25*  3, *// Channel 26*  3, *// Channel 27*  3, *// Channel 28*  2, *// Channel 29*  2, *// Channel 30*  2, *// Channel 31*  2, *// Channel 32*  2, *// Channel 33*  2, *// Channel 34*  2, *// Channel 35*  1, *// Channel 36*  0, *// Channel 37*  0, *// Channel 38*  0, *// Channel 39*  }; |

4.2.6、AOA功能概述

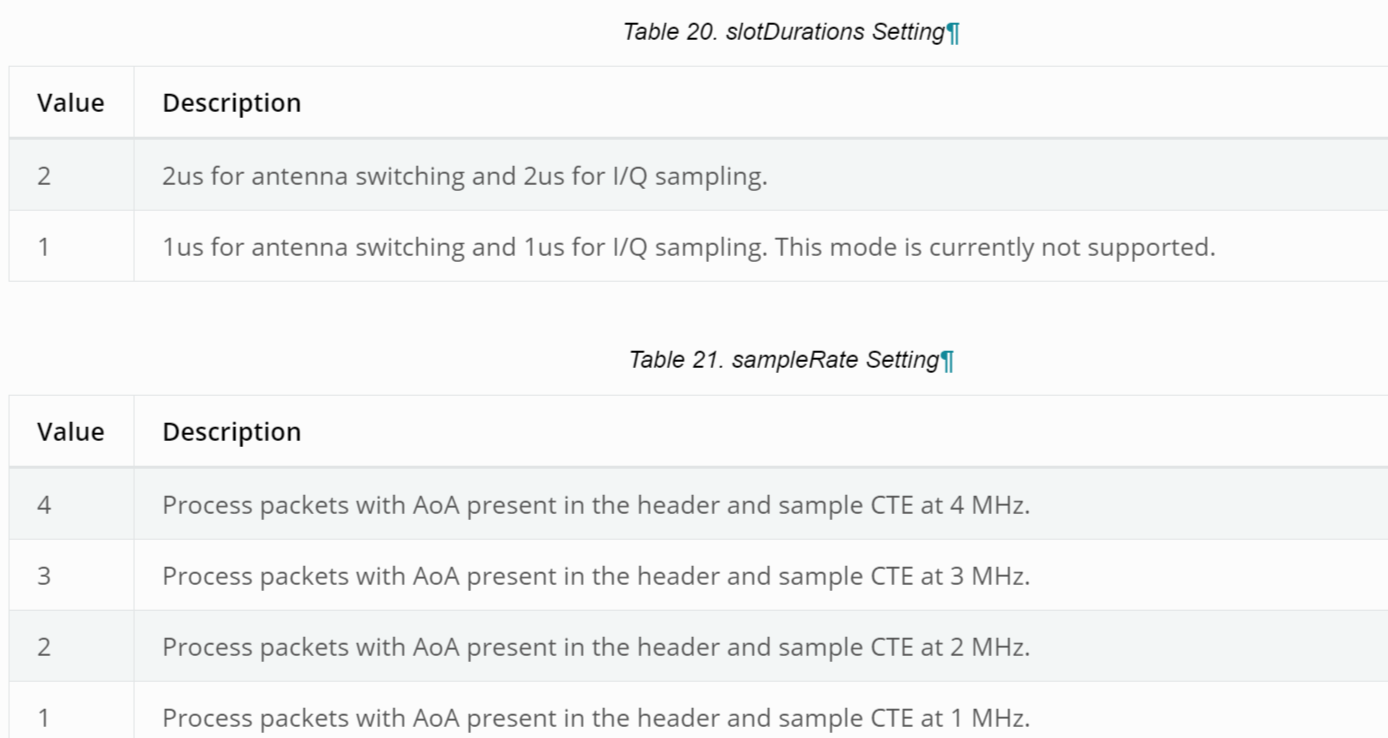
如下是AOA使用者最需要关注的几个参数：

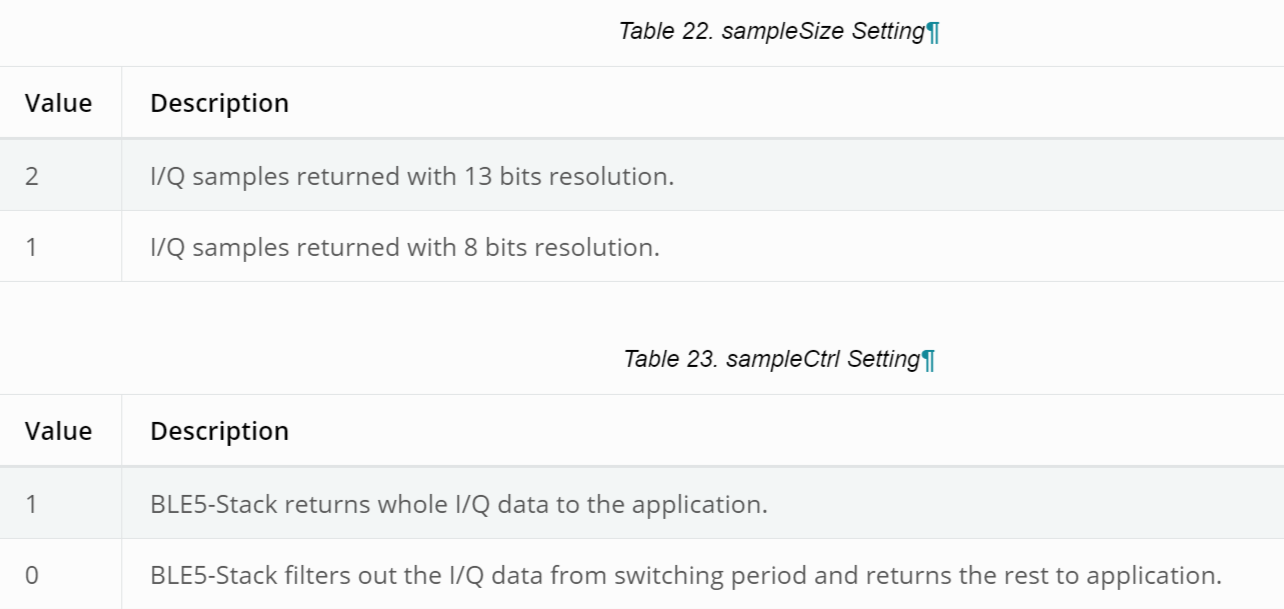
Python

1. aoa\_set\_params:

Listing 107. AoA Set Parameter[¶](http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/localization-index-cc13x2_26x2.html#id17)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16 | **class** **AoaSetParamsReq**(NpiRequest, SyncReq, FromAp):  command **=** Commands**.**RTLS\_CMD\_AOA\_SET\_PARAMS  struct **=** Struct(  "aoaRole" **/** Enum(Int8ul, AoaRole), *# AOA\_MASTER, AOA\_SLAVE, AOA\_PASSIVE*  "aoaResultMode" **/** Enum(Int8ul, AoaResultMode), *# AOA\_MODE\_ANGLE, AOA\_MODE\_PAIR\_ANGLES, AOA\_MODE\_RAW*  "slotDurations" **/** Int8ul, *# 1us/2us sampling slots*  "sampleRate" **/** Int8ul, *# 1Mhz (BT5.1 spec), 2Mhz, 3Mhz or 4Mhz - this enables oversampling*  "sampleSize" **/** Int8ul, *# 8 bit sample (as defined by BT5.1 spec), 16 bit sample (higher accuracy)*  "sampleCtrl" **/** Int8ul, *# sample control flags 0x00-default filtering, 0x01-RAW\_RF no filtering*  "samplingEnable" **/** Int8ul,  *# 0 = mask CTE even if enabled, 1 = don't mask CTE, even if disabled (support Unrequested CTE)*  "numAnt" **/** Int8ul, *# Number of antennas in antenna array*  "antArraySwitch" **/** Int8ul,  *# If antenna board has two arrays installed, toggling this GPIO will switch between them (0xFF = no switch)*  "antArray" **/** Int8ul[this**.**numAnt], *# GPIO's of antennas*  ) |





注意：rtls\_passive仅支持以下配置：slotDuration = 2，sampleRate = 4，sampleSize = 2。

4.2.7、RTLS\_Passive

本节涵盖的功能全部在AOA.c中。

1. AOA\_init：此函数负责IO初始化。
2. AOA应用概述

TI提供的例程中，rtls\_master和rtls\_slave将进行BLE连接。 建立连接后，rtls\_master将通过UART将连接信息发送到PC，然后节点管理器会将这些信息传递给rtls\_passive，后者随后可以跟踪连接。

接下来，节点管理器为主节点和被动节点设置AoA参数，然后主节点将发送无线发送数据包到从节点以设置CTETime。

之后，rtls\_master将通过空中向rtls\_slave和rtls\_passive发送一个启动AoA请求，然后rtls\_slave将在每个连接数据包的末尾附加CTE。

最后，rtls\_passive和rtls\_master可以进行I / Q采样并根据ConnectionCTE数据包计算角度。

下面的序列图说明了out of box例程工作的整个过程。

@startuml
participant rtls_passive as passive
participant "Node_Manager \n(Host MCU)" as manager
participant rtls_master as master
participant rtls_slave as slave

manager -> master: RTLS_CMD_IDENTIFY_Req
master -> manager: RTLS_CMD_IDENTIFY_Rsp
manager -> passive: RTLS_CMD_IDENTIFY_Req
passive -> manager: RTLS_CMD_IDENTIFY_Rsp

manager -> master: RTLS_CMD_SCAN_Req

activate master
master -> master: Start scanning \nfor rtls_slave

group rtls_slave not found

    manager -> master: RTLS_CMD_SCAN_STOP
    deactivate master
    manager -> master: RTLS_CMD_SCAN_Req \nrestart scanning

    activate master
    master -> master: Start scanning \nfor rtls_slave
    manager -> master: RTLS_CMD_SCAN_STOP
    deactivate master
    ...
    ... Repeat until device found ...
    ...
end


group rtls_slave found
    master -> manager: RTLS_CMD_SCAN_AsyncReq
    manager -> master: RTLS_CMD_SCAN_STOP
    manager -> master: RTLS_CMD_CONNECT
    master --> slave: Connection request
    == After connection has established ==
    master -> manager: RTLS_CMD_CONN_PARAMS
    manager -> passive: RTLS_CMD_CONN_PARAMS \npass on the connection info \nfor passive to track connection

    group AoA
        manager -> master: RTLS_CMD_AOA_SET_PARAMS
        master --> slave: RTLS_REMOTE_CMD_AOA_SET_PARAMS
        manager -> passive: RTLS_CMD_AOA_SET_PARAMS
        manager -> master: RTLS_CMD_AOA_ENABLE
        master --> slave: Enable sending \npackets with CTE
        manager -> passive: RTLS_CMD_AOA_ENABLE

        activate passive
        passive -> passive: Start following connection \nand waiting for connectionCTE packets
        master--> slave: Empty packet
        slave --> master: ConnectionCTE packet
        master --> master: Received ConnectionCTE packet \nand then calculate angle \nbetween slave and master
        passive -> passive: Received ConnectionCTE packet \nand then calculate angle \nbetween slave and passive
        master -> manager: RTLS_CMD_AOA_RESULT_ANGLE
        passive -> manager: RTLS_CMD_AOA_RESULT_ANGLE

        ...
        ... master and slave stay in connection ...
        ... master captures I/Q samples and calculates angle ...
        ... passive tracks connection, captures I/Q samples and calculates angle ...
        ...
    end
end

@enduml

五、Time of flight【TOF】

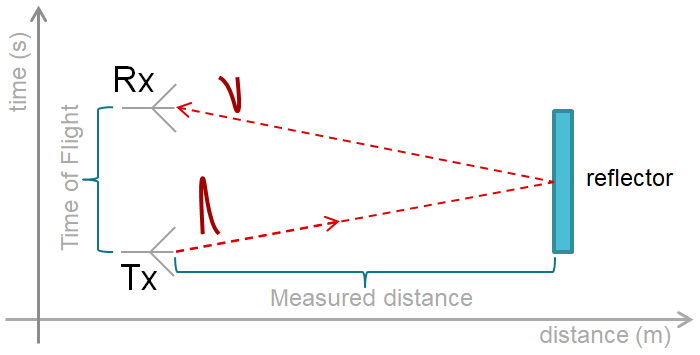
飞行时间（ToF）是TI的一项专有技术，用于通过测量RF数据包交换的往返延迟来划定安全范围界限。这是通过Master-Slave-Passive的配置实现的，在该配置中，master发送challenge，slave在固定的周转时间后返回响应。 然后，主机可以通过测量challenge发送和响应之间的时间差，减去（已知）固定的周转时间，来计算往返延迟。 passive不参与空中交换的过程，但也会测量观察到的飞行时间。

由于无线电采样时钟频率的低速特性，在光速条件下进行评估时，每个单独的测量仅提供非常粗略的结果。 但是，通过执行许多测量（通常在几毫秒内完成几百次测量），可以获得具有更好精度的平均结果。

5.1、操作理论

几乎没有什么东西比光速快，该速度是已知的并且恒定为c。 电磁波以光速传播，因此RF数据包以光速传播。

由于速度是恒定的，这意味着RF波传播所需的时间与距离成正比。 为了找到到物体的距离，我们可以记录传输某物时的时间戳，并将其与接收到反射时的时间戳进行比较，除以二并乘以c。 这也是雷达的操作原理。



如上所示，如果发送和接收之间的时间为t，则距离d为ct / 2。

与雷达相反，TI ToF中的反射器被认为是活动的，因为它不是简单地地反射回信号，而必须主动发出“反射”信息。进行TOF测量存在两个主要挑战：

1、反射器在收到PING和发送PONG之间使用的时间会影响所测距离。

2、光每3.3 ns就行进一米，这意味着测量飞行时间的时钟的tick速度必须至少为303 MHz，才能获得1 m的空间分辨率。

TI ToF解决方案是，通过在slave/反射器设备上实现确定的周转时间来克服第一个挑战。

通过采用统计方法测量距离可以克服第二个挑战。用于测量飞行时间的无线电计时器的频率为8 MHz，这意味着时间分辨率为125 ns。可以通过对单个数据包测量值进行过采样来提高最终测量的精度。这是因为样本中存在抖动，并且此抖动具有正态分布。为了获得更高的精度，将在多个设备上的许多PING / PONG交换中测量飞行时间。

5.2、术语解释

PING：发送启动ToF测量的数据包

PONG：对PING的响应数据包

Run/burst：用于测量距离的一系列PING / PONG交换

Tick：用于测量ToF的RF core时钟的单个周期

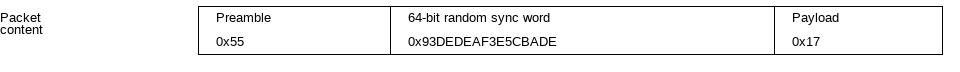
Sample：单个PING / PONG交换测量到的RSSI，tick和channel信息

回顾一下，ToF burst是PING / PONG交换的集合。 每次PING / PONG交换都会产生一个ToF Sample，该Sample是RSSI，时间戳（以tick表示）和频率信息的集合。burst可以包含可配置数量的PING / PONG交换，并且可以在可配置的频率列表上操作。 在burst结束时，所有tick信息被平均计算以得出估计的距离。 来自多个设备的多个burst的信息可以组合在一起，形成给定置信区间【confidence interval】内的距离测量。

*PS：置信区间是指由样本统计量所构造的总体参数的估计区间。 在统计学中，一个概率样本的置信区间（Confidence interval）是对这个样本的某个总体参数的区间估计。 置信区间展现的是这个参数的真实值有一定概率落在测量结果的周围的程度，其给出的是被测量参数的测量值的可信程度，即前面所要求的“一个概率”。*

5.3、数据包类型

通过一个调制格式为2Mhz，deviation为500khz的分段线性整形器进行传输，数据包内容基于标准BLE数据包，并进行了一些修改以满足ToF用例。



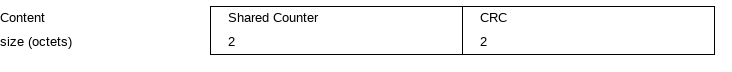
数据包包含一个随机的预共享同步字，该字对于空中发送的每个帧都是唯一的，并且BLE连接的任一侧节点都将在其RX周期内监听。同步字是RF电路检测这个包是不是发给本设备的，以及生成用于测量的时间戳。

执行TOF时，RF core将发送两种类型的数据包：

A、测量数据包：用于执行安全距离测量

B、广播数据包：这些数据包用于同步passive节点，而不是距离测量

测量数据包将使用通过安全模块生成的随机同步字，不使用这些数据包的有效载荷字段。 广播数据包使用恒定的同步字，并利用有效负载字段共享同步信息。 广播数据包的目的是同步passive节点。广播数据包的有效负载如下所示：



注意：ToF的安全性取决于良好的同步字生成算法。 同步字不可轻易被攻击者猜中，并且不可重复。 简而言之，它们的生成方式应使受信任的ToF节点能够确定下一个同步字，但攻击者则不能。由于广播数据包不用于距离测量，因此它们可以安全地使用恒定的同步字。

5.4、TOF角色

ToF方案有三种角色。

master：通过发送PING来启动ToF

slave：监听PING，并回应PONG

passive：同时收听PING和PONG

master和passive将使用RF core中的计时器来测量飞行时间。 运行ToF并不严格要求passive角色，但它将通过在测量中添加其他样本来增加测量结果的稳健性并提供空间多样性。

由于ToF是一种统计量度，因此在给定量度中，样本越多，置信度越高。passive节点增加了空间分布，并可以减少ToF测量中的反射和多径衰落影响。passive节点数量的没有要求【可以0个，也可以多个】，但是必须至少有一个master和一个slave。

passive和master之间的距离必须已知，以便可以校准该距离。

5.5、TOF协议

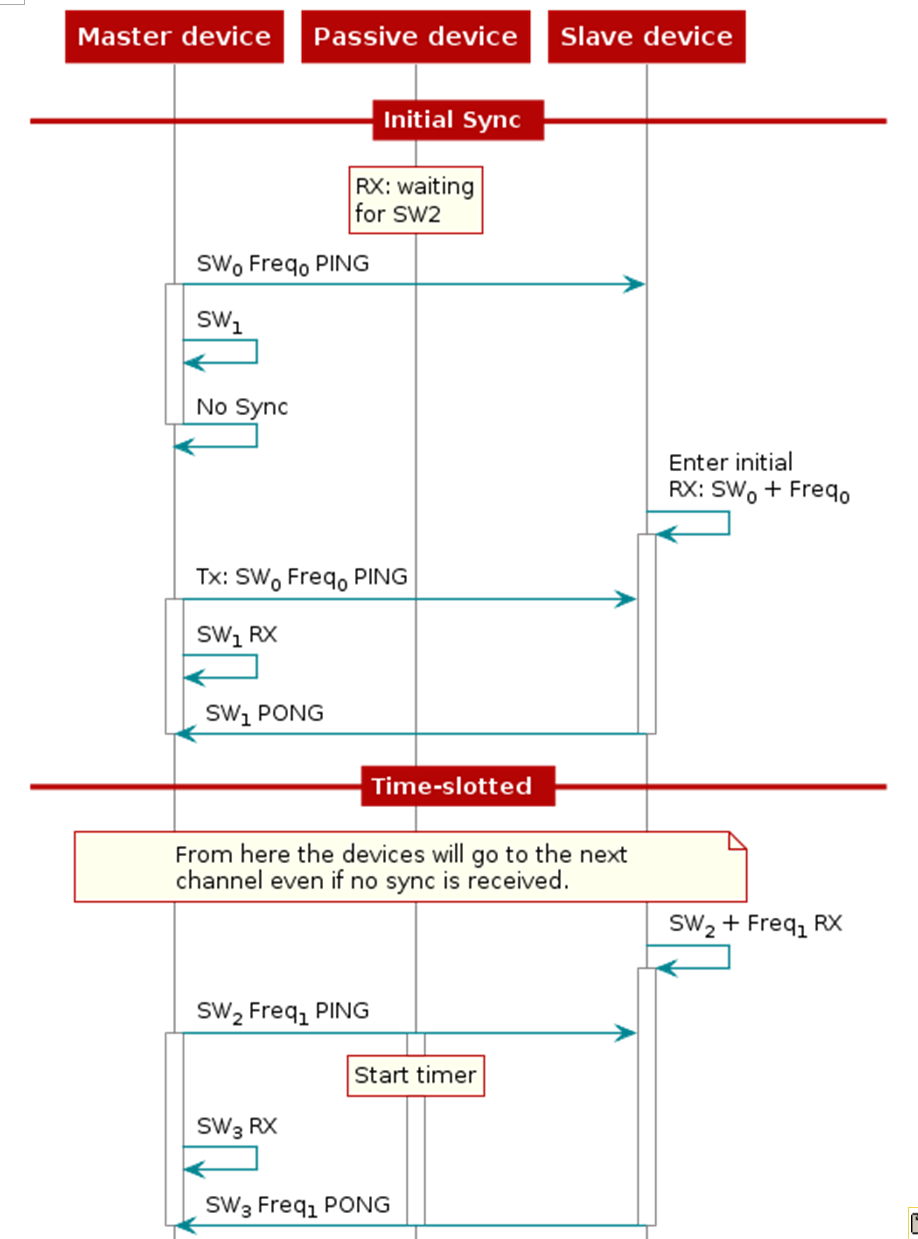
Master设备和slave设备将在应用程序提供的一系列频率上进行监听和发送。同时，passive设备将在主机和从机之间实现同步后，监听所有主机和从机之间传递的数据包。

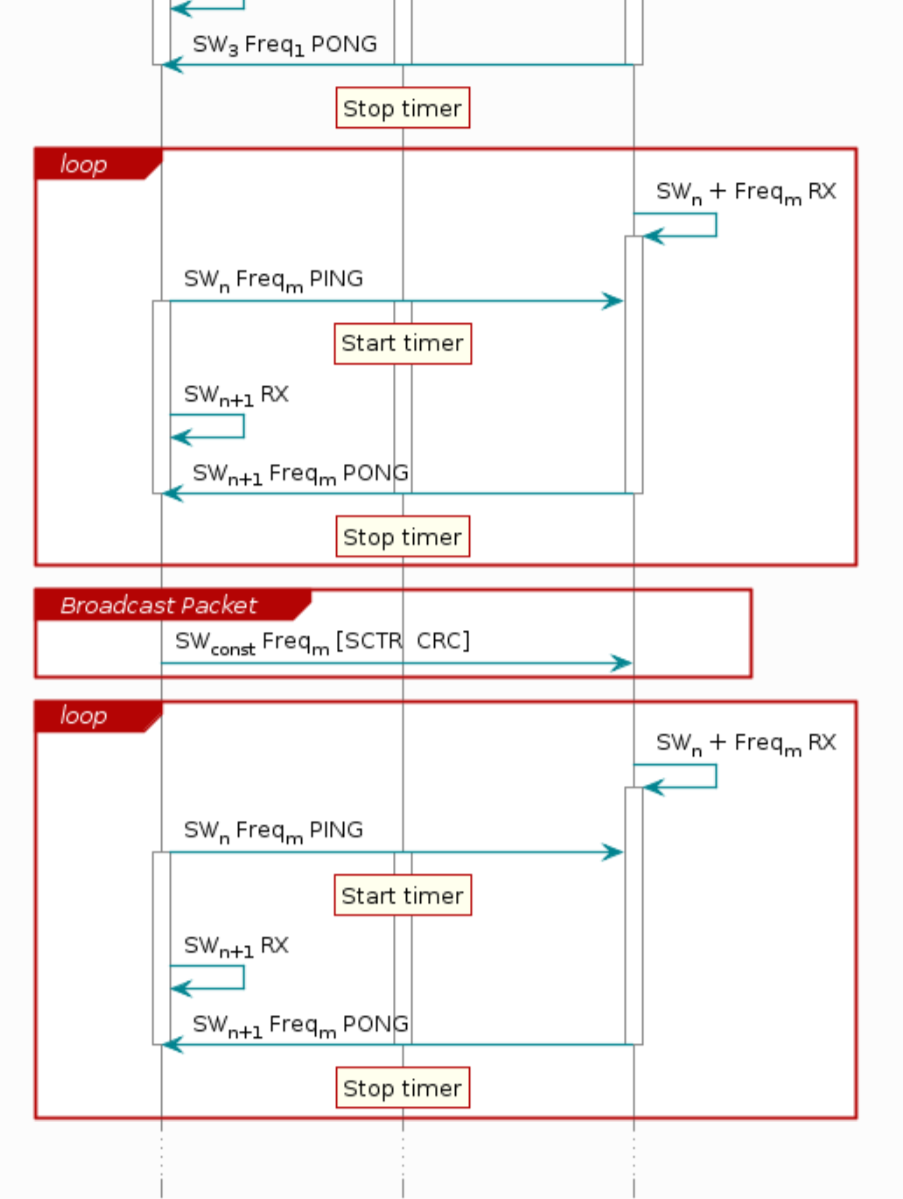
从设备在第一个频率上处于接收模式，并且监听第一个同步字SW0，这就是PING。如果从机在初始侦听周期中接收到匹配的数据包，它将发送一个PONG并开始遵循一个时隙方案，在该方案中它将更改频率和同步字。

主机将发送第一个包，其中包含第一个同步字（SW0）和一些应用程序定义的有效负载。然后它将立即进入接收模式，并等待从机回复发送阵列中的第二个同步字（SW1）。如果主机接收到匹配的PONG数据包，它将遵循相同的时隙和跳频方案。

passive设备将以RX模式启动，等待SW2。一旦passive设备接收到SW2，它也遵循时隙和跳频方案进行切换。passive在接收到SW2n时启动radio timer，然后接收到SW2n + 1时停止计时器。

TOF master会定期发送广播数据包。与测量数据包不同，它们使用恒定同步字，并在有效负载中嵌入一个共享计数器（SCTR--Shared counter）。广播数据包的目的是同步passive设备。可以通过更改TOF.c中的fBroadcast.period来配置广播数据包的周期。注意：广播数据包不用于距离测量。



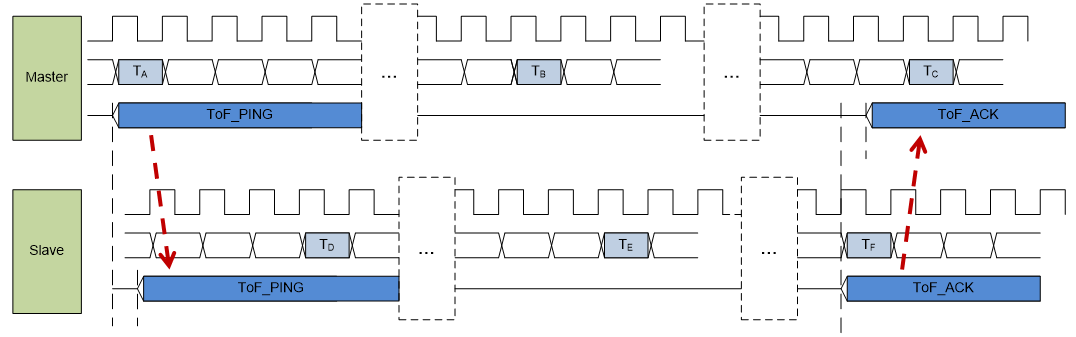


如果仔细观察数据包的时间安排，我们可以将飞行时间用下面两条垂直虚线之间的距离来表示。你会看到两个设备的内部时钟并未同步，如每个设备的第一行所示。并且还可以看到ToF测量的三个阶段：

A、主机发送PING，从机接收

B、设备将RF角色从TX切换为RX，将RX切换为TX

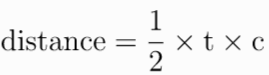
C、已发送响应PONG



具体解释：TA - Master sends challenge PING,TB - Master TX/RX switch, TC - Master detects SW correlation, TD - Slave detects SW correlation, TE - Slave TX/RX switch, TF - Slave sends response PONG.

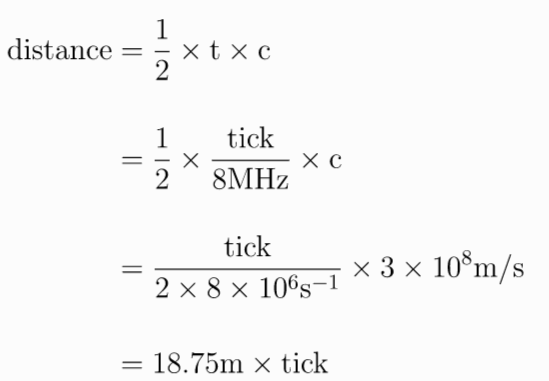
5.6、分辨率

ToF的距离以radio timer的tick来衡量，该计时器以8MHz运行。 必须将这些tick值转换为米，以估计到slave的距离。 距离方程用于执行转换。 本节旨在说明tick转化到米的逐步推导。



由于ToF数据包是电磁波，因此其传播速率固定为c，即光传播一米所需的时间。 结果必须除以2，因为测量的时间是数据包的往返行程时间，即实际距离的2倍。

为了将tick值转换为时间，必须将它除以radio timer的频率。 将其代入上面的距离方程式可得到用于将tick转换为米的比例因子。



ToF数据包每个tick将传输18.75米。 在软件中，此常量由TICK\_TO\_METER定义。 要将tick值转换为米，只需乘以TICK\_TO\_METER。

5.7、精度

有两种物理现象值得注意：

A、设备内部时钟/振荡器中的抖动和相位噪声

B、执行ToF的设备（例如主设备和从设备）之间的同步时钟

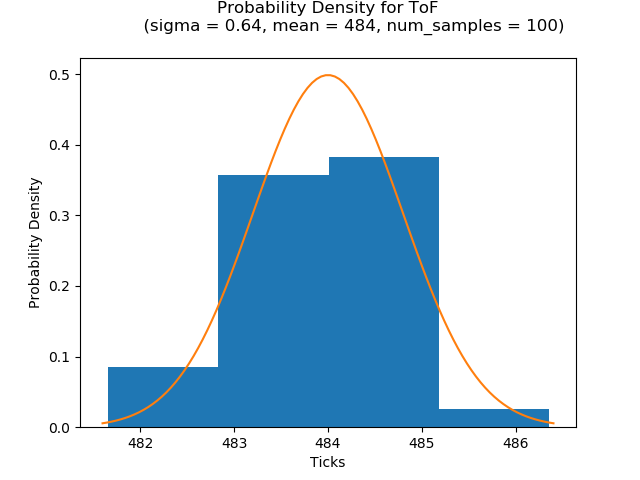
CC13x2或CC26x2的RF core内部有一个称为相关器【correlator】的组件。 相关器负责测量两个同步字之间的相关值。 当RF内核运行ToF命令时，相关器负责启动/停止测量ToF tick的计时器。

ToF协议中的各种设备没有同步其时钟的方法，因此，发送器和接收器中的符号时钟之间存在一些固有的随机偏移。 这意味着在多个ToF采样中，相关器报告同步的时间将略有漂移。

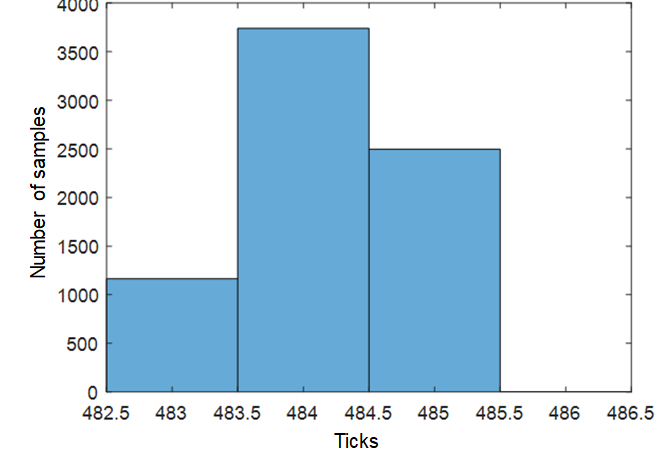
如上所述，相关器负责启动/停止测量ToF tick的计时器。相位噪声会导致同步时间点不是确定在某一点，而是分布在一个范围内【Spreading】，即对ToF结果进行扩展，导致ToF结果分布在相邻的tick值之间。 TI已观察到方差约为0.64。

ToF协议中的各种设备没有用于同步其时钟的方法，因此，彼此之间的时钟存在一些固有偏移。tick值的spreading不会在执行ToF的设备之间同步。最终，这意味着结果将分布在整个系统的3-4个tick值中间。

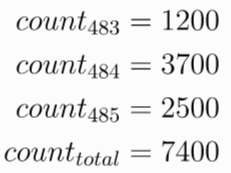
对于具有统计意义的集合，系统中的tick变动分布近似于正态分布，真实值是平均值。下面显示了ToF 的tick直方图示例，纵坐标是概率密度。



如果使用ToF例程采集更多样本，比方说7400。将ToF结果绘制为给定距离和样本数量的直方图，得出的结果类似于下图。



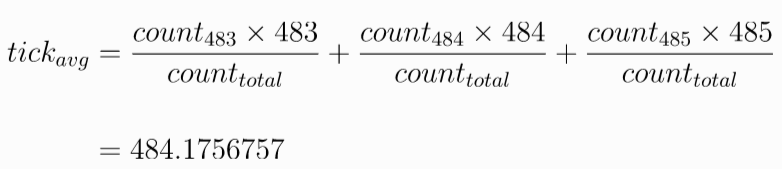
基于上述直方图，可得出如下结果:



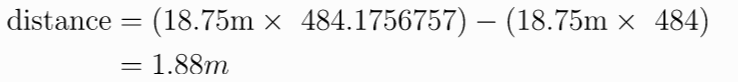
在真实距离计算中，我们必须考虑校准。 校准将在后续章节中深入介绍。 现在，假设校准值是484，单位是tick。



根据上面直方图得出tick的平均值：



再把平均值带入上面的方程式：



根据上面的演算过程可以得知，随着样本数量的增加，ToF可以变得更加准确。上述过程还展示了ToF如何用于测量小于18.75m的距离，这要归功于设备之间的不同步时钟所引起的扩展【spreading】以及单个设备中的相位噪声/漂移。

总体思路是，在给定的置信度下，使用更多的样本可以实现更窄的置信区间。 有关精度和置信区间的更多内容请查看simplelink academy链接：<http://dev.ti.com/tirex/explore/node?node=ADoNVRzIjRH1Ft0aHjErQg__pTTHBmu__LATEST>

5.8、TOF时隙和跳频

飞行时间可以在多个频率上运行并执行跳频。开始ToF测量之前，所有设备必须有统一的频率列表和跳频间隔。 CMD\_FS用于将频率合成器（frequency synthesizer--FS）编程为给定频率。

必须在第一个ToF\_run（）之前对频率合成器进行粗略校准，建议每小时重做一次。打开和关闭ToF驱动程序将触发粗略的重新校准。这部分的更多信息，请参见TOF\_calibrateSynth（）

粗校准完成后，CMD\_TOF将在RF核内部的中频上进行精细校准。运行CMD\_TOF之前不需要调用CMD\_FS。 ToF命令可以分为两部分：

A、精细FS校准：120us

B、ToF测量：400us

第一次跳到新频段时，需要520 us的时隙【slot】，在该频段上的其余测量将需要另外的400 us时隙。由于多径衰落取决于频率，因此建议在多个频率上执行ToF。

在调用TOF\_open（）之后进行第一次运行时，频率合成器也会进行一次预校准，并存储校准值。

5.9、链路质量指示（LQI）和多路径过滤

由多径衰落引起的相长干扰【*Constructive interference，接收信号是两个频率为f的电波的叠加，其相位差为θ，当θ为2π的整数倍时，两个电波相长叠加，接收信号增强。所以称两个电波相长干扰*】会导致非常高的LQI测量值。所以不建议在多径反射上测量ToF，因为这并不表示真实距离（非直接路径）。总之，非常高的LQI测量值通常表明该数据包已受到多径干扰。

ToF RF patch包括一个参数，该参数可让slave忽略LQI太高的数据包。此外，它还能够导出master和passive设备的LQI结果。因此，对LQI执行的过滤类型如下：

A、ToF slave上的RF core：不响应LQI高于阈值的数据包

B、ToF master和passive上的application core：丢弃LQI高于阈值的本地测量

从设备可以通过ToF命令的lqiThreshold参数配置LQI过滤。通过slaveLqiFilter RTLSCtrl参数在应用程序中进行设置。

应用程序的LQI过滤是在TOF\_getBurstStat（）中实现的。通过更改postProcessLqiThresh RTLSCTRL参数来设置阈值。

总之，对PING和PONG数据包执行LQI过滤。从设备接收到具有高于阈值的LQI的数据包，它将不响应（导致该SW超时）；PING / PONG成功完成，但是master或passive测量返回的数据包上的LQI太高，也会被应用程序过滤掉。

小提示：

当LQI或幅度差滤波器设置得过于“激进”时，系统可能会在burst期间误过滤掉很多样本。看上去没有ToF连接，其实是样本被过滤了。在TOF burst之前，确保ToF使用了足够多的样本来得出距离估算，这一点很重要。可以使用RTLS\_CMD\_TOF\_RESULT\_RAW观察没有过滤的所有样本。

多径干扰的另一个标志是在<< 1 >>和<< 0 >>符号中收到的RSSI的差异。 ToF PHY所使用的调制方案是频移键控（FSK），因此每个符号与载波的正负频率偏差相关。 多径的影响是频率选择性的，因此它们将对<< 1 >>和<< 0 >>符号产生不同的影响。 ToF patch可以为每个ToF样本输出<< 1 >>和<< 0 >>符号的幅度差比。 基于幅度差异的过滤在应用程序内核的TOF\_getBurstStat（）中实现， 幅度差异阈值可以通过postProcessMagnRatio RTLSCtrl命令设置。

最终，使用正确设置的LQI和幅度差滤波器可以抵消多径干扰对ToF测量的负面影响。

5.10、安全性

ToF的安全性基于以下前提：攻击者无法猜测ToF要使用的下一个同步字。这意味着同步字绝不能重复利用。如果攻击者可以轻松猜到下一个同步字，则系统很容易受到replay攻击。

因此，使用加密连接将随机种子分发给ToF从设备。使用串行连接将此种子发送到passive节点。

然后将此种子输入AESCTRDRBG算法，以生成新的同步字表，并将这些同步字用于ToF。为了在生成新的同步字表时优化radio的使用，可以对同步字列表进行双缓冲（取决于tofSecCfgPrms-> bUseDoubleBuffer的配置），以便安全模块可以在RFcore使用它们的同时生成新的同步字。

在双缓冲模式下运行时，TOF Security模块在TOF\_SEC\_DBL\_BUFF\_SIZE大小的块上运行。尽管双缓冲模式会减少运行burst所需的连续RAM数量，我们仍然推荐使用这一模式。使用单缓冲模式时，缓冲区大小将为同步字的总数。

注意：使用双缓冲模式（默认）时，建议将总同步字数设置为TOF\_SEC\_DBL\_BUFF\_SIZE的倍数，以充分利用双缓冲方案。

总而言之，TOF安全模块会生成一系列伪随机同步字，随机种子是使用True Random Number Generator（TRNG）硬件模块和驱动程序生成，并通过GATT配置文件分发给设备的。 种子长度由AESCTRDRBG\_SEED\_LENGTH\_AES\_128配置。

5.10.1、随机种子

5.10.2、AESCTRDRBG

同步字表是根据NIST SP 800-90Ar1的10.2.1.2节使用AES-128算法生成的。 这部分使用TI AESCTRDRBG驱动程序。 更多信息请参考：<http://dev.ti.com/tirex/explore/content/simplelink_cc13x2_26x2_sdk_3_30_00_03/docs/ble5stack/ble_user_guide/html/ble-stack-5.x-guide/api-reference-cc13x2_26x2.html#ti-driver-reference>

5.11、ToF多节点同步

如前文所述，ToF使用的同步字必须不断变化，以防止攻击者猜到它们。但是，这给passive节点带来了一个问题，由于以下原因，它可能会错过TOF burst：

A、主/从错过连接事件

B、主/从机无法执行ToF

C、passive错过连接事件

D、passive错过ToF

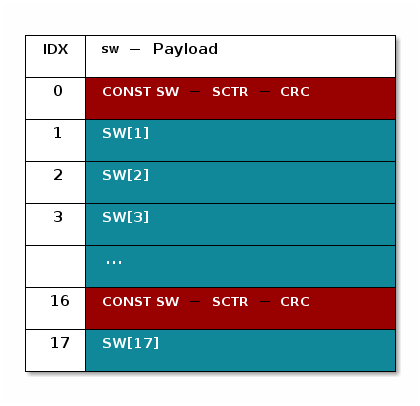
以上任何一种情况，passive节点都无法确定下一个应该是哪个SW，即使它具有安全种子。

广播数据包可以解决这种同步问题。当passive节点失去同步信息时，它将为它的相关器加载预定的恒定同步字，并开始寻找广播数据包。恒定同步字在所有节点上进行硬编码，由TOF\_CONSTANT\_SW设置。

ToF主机在ToF burst期间定期发送广播数据包。这些广播数据包是潜在的passive同步点（passive sync points---PSP）。

广播数据包有效负载的格式在上文“数据包类型”中进行了描述。当passive 节点收到广播数据包时，它将读取SCTR【Shared counter】并将其与本地计数器进行比较，以确定它错过了多少次PING + PONG交换，以及它需要在生成的同步字表中寻找多远。一旦passive节点知道丢失了多少个数据包，它就知道后面要监听的下一个同步字，并且以此重新获得同步。 SCTR的完整性受16位CRC保护，因此passive可以确定它正确接收了消息。主机负责更新SCTR和发送广播数据包。

下图显示了ToF在定期出现PSP /广播数据包时使用的同步字表：



5.12、TOF角色切换

ToF角色独立于RTLSCtrl角色（例如rtls\_passive，rtls\_master等）。这意味着rtls\_passive可以作为ToF主机，而rtls\_master可以作为ToF passive。这个概念称为角色切换。角色切换通过空间多样性提高了系统的稳健性。由于存在多径反射，所以空间多样性很重要。如果当前已将给定节点指定为ToF主节点，但无法测量直接路径，则可以角色切换并任命一个新节点作为主节点。放置在其他位置的新主机有机会通过直接路径从rtls\_slave接收PONG。

角色切换是通过RTLSCtrl命令实现的。有关该命令的更多信息，请参见下文的“TOF的RTLSCtrl”。建议监控每个ToF节点的统计信息（即正确采样的数量），并将结果最好的节点指定为下一个burst的ToF主节点。

角色切换不能在ToF burst期间执行，而必须在TOF\_run（...）的调用之间协调进行。如果在burst期间发送命令，它将在下一个burst中生效。

六、TOF驱动

ToF驱动程序负责管理“time of flight” 中RF命令的设置，以及提供RF驱动程序和应用程序之间的接口。 此外，ToF驱动程序将使用ToF安全模块生成随机种子。 ToF安全模块将使用高级加密标准计数器确定性随机位生成器（AESCTRDRBG）算法将随机种子用于安全同步字生成。

应用程序要做的相对简单。 只需要：

A、初始化

B、校准

C、运行程序

D、收集结果

6.1、初始化

ToF驱动程序包括一个带有信息的参数结构体。例程中已经填写了这些信息，主要的参数是：

tofRole---设备角色：主/从

pT1RSSIBuf---指向采样缓冲区的指针

numSyncwordsPerBurst---同步字数

pFrequency---指向频率数组的指针

numFreq---频率数

pfnTofApplicationCB---TOF运行后的回调

tofSecurityParams---安全性配置参数

freqChangePeriod---更改频率的周期

syncTimeout---等待第一个同步字的时间

pfnTofApplicationCB---应用程序的回调

slaveLqiFilter---自动过滤ToF从角色

postProcessLqiThresh---用于片上后处理的LQI阈值

postProcessMagnRatio---用于片上后处理的幅度无线电阈值

ToF驱动程序由RTLSCtrl模块管理，此处是解释其参数和行为以帮助理解。

6.2、运行

调用TOF\_run（handle，tofEndTime）后TOF将立即启动。 当与蓝牙结合使用时，tofEndTime取自BLE协议栈中的Radio Access Timer (RAT) 的tick。

注意：

tofEndTime以运行在4 MHz的Radio Access Timer (RAT)的滴答度来衡量，这与用于测量ToF的运行在8MHz的RF core timer不同，这一点需要区分。

当在协议栈通信（如蓝牙）之间安排协调ToF事件时，tofEndTime参数非常有用。

6.3、收集结果数据

收集数据是在初始化参数中提供的回调函数里完成的

ToF\_BurstStat tofBurstResults[TOF\_MAX\_NUM\_FREQ] **=** {0};

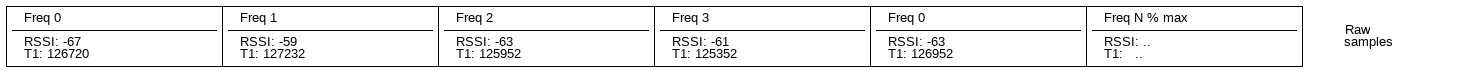
**void** **myCallback**(**uint8\_t** status) {

TOF\_getBurstStat(tofHandle, tofBurstResults);

*// Do something*

}

此函数获取交错的原始样本并按频率将其取平均：



原始样本存储在上图的平面列表中，统计函数按频率显示它们：

**typedef** **struct**

{

**uint16\_t** freq; *// Frequency*

**double** tick; *// Time of Flight in clock ticks, averaged over all valid samples for `freq`*

**double** tickVariance; *// Variance of the tick values*

**uint32\_t** numOk; *// Number of packets received OK for `freq`*

} ToF\_BurstStat;

Tick的值可按照前文的公式转化成米

6.4、校准

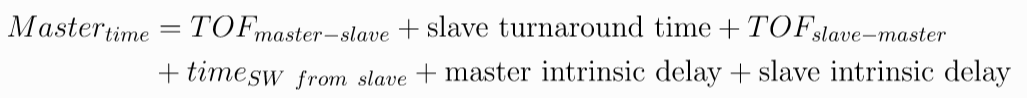
测量到的飞行时间长于实际的飞行时间。 测量误差是由以下原因引起的：

A、slave设备的周转时间（从RX切换到TX）

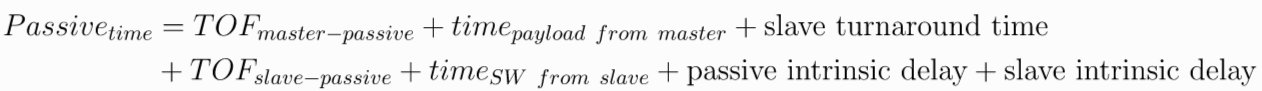
B、使用ToF PHY时设备的固有延迟

这些原因是确定有的且非零值。 根据硅片的特性，它们在设备之间略有不同。 这些特性在设备的整个使用寿命内都不会改变，因此每个设备只用进行一次校准然后存储即可。

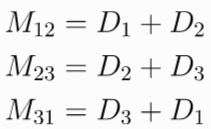
ToF主机的飞行时间是：



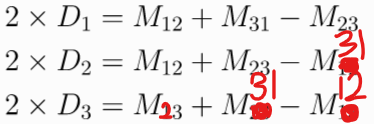
ToF passive设备的飞行时间是：



作为主机，每个ToF测量都包含两个应校准的延迟分量-----每个设备的固有延迟以及从机的周转时间。 假设系统由三个设备组成，则使这些设备轮流做主机，以产生以下结果，其中M\_12表示在设备1为主机而设备2为从机的情况下测量的ToF。 请注意，以下假设ToF = 0



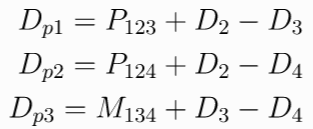
求解这些方程式将得出以下结果【原文有错误，改了一下】：



对于passive设备也是同样的道理，但要记住，passive设备的测量包含所有三个设备（主设备，从设备，passive设备）的延迟。 还是假定ToF = 0。 在以下方程式中， P\_123表示为passive设备的ToF测量数值，其中设备1为passive，设备2为主机，设备3为从机。 我们将在计算passive设备延迟的同时引入第4个设备，以展示如何随着添加passive设备而进行测量。



在设备间切换passive角色可得到以下方程式，请注意，这里我们假设有4个passive设备。 随着更多设备的添加，可以扩展以下方法。



求解上述方程得出：



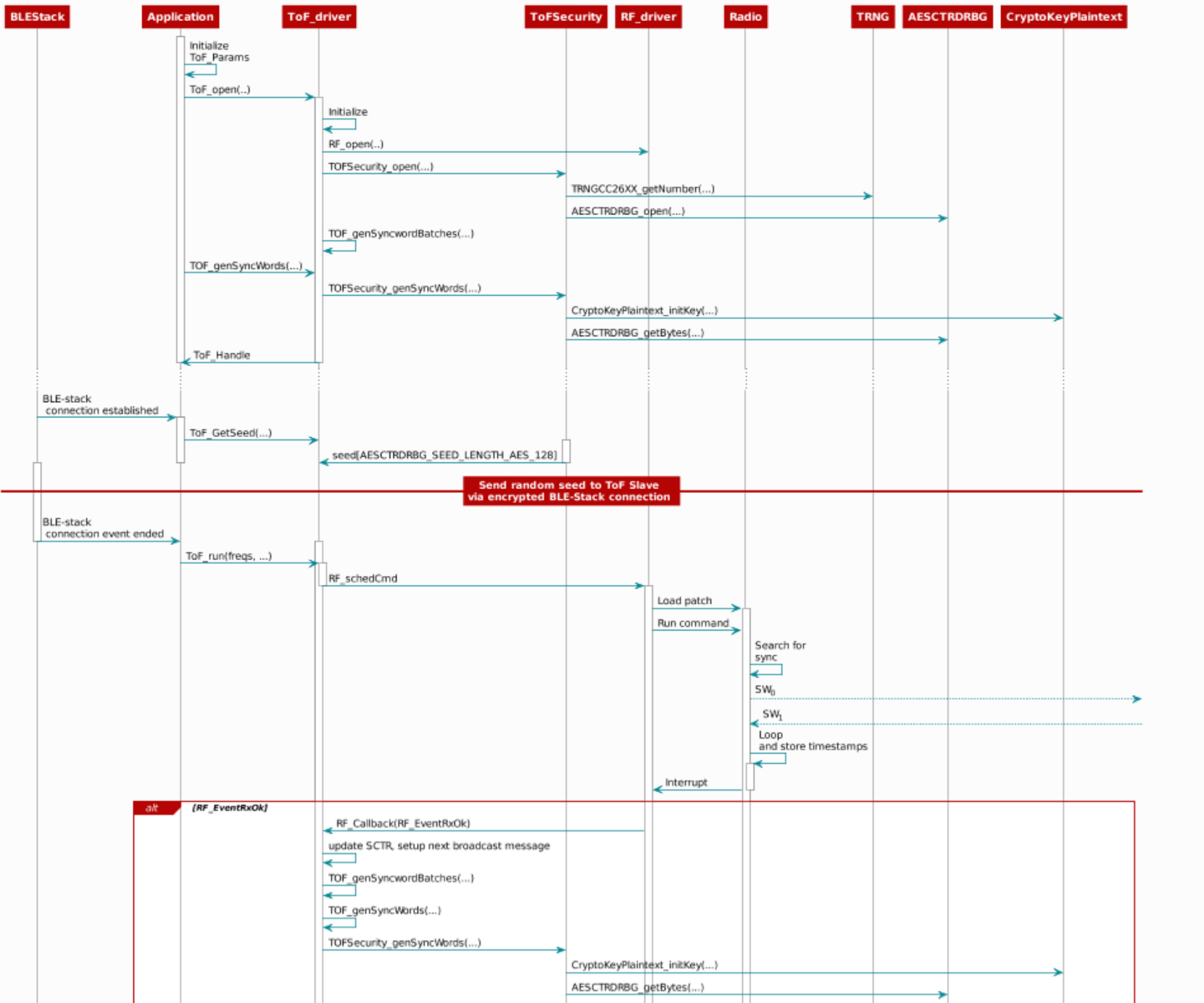
生产时应在可控环境中测量各器件的延迟，并应将值存储在非易失性存储器中，以便可以将其用于调整所有ToF的测量值。

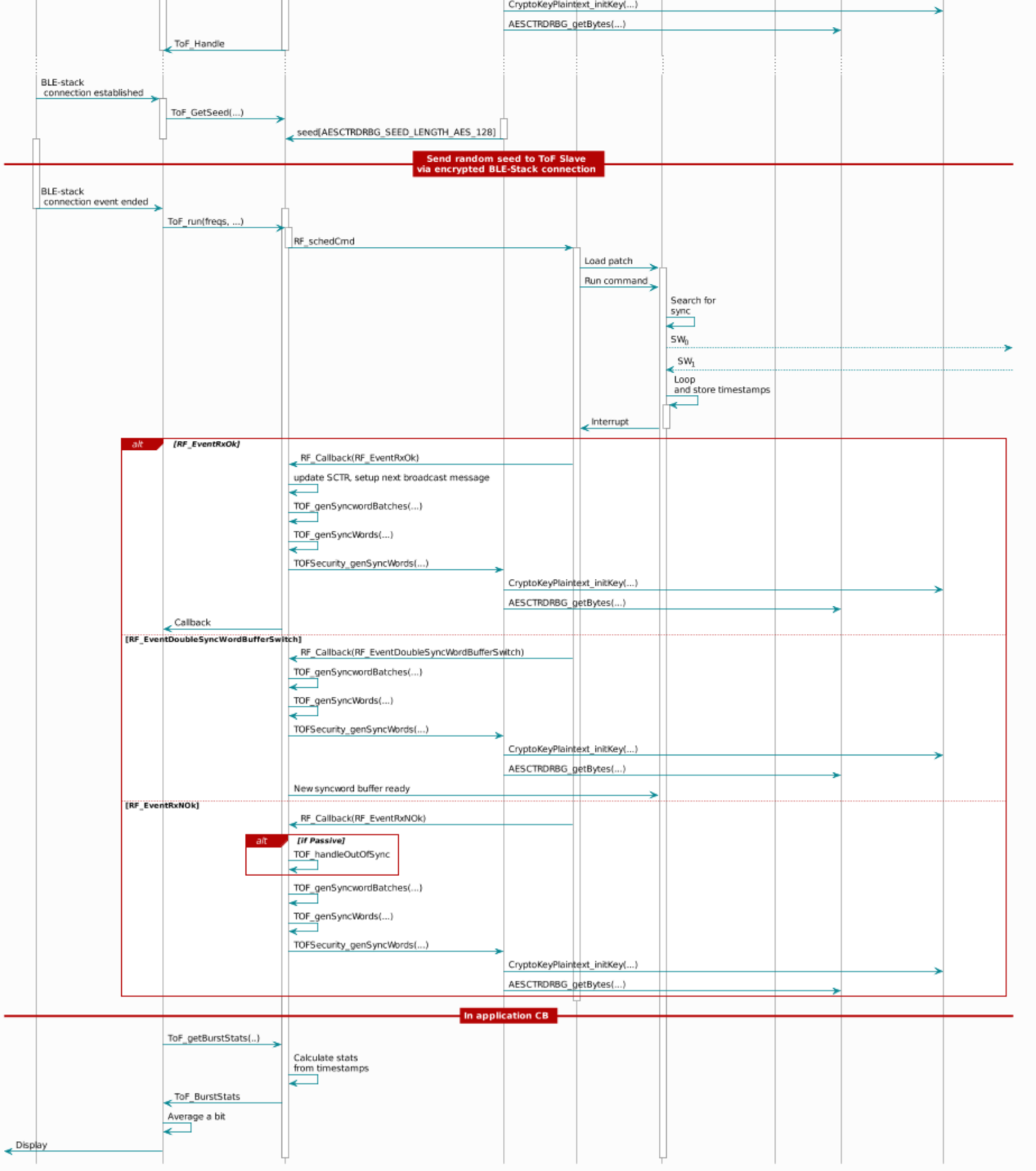
总而言之，ToF校准将产生一个tick值矢量，对于给定的ToF角色，在已知距离的每个频率上获得可计算的tick值。

6.5、应用

应用程序必须以某种方式与配对的设备达成共识----使用什么频率，跳频顺序以及应包含的同步字列表。另外，它必须在适当的时间调用ToF\_run（）来初始化同步字搜索和时隙测量burst。

下面是一个（非常简化的）时序图。 请注意，图中没有显示任何广播消息，但是它们会定期执行：





6.6、调试统计数据

ToF驱动程序包含一项功能，可以跟踪多个ToF测量的统计信息。 这些统计信息存储在全局结构（gTofStatistics）中，可以将其添加到IDE中的调试实时监视视图中。

通过定义RTLS\_TOF\_DEBUG来启用统计信息跟踪。

七、TOF的RTLSCtrl

本指南的前面各节介绍了飞行时间的理论，其RF核的功能以及底层的驱动程序和安全模块。

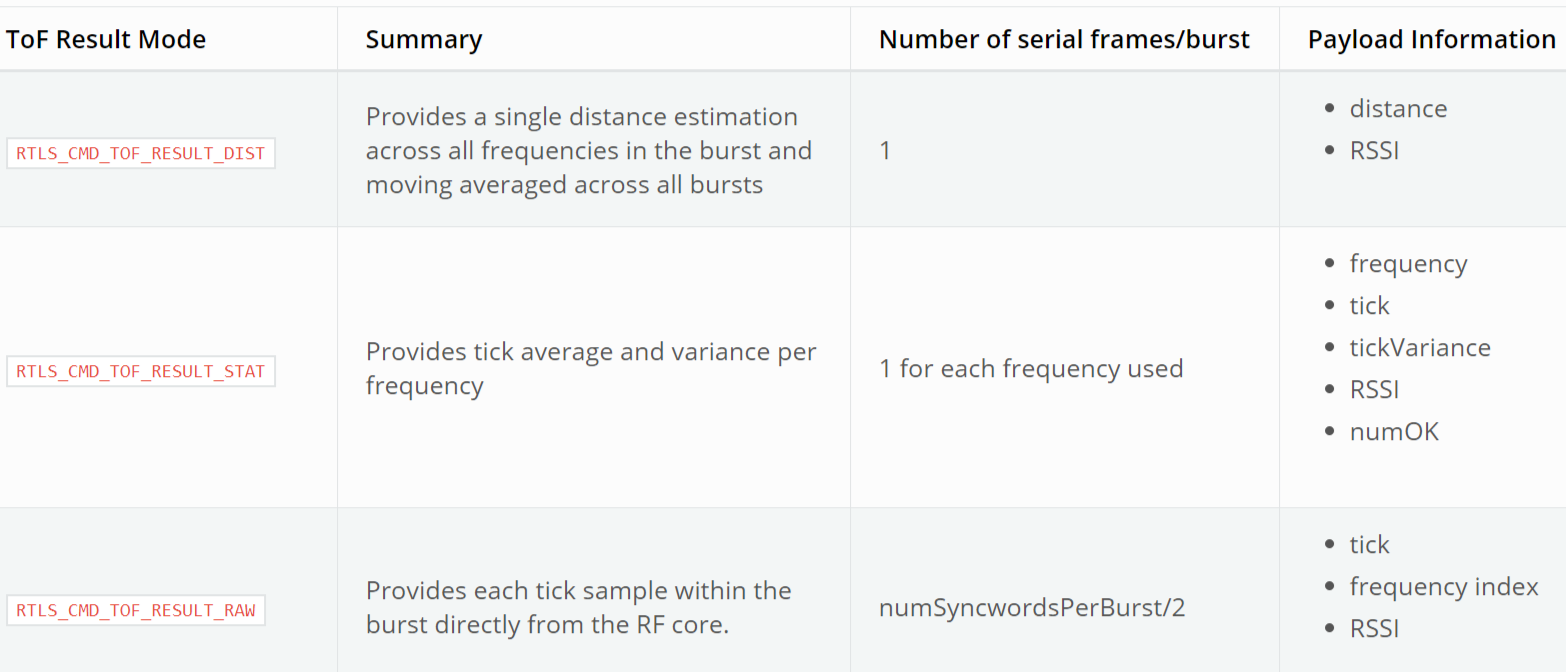
本节旨在描述通过RTLSCtrl接口展露的远程调用过程及TOF的相关参数。 有关RTLSCtrl如何作用于TI RTLS软件体系结构的介绍，请参见上文的“RTLS软件体系架构”一节。

7.1、TOF参数

可通过UNPI请求-- RTLS\_CMD\_TOF\_SET\_PARAMS来控制TOF的行为，该请求被发送到主节点和passive节点。 主机将选定的ToF设置通过BLE发送给从机。 运行ToF之前必须先调用RTLS\_CMD\_TOF\_SET\_PARAMS。更多信息请参见rtlsTofParams\_t。

7.1.1、TOF result mode

RTLS节点可以被配置为如下几种模式来显示TOF的采样数据：



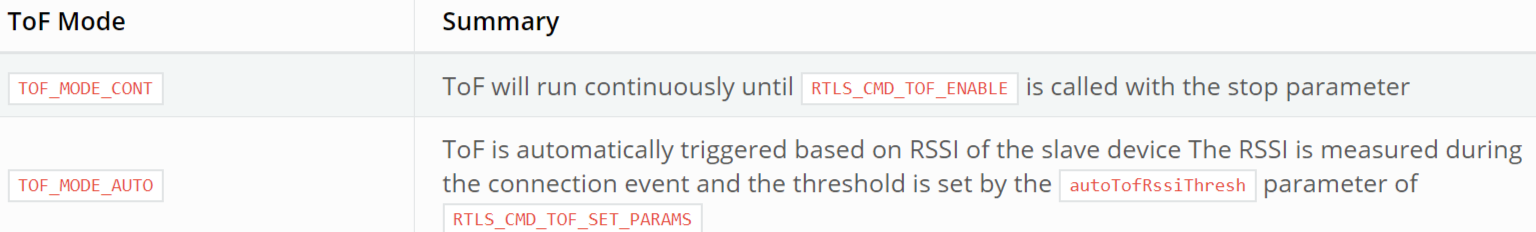
可以通过RTLS\_CMD\_TOF\_SET\_PARAMS的resultMode字段来控制ToF result模式。 表中位置较低的result mode可为PC提供更多数据，但会要求芯片更频繁地发送UNPI帧并使用更多的RAM。

注意：建议使用最长的模式在PC上构建ToF算法原型，然后通过平衡节点管理器和嵌入式设备之间的处理来降低数据发送的频率。

7.1.2、TOF run mode

runMode是RTLS\_CMD\_TOF\_SET\_PARAMS中的参数，它表示什么会触发ToF测量以及运行ToF测量的频率。

选择runMode时，必须考虑功耗预算。 连续运行ToF对于PC端的原型算法非常有用，因为它提供了大量数据，但对于电池供电的设备而言并不是最好的方法。 建议基于RSSI触发ToF，并且仅运行以收集达到给定置信区间所需的最小样本数。



7.1.3、每个TOF burst的同步字

RTLS\_CMD\_TOF\_SET\_PARAMS命令中另一个重点关注的参数是样本数量。 numSyncwordsPerBurst表示每个ToF burst使用多少个同步字。它需要两个同步字来创建一个ToF sample（PING + PONG）。等式如下：

NumTickMeasurements = numSyncwordsPerBurst / 2 - numPSPs

换句话说，每个TOF burst期间执行的tick测量次数为numSyncwordsPerBurst参数的一半减去burst中包含的任何passive同步点（passive sync points--PSP）。 PSP的周期由ToF命令的广播结构----RF\_cmdTof.BC中的字段定义。默认情况下，AESCTRDRBG每批同步字有一个PSP。 （由gTofSecHandle.numOfSyncWordsPerBuffer设置）

注意：请记住，并非所有的ToF模式都报告每个样本的原始tick值。这意味着当增加numSamples时，您可能不会看到更多数据发送到PC。这是因为在诸如RTLS\_CMD\_TOF\_RESULT\_DIST和RTLS\_CMD\_TOF\_RESULT\_STAT之类的模式下，会对设备上获得的样本进行一些预处理。

7.1.4、频率

RTLS\_CMD\_TOF\_SET\_PARAMS还有频率列表和频率数量的参数。 频率列表中的每个条目对应于执行“ToF”的频道。 在为ToF选择频率时，多样性非常重要，因为诸如多径衰落之类的物理因素是频率选择性的。

frequencies是指用于ToF的以MHz为单位的BLE频率列表

numFreq是指此列表中的通道数量

BLE通道的间隔为2 MHz，范围为[2402，2480]MHZ。 尽管ToF不直接使用BLE的物理数据包格式，但它使用相同的通道映射。信道由其载波的频率指定。

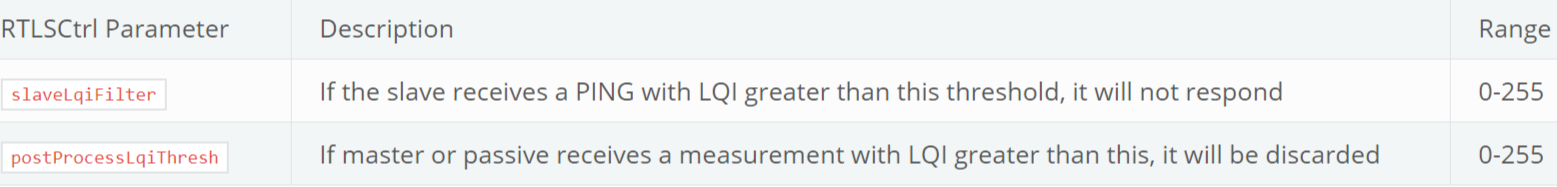
可以使用以下代码在Python中初始化频率列表：

tofFreqList **=** [2408, 2412, 2418, 2424] *#Other options: 2414, 2420*

tofNumFreq **=** len(tofFreqList)

7.1.5、LQI过滤

如前文的 链路质量指示（LQI）和多路径过滤 中所述，有两个可配置的LQI过滤值分别影响主/被动和从设备的LQI处理。 有关每个参数及其用法的说明，请参见下表：



7.1.6、倍率过滤器

多径干扰的另一个特征是<< 1 >>符号和<< 0 >>符号的RSSI之间存在差异。 RF core将报告<< 1 >>和<< 0 >>符号的幅度，并作为ToF结果的一部分，结果将在TOF\_getBurstStat（）中进行后处理。 sym1 / sym0和sym0 / sym1之比将被计算为浮点数倍。 只要其中一个比率大于postProcessMagnRatio / 100.0，其结果就会被丢弃。

7.2、角色切换

角色切换的好处及其理论在前文的“ToF角色切换”中已经进行了描述。 本节介绍实现角色切换的RTLSCtrl，即RTLS\_CMD\_TOF\_SWITCH\_ROLE命令。 该命令包含一个8位参数，该参数是新角色的枚举。 如前所述，仅可将角色从主切换到passive，反之亦然。 角色切换可以独立于设置ToF参数。 角色切换命令将对状态更改进行验证，然后相应地设置RF\_cmdTof.bMaster。 请注意，角色切换不能在ToF burst期间发生，必须在burst之间，由主和passive进行协调执行。

有效的ToF角色在TOF.h :: ToF\_Role中定义。

7.3、校准

如前文的“校准”中所述，ToF的校准向量必须包含以下内容：角色，频率，距离和tick平均值。校准是一种用于测量和存储给定频率，角色和距离的tick平均值的方法。所有频率都以RTLS\_CMD\_TOF\_CALIBRATE 命令中calibDistance参数设定的相同距离进行校准。校准过程包括对足够多的样本进行平均，以产生每个频率的采样数，该数量由RTLS\_CMD\_TOF\_CALIBRATE命令的samplesPerFreq参数指定。

该命令具有以下请求和响应类型：

SyncReq：向节点发送请求以开始校准

SyncRsp：在参数验证和校准开始时，返回状态

AsyncRsp：为每个频段存储了samplesPerFreq个采样数之后，返回校准向量。

校准命令有如下几种行为：

useCalibFromNV == True，并且NV中存在校准信息：无任何操作，立即返回

useCalibFromNV == True，并且NV中不存在校准信息：开始校准，给ToF burst的数据取平均，直到达到samplesPerFreq的采样数。存储结果及calibDistance的值。

useCalibFromNV == False：执行校准，无论NV中有没有校准信息。校准结果将存储在NV中。

校准命令遵循非易失性存储体系结构。它将使用OSAL SNV存储校准数据。这是因为OSAL SNV实现了EEPROM仿真并允许覆盖校准而又不擦除整个页面。

TOF校准使用的NVID从BLE\_NVID\_CUST\_START开始，并使用以下数量的NVID：

1 +（calibParams.numFreq / RTLS\_TOF\_FREQ\_PER\_NV\_BLOCK）+（calibParams.numFreq％RTLS\_TOF\_FREQ\_PER\_NV\_BLOCK）

校准过程将使用第一个NVID来存储有关校准的元数据，包括频率列表和校准距离。后面的NVID用于存储ToF\_BurstStat，即在校准过程中计算出的每个频率的平均值。

可以使用RTLS\_CMD\_TOF\_CALIB\_NV\_READ读取校准向量。

警告：校准取决于角色，频率列表和设备。这意味着，如果更改了ToF频率列表，角色或设备，则必须重新校准。

当前，仅在TOF\_MODE\_DIST中使用校准结果，但是对于所有操作模式都适用相同的概念（对Tick值进行平均来为已知距离建立参考点）。在RAW或STAT模式下，应计算校准并将其存储在PC上。

八、跟BLE交错传递信息

连接中的BLE设备势必会在连接事件的某些时间唤醒进行发送或接收。 这些事件之间的间隔称为“连接间隔”。

passive设备集成了connection monitor功能，因此，它可以获得连接信息（rssi，时间戳），并且在每个连接事件结束时侦听ToF数据包并提供计算出的距离。

如下图所示，在一个连接事件结束到下一个连接事件开始之间的时间里，有时间安排其他RF命令，例如ToF。

可用时间可能有所不同，但是在连接事件完成回调中提供了时间戳，用于使用无线发送TOF数据包。

