源代码指南

DECARANGERTLS ARM 源代码

理解和使用 DecaRangeRTLS ARM 源代码

2.1 版

本文如有更改，恕不另行通知。

文件信息

免责声明

Decawave 保留更改产品规格的权利，恕不另行通知。功能和规格的尽可能更改将在产品特定勘误表或本文档的新版本中发布。建议客户查看 Decawave 网站以获取该产品的最新更新

生活支持政策

Decawave 产品未被授权用于安全关键应用（例如生命支持），在这些应用中，有理由认为 Decawave 产品的故障会导致严重的人身伤害或死亡。以这种方式使用或销售 Decawave 产品的 Decawave 客户完全自担风险，并同意完全赔偿 Decawave 及其代表因在此类安全关键应用中使用 Decawave 产品而造成的任何损害。

警告！ESD敏感器件。
处理设备时应采取预防措施，以防止永久性损坏

免责声明

本免责声明适用于由 Decawave Ltd.（“Decawave”）提供的 DecaRanging RTLS-ARM 源代码和 DecaRanging RTLS-PC 源代码（统称为“Decawave 软件”）。

下载、接受交付或使用 Decawave 软件即表示您同意本免责声明的条款。如果您不同意本免责声明的条款，请勿下载、接受交付或使用 Decawave 软件。

Decawave 软件包含 ST Microelectronics (“STM”) 在 STM 的 Liberty V2 下提供给 Decawave 的 STSW-STM32046（STM32F105/7、STM32F2 和 STM32F4 USB on-the-go 主机和设备库 (UM1021)）软件（“STM 软件”） 11 月 16 日的软件许可协议th2011 年可用这里 （“STM 软件许可协议”）。下载、接受交付或使用包含在 Decawave 软件中的 STM 软件即表示您同意 STM 软件许可协议的条款，特别是要求 STM 软件仅与 STM 微控制器一起使用，而不与任何其他制造商的微控制器一起使用。如果您不希望接受 STM 软件许可协议的条款，那么您仍然可以使用 Decawave 软件，但前提是您不使用其中包含的 STM 软件。

Decawave 软件仅旨在帮助您开发包含 Decawave 半导体产品的系统。您理解并同意您仍然有责任在设计您的系统和产品时使用您的独立分析、评估和判断。在您的系统和产品中全部或部分使用 DECAWAVE 软件的决定完全由您决定。

DECAWAVE 软件按“原样”提供。DECAWAVE 对 DECAWAVE 软件或 DECAWAVE 软件的使用不作任何明示、暗示或法定的保证或陈述，包括准确性或完整性。DECAWAVE 不提供与 DECAWAVE 软件或其使用相关的任何所有权保证以及对适销性、特定用途的适用性和不侵犯任何第三方知识产权的任何默示保证。

对于与或基于 DECAWAVE 软件或将 DECAWAVE 软件与 DECAWAVE SEMICONDUCTOR 技术一起使用的任何第三方侵权索赔，DECAWAVE 概不负责，也不应为您辩护或赔偿您。在任何情况下，DECAWAVE 均不对任何实际的、特殊的、附带的、间接的或间接的损害负责，包括但不限于上述一般性损失、预期利润、商誉、声誉、业务收据或合同、采购成本的损失替代商品或服务；使用、数据或利润的损失；或业务中断）、第三方索赔导致的损失或费用。无论采取何种行动形式，无论是否根据法规，这些限制都将适用，

您被授权在您的最终产品中使用 Decawave 软件，并在您的最终产品开发过程中修改 Decawave 软件。但是，此处未授予任何其他明示或暗示的许可，以禁止反言或以其他方式授予任何其他 DECAWAVE 知识产权，也未授予任何第三方技术或知识产权的许可，包括但不限于任何专利权、版权、与使用 Decawave 半导体产品或 Decawave 软件的任何组合、机器或过程相关的掩模工作权或其他知识产权。

您承认并同意，您全权负责遵守与您的产品有关的所有法律、法规和安全相关要求，以及在您的应用程序中使用 Decawave 软件，尽管 Decawave 可能提供任何与应用程序相关的信息或支持。

Decawave 保留随时对其软件进行更正、增强、改进和其他更改的权利。

邮寄地址： -

迪卡波有限公司，
阿德莱德钱伯斯，
彼得街，
都柏林 8

目录

1个检视

这个文件，”*DecaRangeRTLS ARM 源代码指南*》是Decawave的“TREK1000”双向测距RTLS演示应用程序的应用程序源代码指南，该应用程序运行在EVB1000开发平台上的ARM微控制器上。

本文件应与“*TREK1000 用户手册*”，其中概述了 DW1000 RTLS 评估套件 (TREK1000)，并描述了如何操作 DecaRangeRTLS 应用程序。

本文档讨论了 DecaRangeRTLS ARM 应用程序的源代码，涵盖了软件的结构和测距 RTLS 演示应用程序的操作，特别是范围的计算方式。

第 6 节以软件执行流程演练的形式编写。它应该能够很好地理解发射和接收的基本操作步骤，这反过来应该有助于将测距功能集成/移植到客户平台。

本文档涉及以下版本："DecaRangeRTLS ARM 2.10"应用程序版本和"DW1000 设备驱动程序版本 02.16.01"驱动版本。设备驱动版本信息可以在源代码文件“deca_version.h”，应用程序版本在“主程序”。

图1下面显示了 DecaRange RTLS 应用程序的分层结构，给出了与每一层关联的主要文件的名称以及该层提供的功能的简要说明。

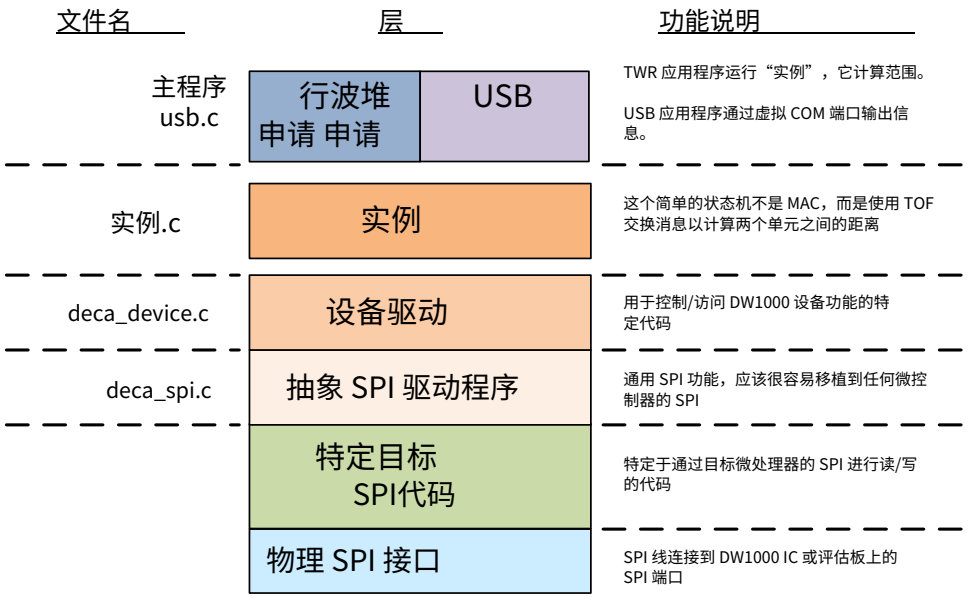

图 1：DecaRange RTLS ARM 应用程序中的软件层

所涉及的层、函数和文件将在下一节中描述。

2 天铭文D非洲经委会R安吉RTLS 臂代码结构

参考图1 ，所识别的层将在下面更详细地描述。

2.1 目标特定代码

低级 ARM 特定代码可以在 \src\platform\ 中找到——这两个文件*端口.c*和*端口.h*定义启用和使用的目标外设和 GPIO，即 SPI1 用于与 DW1000 进行 SPI 通信，SPI2 用于与 LCD 进行 SPI 通信，其他 GPIO 线用于应用程序配置和控制。

SPI1：

```
# 定义 SPIx                    SPI1
# 定义 SPIx_GPIO               通用输入输出接口
# 定义 SPIx_CS                 GPIO_Pin_4
# 定义 SPIx_CS_GPIO            通用输入输出接口
# 定义 SPIx_SCK                GPIO_Pin_5
# 定义 SPIx_MISO               GPIO_Pin_6
# 定义 SPIx_MOSI               GPIO_Pin_7
```

SPI1 外设用于与 DW1000 SPI 总线通信。

中断线：

```
# 定义    DECAIRQ              GPIO_Pin_8
# 定义    DECAIRQ_GPIO         通用输入输出接口
# 定义    DECAIRQ_EXTI         EXTI_Line8
# 定义 DECAIRQ_EXTI_PORT       GPIO_PortSourceGPIOA
# 定义    DECAIRQ_EXTI_PIN     GPIO_PinSource8
# 定义    DECAIRQ_EXTI_IRQn    EXTI9_5_IRQn
# 定义    DECAIRQ_EXTI_USEIRQ  使能够
```

DW1000 中断线连接到 GPIOA 引脚 8。 注意：对于 MP，该线为高电平有效。

液晶驱动器：

```
# 定义 SPIy                    SPI2
# 定义 SPIy_GPIO               通用输入输出接口
# 定义 SPIy_CS                 GPIO_Pin_12
# 定义 SPIy_CS_GPIO            通用输入输出接口
# 定义 SPIy_SCK                GPIO_Pin_13
# 定义 SPIy_MISO               GPIO_Pin_14
# 定义 SPIy_MOSI               GPIO_Pin_15
```

SPI2 外设用于与 LCD 通信。

应用程序配置开关（S1）：

```
# 定义 TA_SW1_3                GPIO_Pin_0
# 定义    TA_SW1_4             GPIO_Pin_1
# 定义    TA_SW1_5             GPIO_Pin_2
# 定义    TA_SW1_6             GPIO_Pin_3
# 定义    TA_SW1_7             GPIO_Pin_4
# 定义    TA_SW1_8             GPIO_Pin_5
# 定义    TA_SW1_GPIO          GPIOC
```

配置开关（**S1**) 用于在*锚*和*标签*模式和各种通道配置。看 ” *TREK1000 用户手册* “ 更多细节。

src\compiler\compiler.h 包含可以根据需要替换的标准库文件（例如，如果希望使用针对较小代码大小优化的函数，例如）。

2.2 抽象 SPI 驱动程序——SPI 级代码

文件*deca_spi.c*提供抽象的 SPI 驱动函数*openspi()*,*closespi()*,*writetospi()*和*读取spi()*. 这些映射到 ARM 微控制器的 SPI 接口驱动程序。

2.3 设备驱动——DW1000 设备级代码

文件deca_device_api.h提供与 API 函数库的接口，以控制和配置 DW1000 寄存器并实现设备级控制功能。API 函数在“*DW1000 设备驱动程序应用程序编程接口 (API) 指南*”文档。

2.4 实例代码

实例代码（在实例.c) 提供了一个简单的双向测距 RTLS 演示应用程序。此实例代码位于 MAC 通常驻留的位置。为了方便开发测距 RTLS 演示以展示 DW1000 的测距和性能，测距 RTLS 演示应用程序直接在 DW1000 驱动程序 API 之上实现。

双向测距 RTLS 演示应用由状态机在函数中实现*测试应用程序（）*, 从函数调用*实例运行（）*，这是运行实例代码的主要入口点。实例以不同的模式运行（*标签*或者 *锚*) 取决于在应用层设置的角色配置。这*标签*和 *锚*模式成对运行以提供两个单元之间的双向测距演示功能。在 TREK 中，单个标签范围最多 4 个锚点，然后单独的 DecaRangeRTLS PC 应用程序可以使用这些范围来计算标签相对于锚点的位置并显示在 GUI 上。

2.4.1 操作模式——标签

一旦通电，标签单元将尝试定位到 4 个锚点，然后进入睡眠状态。经过一段时间（超帧/调度周期，$T_{科幻}$) 它将再次唤醒并范围为 4 个锚点。标签同时覆盖所有 4 个锚点。它以广播消息的形式发送 Poll（目标地址设置为 0xFFFF），接收任何响应，然后发送 Final。如果没有收到来自锚#0 的响应，则不会发送最终消息。

110 kbps 的调度周期为 280 ms，6.81 Mbps 模式的调度周期为 100 ms。这是从标签发送轮询、完成测距交换或超时到再次发送轮询的同一时刻的时间段。这些调度周期时间与测距交互的持续时间和为超帧定义的交互时隙的数量有关（如下所述）。请注意，6.81 Mbps 模式可能具有更快的更新速率，但为 6.81 Mbps 模式超帧确定并定义了 10 Hz 更新速率。

图 2 概述了超帧结构，图 6 和图 7 给出了帧时序。

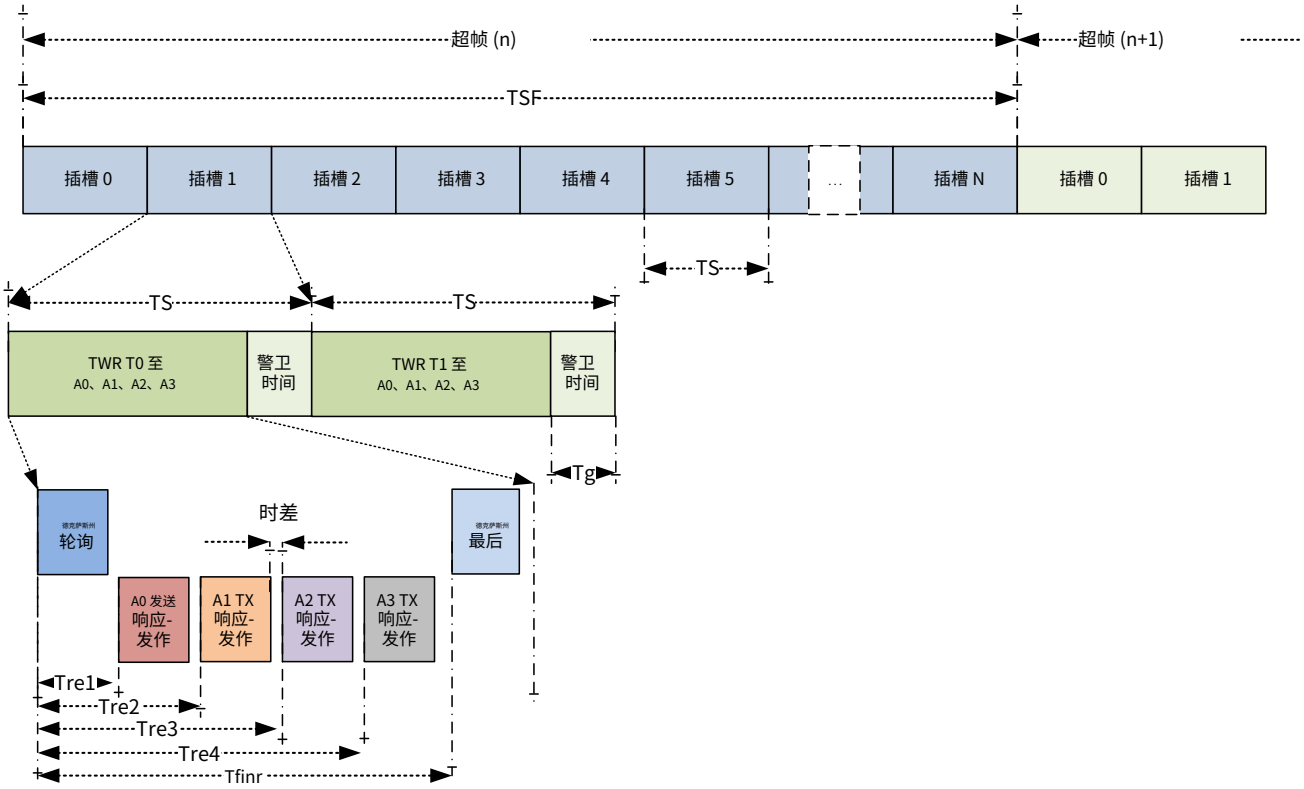

图 2：标记 TWR RTLS 时间配置文件

为了支持多个标签（不干扰），我们使用了 TDMA 方法。在我们的 TREK 示例模式中，超帧中有 10 个插槽。每个时隙在 110 kbps 模式下为 28 ms，在 6.81 Mbps 模式下为 10 ms，并且，由于我们为每个标签分配一个时隙，因此最多可以支持 8 个标签。为锚点到锚点的测距预留了两个时隙。

锚#0，负责将标签分配和维护到它们自己的槽中。

2.4.2 操作模式——锚#0

首先，Anchor #0 为每个标签分配活动时隙，以便它们的测距交换不会相互干扰。锚#0 通过在每次响应标签的测距尝试时包含一个睡眠时间调整值来做到这一点。计算此休眠时间调整以根据它们的地址（#0 到 #7）将要唤醒的每个标签定位在单独的插槽中。

其次，Anchor #0 通过其 USB 端口报告所有测距交换的结果。它通过侦听和接收其他锚点嵌入其响应消息中的飞行时间结果来做到这一点，并将这些结果与自己计算的 TOF 结果一起收集，然后将每个标签的全套测距结果发送到并附加PC 通过其 USB 端口。

Anchor #0 也开始锚点到锚点的测距，用于自动定位功能。超帧的最后两个时隙用于此目的。锚#0最初的范围是锚#1和#2，然后锚#1的范围是锚#2。所得的六个范围通过 USB 输出，以便可以计算相对于彼此的锚点位置。

2.4.3 操作模式——锚#1 和#2

锚点 #1 和 #2 涉及到锚点测距和锚点到锚点测距的标签。

2.4.4 操作模式——锚#3

Anchor #3 仅涉及到锚定测距的标签。它忽略任何锚点来锚定测距消息。

2.4.5 范围结果

测距结果通过 USB 端口输出。标签将报告它从与其范围内的锚点接收到的任何范围结果，由锚点的响应消息返回给它。

所有锚点都会报告他们为与它们一起范围内的标签计算的范围，而且还会报告他们从其他锚点收到的任何范围报告（都将在响应消息中接收 TOF）。不使用帧过滤，因此可以接收空中的任何消息。

2.4.6 测距方法

测距方法使用一组三个消息来完成计算距离的两次往返测量。随着消息的发送和接收*十进制RTLS*ARM 应用程序从 DW1000 检索消息发送和接收时间。这些发送和接收时间戳用于计算往返延迟并计算范围。图 3 显示了由 DecaRangeRTLS ARM 应用程序实现的双向测距的安排和一般操作。

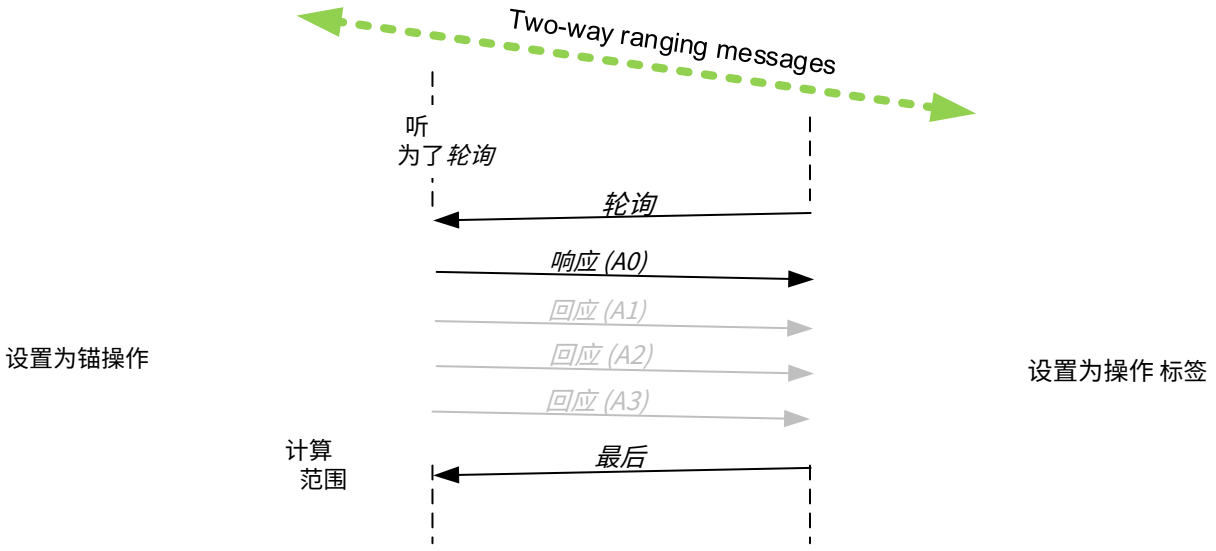

图 3：DecaRangeRTLS ARM 中的两种测距方式

在此实例级别之上是提供下面第 2.5 节中描述的用户界面的应用程序。除了*实例运行（）*函数，实例代码为应用程序提供控制功能。此处列出了这些功能，以便读者快速了解功能：-

instance_init()、*instance_config()*、*instance_run()*、*instance_close()*、*instancedevicesoftreset()*、*instancegetrole()*、*instancesetrole()*、*instancesetantennadelays()*、*instancereaddeviceid()*、*instance_readaccumulatordata()*、*instancesetreporting()*、*instancegetcurrentrangeinginfo()*、

instancesetaddresses()、*instancesettagsleepdelay()*、*instancesetreplydelay()*、*instancesetspibitrate()*、*instance_rxcallback()*、*instance_txcallback()*、*inst_processrxtimeout()*。

读者被定向到文件中的代码实例.c有关这些功能的更多详细信息。

2.5 顶级应用代码

顶级应用程序（主程序) 包含 DecaRanging 测距演示应用程序的主要入口点以及所有用户界面代码。

应用层展示结果的能力取决于硬件平台的能力。在 EVB1000 评估板上，LCD 用于显示范围测量的结果范围。

表 1：DecaRange RTLS 应用程序中的 LCD 显示消息

文件名	简要描述;简介
1234567890123456	此行仅用于调整固定空间字体的大小，以便此处定义的文本适合 EVB1000 上的 16x2 字符显示
USB 转 SPI	单元处于 USB 到 SPI 转换模式
十进制RTLS L2 T3 xxxxxxxxxxxx	设备处于 RTLS，通道“2”上的“L”（远程）模式，并作为标签 #3 运行，请参见下文以了解 xxxxxxxxxxxx 的定义。
十进制RTLS S5 A0 xxxxxxxxxxxx	设备处于 RTLS，通道“5”上的“S”（短帧）模式，并作为 Anchor #0 运行，有关 xxxxxxxxxxxx 的定义，请参见下文。
十进制RTLS L2 LS xxxxxxxxxxxx	设备处于 RTLS，通道“2”上的“L”（远程）模式，并作为监听器运行，请参见下文了解 xxxxxxxxxxxx 的定义。
AiTj:rrr.rrm	其中 xxxxxxxxxxxx 是文本（左），显示从设备收到的最后一个测距报告中提取的信息，其中： 一世- 是锚地址（最不重要的半字节）， j - 是标签地址（最不重要的半字节）， rrr.rr - 是这个标签和锚点之间的范围，以米为单位到小数点后两位，字符“A”、“T”、“:”和“m”就是这些字符。
连续 TX L2 频谱测试	表示连续传输测试模式处于活动状态。在这种情况下，在通道 2 上使用 LR（远程）帧格式。

2.6 发送到 USB/虚拟 COM 端口的测距结果格式

该应用程序还通过虚拟 COM 端口输出测距和一些调试信息。**图 4**显示在 Teraterm 终端仿真器（通信程序）上查看的锚点 0 的示例输出。

Windows PC 驱动程序可从 ST Microelectronics 网站获得，有关其安装的详细信息，请参阅 TREK1000 用户手册。

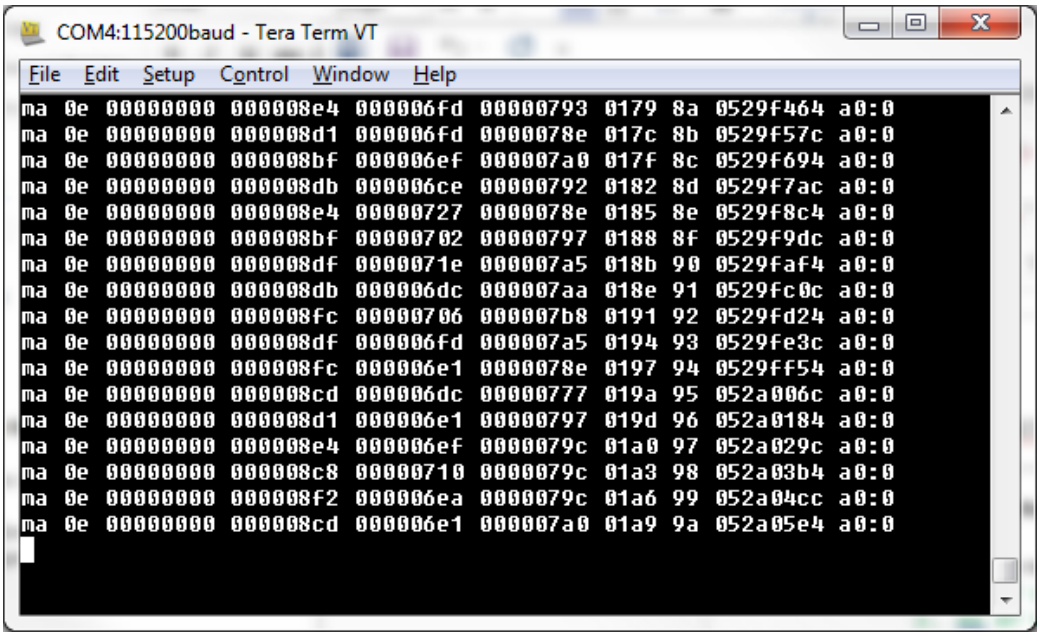

图 4：显示通过 COM 端口发送的调试信息的示例 Teraterm 窗口

通过 USB 端口发送三个测距报告消息：

中间掩码 RANGE0 RANGE1 RANGE2 RANGE3 NRANGES RSEQ DEBUG aT:A

1. 先生 0f 000005a4 000004c8 00000436 000003f9 0958 c0 40424042 a0:0
2. ma 07 00000000 0000085c 00000659 000006b7 095b 26 00024bed a0:0
3. mc 0f 00000663 000005a3 00000512 000004cb 095f c1 00024c24 a0:0

“mr”消息由标签到锚原始范围、“mc”标签到锚范围偏差校正范围——用于标签定位和“ma”锚到锚范围偏差校正范围——用于锚自动定位。

中	这是消息 ID，如上所述：mr、mc 和 ma
面具	这表明哪些 RANGE 有效，如果 MASK=7，则只有 RANGE0、RANGE1 和 RANGE2 有效（十六进制，8 位数字）
范围0	如果 MID = mc/mr（以毫米为单位，32 位十六进制数），这是锚定 ID 0 范围的标签
范围1	如果 MID = mc/mr，这是锚点 ID 1 范围的标签；如果 MID = ma，这是锚点 0 到锚点 1 范围的标签（以 mm 为单位，32 位十六进制数）
范围2	如果 MID = mc/mr，这是锚点 ID 2 范围的标签；如果 MID = ma，则锚点 0 到锚点 2 范围（单位为 mm，32 位十六进制数）
范围3	如果 MID = mc/mr，这是锚点 ID 3 范围的标签；如果 MID = ma，这是锚点 1 到锚点 2 范围的标签（单位为 mm，32 位十六进制数）
范围序列号	这是通过报告单元原始范围（16 位十六进制数）完成的范围数这是范围序列号（8 位十六进制数）
调试	这是 TX/RX 天线延迟（如果 MID = ma）——两个 16 位数字或最后报告范围的时间——如果 MID = mc/mr（32 位十六进制数字）
aT:A	T 是标签 ID，A 是锚点 ID

2.7 源代码文件完整列表

表 2 列出了构成 DecaRangeRTLS ARM 应用程序源代码的文件。文件名连同文件及其用途的简要说明一起给出。有关代码结构和组织的更多详细信息，请参阅本文档的其他部分。

表 2：DecaRangeRTLS ARM 应用程序中的源文件列表

文件名	简要描述;简介
deca_version.h	Decawave 的 DW1000 驱动程序/API 代码的版本号
deca_device.c	设备级函数——源代码
deca_device_api.h	设备级功能 - 标题
deca_mutex.c	用于互斥的 IRQ 禁用占位符 - 源代码
deca_param_types.h	定义参数和配置结构的标题
deca_params_init.c	初始化用于设置 DW1000 的配置数据
deca_range_tables.c	包含测距校正表
deca_regs.h	设备级别 - 标头（设备寄存器定义）
deca_spi.c	SPI接口驱动——源代码
deca_spi.h	SPI接口驱动——头
dma_spi.c	SPI接口驱动——DMA实现的源代码
deca_types.h	数据类型定义——标题
端口.c	ARM 外设和 GPIO 配置
端口.h	ARM 外设和 GPIO 配置定义
主程序	应用程序——源代码（主线）
Instance_calib.c	应用程序的校准功能和数据 - 源代码
Instance_common.c	常用应用函数——源代码
实例.c	测距应用实例 – 源代码
实例.h	测距应用程序实例 - 标头
编译器.h	包含标准库文件
stm32f10x_conf.h	STM 库配置/包含文件
stm32f10x_it.c	中断处理程序在这里定义。
stm32f10x_it.h	此处声明了中断处理程序。
usb.c	USB 应用程序 – 通过虚拟 COM 端口输入/输出

3 R愤怒一个逻辑

本节介绍 DecaRangeRTLS 演示应用程序中使用的测距算法。标签将尝试定位到四个锚点，然后进入深度睡眠模式。（它会在超帧周期后被唤醒以重新开始循环。）

3.1 DecaRangeRTLS ARM应用的Tag/Anchor双向测距算法

对于该算法，一端充当标签，周期性地启动距离测量，而另一端充当锚点，监听并响应标签并计算距离。

在测距方案中，标签发送轮询消息，该消息由基础设施中的三个（或四个）锚点接收。锚点以 RespA、RespB 和 RespC 包连续响应，之后标签发送所有锚点接收到的 Final 消息。这允许在仅发送 2 条消息并接收 3 条消息后定位标签。该方案如下图所示。

这代表了消息流量的大量节省，从而节省了电池电量和通话时间。在 DecaRangeRTLS 演示中，锚将计算范围的测距报告发送到标签，以便它也知道范围，这是在下一个响应消息中完成的。这意味着可以在标签一侧使用定位引擎来计算标签相对于锚点的位置（导航模式或地理围栏模式）。

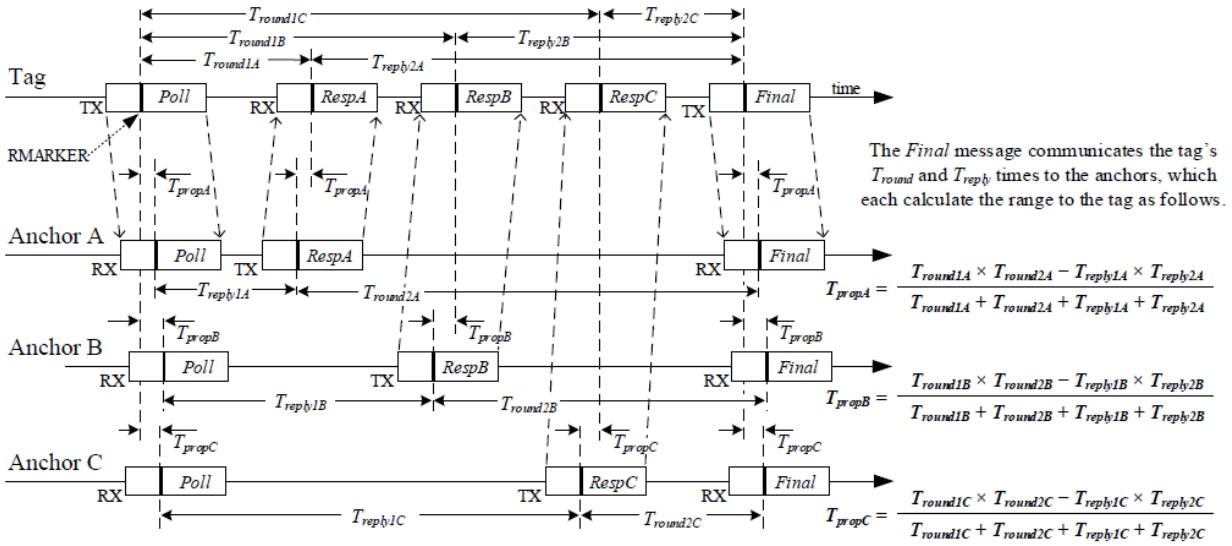

图 5：DecaRangeRTLS 中的范围计算

3.2 实际产品测距的实际考虑

应用笔记 APS016：“*从 TREK 转向（商业 TWR RTLS）产品*”任何想要开发商业 TWR RTLS 产品的人都应该阅读。该文档旨在对从使用 Decawave 的 DW1000 超宽带 (UWB) 收发器 IC 的 Decawave 的 TREK1000 双向测距 (TWR) RTLS IC 评估套件开始开发商业 RTLS 产品所涉及的步骤进行评估和概述。

3.3 帧时间调整

成功的测距依赖于系统能够准确地确定消息离开一个天线并到达另一个天线时的 TX 和 RX 时间。这对于天线到天线的飞行时间测量和由此产生的天线到天线的距离估计是必需的。

导致 TX 和 RX 时间的重要事件在 IEEE 802.15.4 中定义为“测距标记 (RMARKER)：物理层 (PHY) 标头 (PHR) 第一位的第一个超宽带 (UWB) 脉冲测距框架（RFRAME）”。时间戳应反映 RMARKER 离开或到达天线的时刻。但是，标记 RMARKER 生成或接收的是数字硬件，因此需要进行调整以将 TX 天线延迟添加到 TX 时间戳，并从 RX 时间戳中减去 RX 天线延迟。

作为 TREK1000 套件的一部分的 EVB1000 单元已校准天线延迟，并编程到 DW1000 OTP（一次性可编程）存储器中。但是，如果将 DecaRangeRTLS SW 下载到尚未针对 TREK 校准的 EVB1000 上，则应用程序将使用在instance_calib.c文件，有两个值，每个通道选项一个（2/5）**514.83**ns 和**514.65** ns分别。指定的值在 TX 和 RX 天线延迟之间平均分配。默认值已通过调整实验设置，直到报告的距离平均为测量距离。

4M使用的留言TREK1000 行波堆

如图 3 所示，双向测距中使用了三种消息：轮询消息、响应消息和最终消息。消息的详细格式记录如下。这些遵循 IEEE 消息编码约定，但这些是不是 标准化 RTLS 消息。读者可参考 ISO/IEC 24730-62 国际标准，了解在基于 IEEE 802.15.4 UWB 的 RTLS 系统中使用的标准化消息格式的详细信息。

4.1 一般测距帧格式

通用消息格式是数据帧的 IEEE 802.15.4 标准编码。图 6 显示了这种格式。两个字节的帧控制八位字节对于 TREK 应用程序是恒定的，因为它始终使用具有 2 个八位字节（16 位）源地址和目标地址以及单个 16 位 PAN ID（值 0xDECA）的数据帧。在真实的 802.15.4 网络中，PAN ID 可能会作为与网络关联的一部分进行协商，也可能是基于应用程序定义的常量。

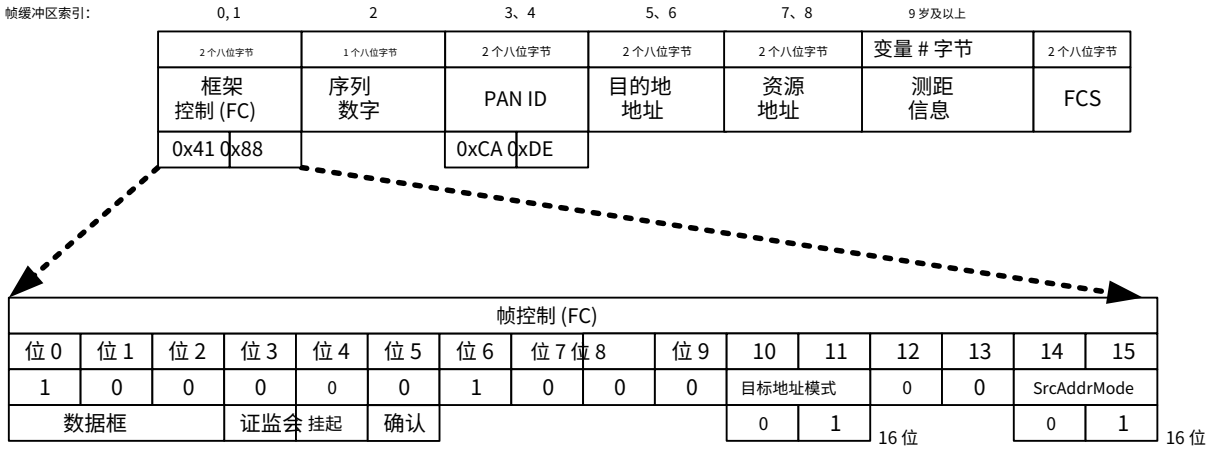

图 6：一般测距帧格式

对于发送的每一帧，序列号八位字节以 256 为模递增。

源地址和目标地址是 16 位值，基于 EVB1000 板的配置开关设置，选择模式为标签或锚点以及标签/锚点编号。

2 字节 FCS 是 CRC 帧校验序列。这由 DW1000 IC（在软件控制下）自动生成并附加到传输的消息中。

帧的测距消息部分的内容取决于它是四个测距消息中的哪一个。这些在图 7 中显示并在第 4.2 至 4.5 节中进行了描述。在这些中，仅显示和讨论了框架的测距消息部分。该数据被封装在图 6 的一般测距帧格式中，以在每种情况下形成完整的测距消息。

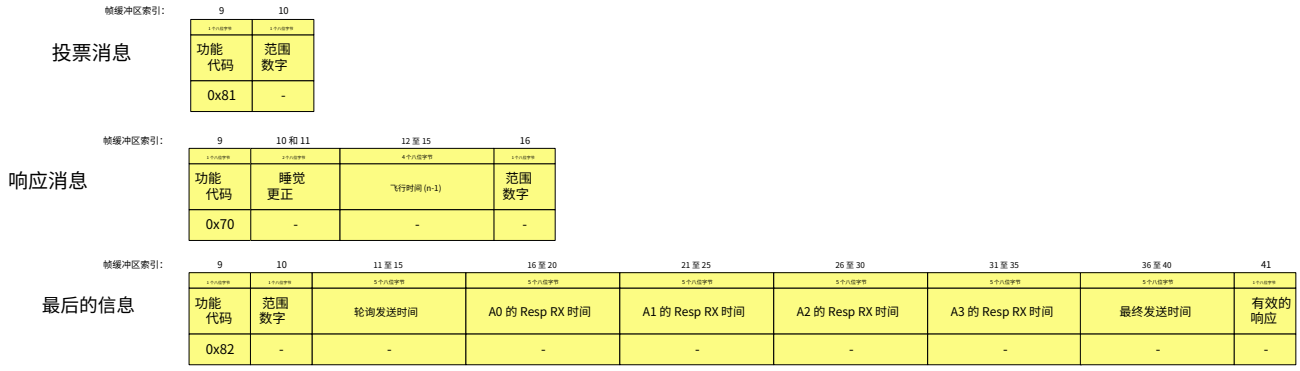

图 7：排列消息编码

4.2 投票消息

轮询消息由标签发送以启动距离测量。表 3 列出并描述了轮询消息中的各个字段。

表 3：测距轮询消息中的字段

八位字节#	价值	描述
1	0x81	功能代码：此八位字节 0x81 将其标识为标记轮询消息
2	-	范围编号：这是一个范围序列号，每次唤醒时，此编号都会递增。

4.3 响应消息

响应消息由 Response 中的锚点发送到来自标签的轮询。表 4 列出并描述了响应消息中的各个字段。

表 4：测距响应消息中的字段

八位字节#	价值	描述
1	0x70	功能代码：此八位字节 0x70 将其标识为响应消息
2、3	-	睡眠校正：这两个八位组参数是调整标签的睡眠持续时间的校正因子，以便标签的测距活动可以分配并对齐到不干扰系统中其他标签的插槽中。锚 # 0，网关锚点，将控制/设置该字段。所有其他锚点将此字段设置为 0。
12 至 15	-	上一次交换的 32 位 TOF，对应于下一个八位字节中给出的范围号
16	-	范围号：这是一个范围序号，对应于上报的 TOF。

4.4 最后的信息

最终消息由标签在收到锚点的响应消息后发送。最终消息的长度为 44 个八位字节。表 5 列出并描述了最终消息中的各个字段。

表 5：测距最终消息中的字段

八位字节#	价值	描述
1	0x82	功能代码：此八位字节将消息标识为标记最终消息
2	-	范围号：这是一个范围序列号，对应于轮询中发送的范围号。
3 到 7	-	Poll TX time：这个 5 字节的字段是标签轮询消息的 TX 时间戳，即帧被传输的精确时间。
8 至 12	-	Resp RX time：这个 5 个字节的字段是来自锚点 0 的响应的 RX 时间戳，即标签从锚点接收到响应帧的时间。
13 至 17	-	Resp RX time：这个 5 个八位字节字段是来自锚点 1 的响应的 RX 时间戳，即标签从锚点接收到响应帧的时间。
18 至 22	-	Resp RX time：这个 5 个八位字节的字段是来自锚点 2 的响应的 RX 时间戳，即标签从锚点接收到响应帧的时间。
23 至 27	-	Resp RX time：这个 5 字节的字段是来自锚点 3 的响应的 RX 时间戳，即标签从锚点接收到响应帧的时间。
28 至 32	-	Final TX time：这个 5 个字节的字段是这个最终消息的 TX 时间戳，即帧被传输的时间，（这是由标签预先计算的）。
33	-	8 位值，指定哪些响应时间有效。如果响应时间显示为有效，则接收最终结果的锚应仅计算 TOF。

除了将 TOF 发送到标签外，每个锚点还通过其 USB 端口报告测距结果。

4.5 消息时序

TREK 演示支持以下模式：

a) 110 kbps 数据速率，1024 个前导码长度和 16 MHz PRF，使用 64 个符号的非标准 SFD

b) 6.81 Mbps 数据速率，128 个前导码长度和 16 MHz PRF，并使用 8 个符号的标准 SFD。

如上部分所示的消息长度（以字节为单位）为：Poll = 13，Response = 19，Final = 44。如果我们取最长的消息，那么 110 kbps 模式的总帧持续时间是 4.929 ms，对于 0.214 ms 6.81 Mbps 模式。

图 8：TWR 时序曲线

在此 TREK 演示中，使用了延迟响应时间，如图 8 所示。响应时间因锚点 ID 而异。它们被保持在最低限度，即锚点将在收到 Poll 或 Response 后尽快尝试回复。周转时间受 SPI 频率和微处理器事件处理时间的限制。这意味着测距交换的总时间在 110 kbps 模式下大约需要 26 ms，在 6.81 Mbps 模式下需要 2.25 ms。

表 6：插槽时序

数据速率	N 个插槽	吨重1(微秒)	吨重2(微秒)	吨re3(微秒)	吨re4(微秒)	吨芬儿(微秒)	吨s分钟（毫秒）
110 kbps	8	2620	5720	8820	11920	16000	26
6.81 Mbps	8	320	658	995	1335	1800	2.25

表 7：帧时序

信息	字节	数据速率	前言（符号）	SFD（符号）	框架 持续时间（微秒）
轮询	13	110 kbps	1024	64	2500
回复	16	110 kbps	1024	64	2894
最后	44	110 kbps	1024	64	4929
轮询	13	6.81 Mbps	128	8	176
回复	16	6.81 Mbps	128	8	182
最后	44	6.81 Mbps	128	8	214

4.6 定位率

TREK1000 支持 2 种定位速率，如表 8 所示。由于使用自动定位，最后两个时隙由锚点到锚点双向测距占用，因此支持的最大可能标签数为 8。时隙时间是从表 6 中所示的时间得出的，并添加了一些保护时间。

表 8：定位率

模式	数据速率	N 个插槽	吨s（小姐）	吨科幻（小姐）	赫兹
一个	6.81 Mbps	10	10	100	10
乙	110 kbps	10	28	280	3.5

5乙建和R联合国宁C颂

5.1 外部库

DecaRangeRTLS ARM 应用程序由 STM 库和 Decawave 应用程序和驱动程序源组成。所有这些都在源代码的 zip 中提供。用户只需要解压源码并打开项目文件*DecaRangeRTLS_ARM.coproj*(如果已经安装了 CooCox IDE，下面的段落解释了如何安装 CooCox IDE）。

5.2 构建代码

作为示例开发环境，可以使用 CooCox IDE 构建此代码。本代码构建指南假定读者已安装 ARM 工具链并熟悉使用 CooCox IDE 构建代码。在 DecaRangeRTLS ARM 软件项目中，我们使用 GNU Tools ARM for Embedded 工具链。用于嵌入式的 GNU 工具 ARM 可以在以下位置找到：https://launchpad.net/gcc-arm-embedded

CooCox IDE 可以从以下位置下载：http://www.coocox.org/software.html . 请点击“阅读更多”链接并下载 1.7.8 版本。发布的代码是使用版本 1.7.8 构建的。

5.3 构建配置选项

示例应用程序有几个不同的构建配置选项可以使用（参见 实例.h和端口.h更多细节）：

定义沉睡 (0)//要在标签中禁用深度睡眠，请将其设置为 0

定义CORRECT_RANGE_BIAS (1)// 要消除由于近距离高功率时不均匀的蓄电池增长而导致的小偏置补偿，请将其设置为 0

定义祖先 (1)// 要禁用锚点到锚点的范围，这应该设置为 0

定义USB_支持// 要禁用 USB 虚拟 COM 端口功能，应删除此行

定义LCD_UPDATE_ON (1)// 要在测距期间停止更新 LCD，应将其设置为 0

6 欧执行流程

6.1 简介

本节旨在指导软件在运行时的执行流程，阅读本文并通过查看代码同时遵循它应该让读者很好地理解软件作为控制运行的基本方式流经层层实现发送和接收。这种理解应该有助于将测距功能集成/移植到其他平台。

为了有效地使用它，鼓励读者在阅读本描述的同时浏览源代码（例如在 eclipse IDE 中），并在源代码中找到每个引用的项目并按照此处描述的流程进行操作。

6.2 主应用入口

应用程序被初始化并从*主要的（）*. 首先，我们初始化硬件和各种 ARM 微控制器外设，*外围设备初始化（）*和*spi_peripheral_init()*用于此的功能也初始化 LCD（*初始化液晶显示器（）*）。然后实例角色（Tag 或 Anchor）和通道配置（通道、PRF、数据速率等）通过调用*初始化测试应用程序（）*功能。最后*实例运行（）*定期从*而（1）*运行下面描述的实例状态机的循环。并行启用 DW1000 中断线，因此任何事件（例如发送帧或接收帧）都在*dwt_isr()*称呼。

如果有新的测距计算或收到测距报告（*实例新范围（）*），应用程序将准备输出缓冲区以通过虚拟 COM 端口发送，并更新 LCD 显示。

这*usb_run()*也被称为*而（1）*循环，它处理通过 USB/虚拟 COM 端口发送到应用程序的任何数据，并输出 tx_buff[] 中存在的任何数据。

6.3 实例状态机

实例状态机提供距离测量的主要 DecaRangeRTLS 功能。实例状态机通过形成用于传输 (TX) 的消息、命令它们的传输、通过命令接收 (RX) 活动、通过记录 TX 和 RX 时间戳、通过从收到的*最后*消息，并通过执行飞行时间计算。

使用函数调用实例代码*测试应用程序（）*，下面的段落跟踪此实例状态机从初始化到测距交换的 TX 和 RX 操作的执行流程。这主要是通过查看标签端的操作来完成的。它首先发送一个*轮询*消息，等待*回复*(最多 4 个*回应*可以接收），然后发送*最后*消息完成测距交换。Anchor 计算出来的 ToF 会在下一个发送*回复*信息。

锚转换在这里没有详细讨论，但是在阅读了下面标签执行流程的描述之后，阅读器应该能够很好地遵循锚执行流程。

这*实例运行（）* function 是实例的主要功能；它应该定期运行或作为未决中断的结果运行。它检查是否有任何未完成的事件（一旦发生中断

*instance_txcallback()*或者*instance_rxcallback()*被调用并采取行动（它将安排响应或重新启用接收器），然后处理 DW1000 TX 或 RX 事件并在事件队列中排队）需要处理并调用*测试应用程序（）*处理它们的功能。它还检查是否有任何计时器已过期（例如睡眠计时器）。下面的段落描述了*测试应用程序（）*状态机详细介绍：

6.3.1 初始状态：TA_INIT

功能*测试应用程序（）*包含实现双向测距功能的状态机，执行的代码部分取决于状态并由“转变(inst->testAppState)”函数开头的语句。初始状态“案子TA_INIT”[1]执行初始化并根据“安装->模式”正在选择 Tag、Anchor 或 Listener 操作。让我们假设它是一个标签并跟随下一个状态的执行。在标签的情况下，我们想要进入睡眠状态，之后我们将启动测距交换，因此状态“inst->testAppState”改为

“TA_TXPOLL_WAIT_SEND”和 ”inst->instToSleep”设置为 TRUE。

6.3.2 状态：TA_SLEEP_DONE

在这种状态下，一旦睡眠超时到期，微处理器就会将 DW1000 从深度睡眠中唤醒。唤醒任何未保存的 DW1000 寄存器后，将重新编程和状态
将更改为inst->testAppState = inst->nextState;这将是“TA_TXPOLL_WAIT_SEND”。

注意：为了最小化功耗，微处理器使用 DW1000 的 RSTn 引脚来通知 DW1000 在唤醒后进入 INIT 模式并准备好运行。这最大限度地减少了微处理器在轮询检查 DW1000 是否已进入 INIT 状态之前等待的时间。在通过 SPI 读取或写入之前，micro 需要确保 DW1000 处于 IDLE 状态，IDLE 的时间将花费 35 us。在这里，我们进行 4 次虚拟读取以确保 DW1000 在写入之前处于空闲状态。

6.3.3 状态：TA_TXPOLL_WAIT_SEND

在该州 ” 案子TA_TXPOLL_WAIT_SEND”,我们想发送*轮询*消息，所以首先我们设置目标地址和所有其他参数/字节*轮询*信息。这*轮询*message 是一个广播消息，因为目标地址设置为 0xffff。

这*测试应用程序（）*状态机状态设置为“TA_TX_WAIT_CONF”,由于该状态不止一种用途，
“inst->previousState = TA_TXPOLL_WAIT_SEND”设置为控制变量。

注意：如果标签发送*轮询*消息，此消息会立即发送（通过调用 *dwt_starttx()*和标签*最后*使用延迟发送命令发送消息（状态“案子 TA_TXFINAL_WAIT_SEND”如下面第 6.3.8 节所述），需要在相对于请求响应的消息到达的准确和特定时间发送消息。为此，我们使用延迟发送。这是由“延迟发送”函数的第二个参数*实例enddlypacket()*.

我们还配置并启用了 RX 帧等待超时，这样如果没有响应，标签就会超时并重新开始测距。接收器也会自动打开（DWT_RESPONSE_EXPECTED）

[1] “TA_”前缀是因为这些是“测试应用程序”中的状态。

延迟，这是因为*回复*预计在一段时间后*轮询*传输完成，过早打开接收器只会浪费功率。

6.3.4 状态：TA_TXE_WAIT

这是在下一次测距交换开始之前调用的标签的状态（即在发送下一个*轮询*信息）。这里我们检查标签是否需要在下一个之前进入 DEEP_SLEEP 模式*轮询*发送，并调用*dwt_entersleep()*如果需要睡眠。

注意：为了节省 TWR RTLS 系统中标签的电量，标签将使用多个锚点进行轮询和测距，然后进入睡眠模式，然后再次开始该过程。

6.3.5 状态：TA_TX_WAIT_CONF

在该州 ” 案子TA_TX_WAIT_CONF”,我们等待确认消息传输已完成。当 IC 完成传输时，设备驱动程序中断程序会拾取“TX 完成”状态位，该程序会生成一个事件，然后由 TX 回调函数处理

(*instance_txcallback()*) 。实例在确认传输成功后，将读取并保存 TX 时间，然后进入下一个状态 (TA_RXE_WAIT）。它只会打开接收器，如果inst->wait4ack
未设置并等待响应消息。因此设置了下一个状态“inst->testAppState = TA_RXE_WAIT”。 有关其作用的详细信息，请参见下面的 6.3.6。

6.3.6 状态：TA_RXE_WAIT

这是预接收器启用状态。此处启用了接收器，然后实例将继续
TA_RX_WAIT_DATA它将等待处理任何收到的消息或将超时。由于接收器将自动打开（如我们 DWT_RESPONSE_EXPECTED设置为 TX 命令的一部分），状态更改为TA_RX_WAIT_DATA等待来自锚点的预期响应消息或超时。我们使用接收器的自动延迟打开，因为我们知道发送响应的确切时间，因为它们使用延迟传输。这有可能（并且对于功率效率而言是可取的）延迟开启接收器，直到恰好在预期响应之前。（延迟 RX 不是 IEEE 标准原语的一部分，而是支持此 DW1000 功能的扩展）。下一个状态是：“inst->testAppState =

TA_RXE_WAIT_DATA”。

注意：如果延迟传输失败，收发器将被禁用，然后接收器将在此状态下正常启用。

6.3.7 状态：TA_RX_WAIT_DATA

国家“案子TA_RX_WAIT_DATA”很长，因为它处理所有预期的 RX 消息。这不是非常健壮的行为。标签实际上应该只寻找来自锚点的消息，（反之亦然）。我们 ” 转变（信息）”，并根据接收到的事件处理消息到达。如果接收到一个好的帧（SIG_RX_OKAY),我们首先检查使用哪种寻址模式（短或长或两者），然后我们查看 MAC 有效负载数据的第一个字节（超出 IEEE MAC 帧头字节）和“转变(rxmsg->messageData[FCODE])”。FCODE 是 Decawave 为不同的 DecaRanging 消息定义的标识符；有关详细信息，请参见图 7。

从这里讨论的角度来看，标签正在等待锚点的响应消息，因此我们希望 FCODE 匹配“RTLS_DEMO_MSG_ANCH_RESP”。在这段代码中，我们注意到消息的 RX 时间戳“锚点响应时间”并计算“延迟回复时间”这是我们应该发送*最后*消息完成测距交换。在这种情况下，我们的下一个（和后续状态）设置为：

```
inst->testAppState = TA_TXFINAL_WAIT_SEND ;// 然后发送最终响应
```

国家“案子TA_RX_WAIT_DATA”还包括处理“SIG_RX_TIMEOUT”消息，用于预期消息未到达且 DW1000 触发帧等待超时事件的情况。DW1000 具有 RX 超时功能，允许主机等待 I2C 发出数据消息中断或无数据超时中断信号[2]. 当超时发生时，标签将返回以重新启动测距交换。

```
inst->testAppState = TA_TXE_WAIT ;// 在下一次测距之前检查是否应该进入睡眠状态 inst->nextState =
TA_TXPOLL_WAIT_SEND ； // 然后发送投票
```

6.3.8 状态：TA_TXFINAL_WAIT_SEND

在该州 ” 案子TA_TXFINAL_WAIT_SEND”,我们想发送*最后*信息。

这*最后*消息包括嵌入标签轮询消息的 TX 时间戳“inst->tagPollTxTime” 连同锚点响应消息的 4 个 RX 时间戳，以及最终消息的嵌入式预测（计算）TX 时间戳，其中包括添加天线延迟“inst->txantennaDelay”。显示哪些响应时间有效的掩码也插入到*最后*信息。

最终消息在请求响应的消息到达的特定时间发送，这是使用延迟发送完成的，由“延迟发送”函数的第二个参数 *实例enddlypacket().*

我们通过设置控制变量“inst->previousState = TA_TXFINAL_WAIT_SEND”指示我们来自哪里，我们设置“inst->testAppState = TA_TX_WAIT_CONF”选择它作为下一次调用的新状态*测试应用程序（）*状态机。

6.3.9 状态：TA_TX_WAIT_CONF（对于*最后*消息 TX)

在该州 ” 案子TA_TX_WAIT_CONF”, (如上文第 6.3.5 0 节所述）我们等待确认消息传输已完成。这里我们将保存TX时间，然后进入下一个状态 (TA_RXE_WAIT)。它只会打开接收器，如果inst->wait4ack未设置并等待响应消息。
因此设置了下一个状态“inst->testAppState = TA_RXE_WAIT”。

6.3.10 结论

以上应该足以让阅读器能够破译锚活动（以及标签的任何剩余活动）的状态机演练。

[2]这里的这个想法（尽管尚未为此编写代码）是为了促进主机处理器进入低功耗状态，直到被 RX 数据到达或无数据超时唤醒。

总之，主播无限期地等待状态“案子TA_RX_WAIT_DATA”直到它收到一个*轮询*信息。一旦它收到轮询，它就会开始测距交换并完成 TOF（范围）报告的计算，它报告给 LCD/USB，并发送回标签中的标签*回复*信息。

锚 ID 0（第 2.4.2 节操作模式 - 锚）还将计算正确的睡眠延迟校正以发送回标签，以便下一次测距交换在正确的超帧时隙中开始。

7乙参考书目

参考	作者	日期	版本	标题
[1]	十波		当前的	DW1000 数据表
[2]	十波		当前的	DW1000 用户手册
[3]	十波		当前的	TREK1000 用户手册
[4]	IEEE		2011	IEEE 802.15.4-2011 或“IEEE Std 802.15.4™-2011”（IEEE Std 802.15.4-2006 修订版）。局域网和城域网的 IEEE 标准 - 第 15.4 部分：低速率无线个域网 (LR-WPAN)。由 LAN/MAN 标准委员会赞助的 IEEE 计算机协会。 可从http://standards.ieee.org/

8天文件H历史

表 9：文档历史

修订	日期	描述
1.1	31英石2015 年 3 月	生产设备的初始版本。
2.0	30th2015 年 9 月	预定更新
2.1	30th2015 年 10 月	更新以包括对时隙/超帧时序的更改

9 米亚约C挂起

9.1 发布 2.0

页	变更说明
全部	将版本号更新为 2.0
全部	各种排版变化
第 2、3、4 和 6 节	将文档更新为新的 TWR 方案（非对称、交错）

9.2 2.1 版

页	变更说明
全部	将版本号更新为 2.1
全部	各种排版变化
第 2.4.1 节	更新了 6.81 Mbps 速率的时隙/超帧时序（现在使用 10 个 10 ms 的时隙），10 Hz 的定位速率保持不变。

10安回合D电波

Decawave 是一家开创性的无晶圆半导体公司，其旗舰产品 DW1000 是一款基于 IEEE 802.15.4 标准 UWB PHY 的完整单芯片 CMOS 超宽带 IC。该设备是一系列部件中的第一个。

由此产生的硅片在制造、医疗保健、照明、安全、运输以及库存和供应链管理等领域具有广泛的基于标准的应用，适用于实时定位系统 (RTLS) 和超低功耗无线收发器。

有关此产品或任何其他 Decawave 产品的更多信息，请按以下方式联系销售代表：-

迪卡波有限公司，
阿德莱德钱伯斯，
彼得街，
都柏林 8,
爱尔兰。

邮寄地址：sales@decawave.com
http://www.decawave.com/