编 号	HR_SJ_QRSW4
密 级	内部公开
阶 段	发布
页 数	18

代 号

名 称 **HR-RTLS4 嵌入式软件开发手册**

基于 TDOA 的高精度 UWB 定位开发套件

会 签

大连浩如科技有限公司

文档控制

变更记录

版本号	日期	增加/修改/删除	简单描述	更改申请单号
V1.0	20240421	创建	根据模板创建，描述嵌入式软件方案和流程。	

目 录

1 系统概况

本嵌入式开发手册适用于浩如科技 HR-RTLS4 系列 UWB 定位模块，HR-RTLS4 采用 TDOA 定位方式，嵌入式软件基于 DecaWave 官方 API 开发，分基站和标签两部分，基站部分主要完成标签广播数据时间戳接收、基站之间时间同步消息的发送接收等，标签部分主要完成定位请求广播消息发送和时隙分配校准防冲突功能。系统基于 DecaWave 公司 DW1000 与 DW3000 系列（以下统称为 DWIC）的 UWB 芯片，代码结构清晰、维护简单并方便移植。

ULM1-SH、ULM1-GP、ULM3-SH 模块采用 STM32L051 低功耗单片机；其他模块均采用 STM32F103 单片机或其完全兼容替代的国产 GD 型号，使用 CUBEmx 初始化配置，HAL 库函数开发，可迅速方便移植到其他型号单片机。

在阅读本手册之前，建议先观看开箱测试视频和阅读对应产品型号的用户手册，以便初步了解系统功能和使用方法。本手册也可以结合《课程 2：TDOA 嵌入式软件流程》视频教程进行同步学习。

2 嵌入式软件整体架构

嵌入式软件整体架构如图 2-1 所示：主要分为驱动层、DW_API 层、应用层。

驱动层主要实现 STM32 与 DWIC 的 SPI 通信，一般使用 CUBEmx 进行初始化配置，使用 HAL 库进行开发，初始化完成后，自动生成初始化代码，使用 SPI 数据收发函数，对接到 DW_API 层，完成驱动层搭建。另外还有中断配置、IIC 配置、串口配置、看门狗配置等都在 CUBEmx 进行初始化配置，驱动层代码用户开发工作量较小，主要熟练掌握 CUBEmx 配置即可。

DW_API 层使用 DecaWave 官方 API 进行移植搭建，API 将常用功能进行函数封装，DWIC 的主要功能实现基本通过读写相应的寄存器实现，通过官方 API 内的简单 example 结合《DW1000_Software_API_Guide.pdf》可大概了解常用 API 功能。DW_API 层代码用户开发工作量较小，开发相应应用层功能时会调用即可。主要程序目录为 Src\decadriver，Src\platform。

应用层是实现 TDOA 嵌入式部分的具体操作流程，主要完成标签广播定位请求、基站同步等功能，并将测得数据通过串口发送。应用层由浩如科技开发，

是实现整体功能的核心代码，建议重点学习和熟练掌握。二次开发基本也是基于应用层来开发。主要程序目录为 Src\application。

应用层	主要文件 dw_main.c instance_anchor.c instance_tag.c	主要功能 实现基站间同步，标签发送定位请求，时间戳读取，标签时序校准，串口通信等功能
DW_API层	主要文件 deca_device.c port.c deca_spi.c	主要功能 使用官方API库，通过SPI控制寄存器读写完成参数配置、数据收发等功能，功能封装成相应函数，供用户调用使用
驱动层	主要文件 STM32F1xx_HAL_Driver spi.c i2c.c	主要功能 完成SPI、IIC、GPIO、看门狗、时钟、外部中断等相应配置，使用CUBEmx生成初始化代码

图 2-1 嵌入式软件整体架构

3 TDOA 无线时钟同步原理

3.1 基本结构

TDOA 定位算法又称到达时间差定位，顾名思义，就是利用无线信号到达不同基站的时间差确定目标标签的位置。设标签信号到达不同基站的时间差为 Δt，乘上信号传输的速度 v，就可以求出无线信号到达不同基站的路程差 Δd。给定两个基站，如果知道标签到两个基站的里程差为 Δd，就可以推算出标签可能在以两个基站为焦点的双曲线上。在此基础上，我们就可以利用几何知识（即双曲线）求解出目标的位置。具体细节可参考《课程 1：TDOA 定位原理》详细了解。

时间同步分有线时间同步和无线时间同步两种，有线时间同步精度高，无线时间同步精度略低，因有线时间同步需要将基站之间全部通过有线连接到时间同步器，对安装部署来讲难度较大，HR-RTLS4 采用无线时钟同步方案进行基站之间的时钟同步。

系统分 1 个主基站(A0)、3 个从基站(A1、A2、A3)和若干个定位标签构成，主基站负责无线广播时间同步消息用于基站间的时间同步。标签定期广播定位请求，各基站将接收的标签广播时间戳和同步消息时间戳上报给上位机软件进行 TDOA 值计算，并通过定位算法进一步解算标签坐标。

3.2 标签广播定位请求消息

标签按一定的频率定期广播定位请求消息，所有基站都处于接收状态可接收到请求信息。主从基站接收到标签请求信息后，分别记录接收时间戳为 tart0，tart1，tart2 和 tart3。

3.3 主基站广播时间同步消息

主基站在接收到标签的广播消息后，会广播时间同步消息，此时所有从基站都处于接收状态，可以接收到基站的时间同步消息，主基站发送时间同步消息的时间戳记为 tas0，时间同步消息内同时携带标签 SLOT 校准消息，标签可在发送定位广播请求消息后，接收 SLOT 校准消息以校准下次请求时间，防止多标签冲突；关于标签 SLOT 定义和操作方法，将在第 5 节详细描述。

从基站接收到同步消息后，记录时间戳分别为 tara1，tara2 和 tara3，同时从基站分别记录于主基站的频偏值 ferr(1-0)，ferr(2-0)，ferr(3-0)。

3.4 TDOA 计算

因从基站和主基站之间存在固定距离，所以时间同步时也要考虑主基站到从基站的飞行时间，这也是无线时间同步比有线时间同步精度略低的主要误差来源之一。主基站与各个从基站飞行时间分别记为 ttof(1-0)，ttof(2-0)，ttof(3-0)。

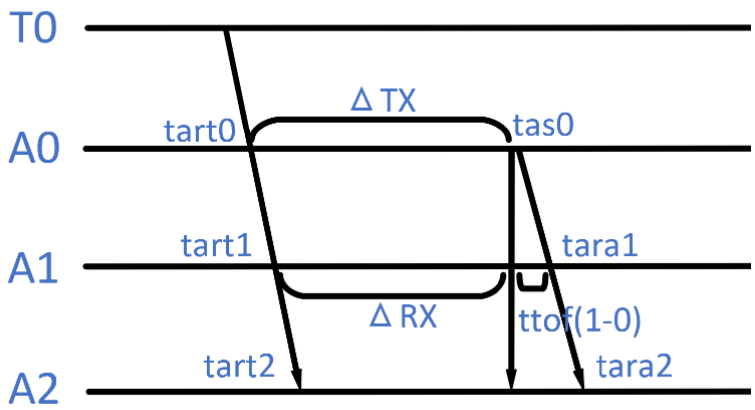

图 3-1 TDOA 无线同步时序图

TDOA 到达时间差计算公式如下：

*ttdoa(1-0) = ΔRX - ΔTX = (tara1 – tart1 - ttof(1-0)) – (tas0 – tart0) * (1 - ferr(1-0))*

- tara1：从基站 1 接收到主基站广播的时间同步消息的时间戳
- tart1：从基站 1 接收到标签广播定位请求消息的时间戳

- ttof(1-0)：主基站到从基站 1 的飞行时间
- tas0：主基站广播时间同步消息的时间戳
- tart0：主基站接收到标签广播定位请求消息的时间戳
- ferr(1-0)：主基站与从基站 1 之间的频偏值

*ttdoa(2-0) = (tara2 – tart2 - ttof(2-0)) – (tas0 – tart0) * (1 - ferr(2-0))*

*ttdoa(3-0) = (tara3 – tart3 - ttof(3-0)) – (tas0 – tart0) * (1 - ferr(3-0))*

3.5 天线延时

实际硬件应用中，因天线型号、射频馈线长度等因素，主基站 A0 的天线延时参数需在公式中考虑进去，以消除天线射频馈线传输延时对时间戳计算造成的误差。设主基站 A0 的发射和接收天线延时参数一致均为 tant0，在主基站接收标签广播信号读取时间戳 tas0 后加上接收天线延时 tant0，在主基站发送同步消息时间戳减去发射天线延时参数 tant0，则 ttdoa(1-0)公式进一步如下：

*ttdoa(1-0) = (tara1 – tart1 - ttof(1-0)) – ((tas0 + tant0) – (tart0 – tant0)) * (1 - ferr(1-0))*

同理从基站 A2、A3。

4 数据帧结构

4.1 通信数据帧基本结构

通信数据遵循 IEEE 802.15.4 协议 MAC 层帧格式，如表 4-1 所示，数据帧包含帧头 MAC Header (MHR)、负载 MAC Payload 和帧尾 MAC Footer (MFR)三部分。帧头部分由帧控制字节、帧序列号和地址等信息构成；负载部分长度可变，可用户自定义；帧尾部分是帧头和负载数据的 16 位 CRC（FCS）校验序列，一般由 DWIC 自动生成。

表 4-1 通信数据帧格式

2 字节 0-1	1 字节 2	2 字节 3-4	2 字节 5-6	2 字节 7-8	可变字节 9+	2 字节 +
Frame Control (FC)	Sequence Number	PAN ID	Destination Address	Source Address	Tdoa Message	FCS
MHR 帧头					MAC 负载	MFR 帧尾

4.1.1 Frame Control

表 4-2 Frame Control 格式

Frame Control (FC)															
Bit0	Bit1	Bit2	Bit3	Bit4	Bit5	Bit6	Bit7	Bit8	Bi9	Bit10	Bit11	Bit12	Bit13	Bit14	Bit15
1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1
Frame Type			SEC	PEND	ACK	PIC	Reserved			DestAddrMode		Frame Version		SrcAddrMode	

Frame Control 是 16bit 寄存器，其中 Frame Type 设置为 001，表示 Data 数据，PIC 设置为 1 表示 PAN ID 开启，DestAddrMode 设置为 10，表示源地址（本机地址）采用 16bit 设备短地址，SrcAddrMode 表示目标地址（接收方地址）采用 16bit 设备短地址。TWR 通信中，Frame Control 设置为 0x8841。

表 4-3 Frame Type 含义

Frame Type Field (FC bits 2 to 0)	Frame
0, 0, 0	Beacon
0, 0, 1	Data
0, 1, 0	Acknowledgement
0, 1, 1	MAC command
1, 0, 0	Reserved
1, 0, 1	Reserved
1, 1, 0	Reserved
1, 1, 1	Reserved

表 4-4 DestAddrMode 含义

Destination addressing mode (FC bits 11 & 10)	Meaning
0, 0	No destination address or destination PAN ID is present in the frame
0, 1	Reserved
1, 0	The destination address field is a short (16-bit) address.
1, 1	The destination address field is an extended (64-bit) address.

表 4-5 SrcAddrMode 含义

Source addressing mode (FC bits 15 & 14)	Meaning
0, 0	No source address or source PAN ID is present in the frame
0, 1	Reserved
1, 0	The source address field is a short (16-bit) address.
1, 1	The source address field is an extended (64-bit) address.

4.1.2 Sequence Number

帧序列号，每帧数据自增 1。

4.1.3 PAN ID

网络 ID，收发数据的设备需设置同一个 PAN ID 下才能正常收发，否则将产生帧拒绝中断。

4.1.4 Destination Address

目标设备地址，在帧过滤模式下，接收方设备地址和发送方目标地址相同时，产生接收正确中断，否则产生帧拒绝中断。

4.1.5 Source Address

本机设备地址。

4.1.6 FCS

Frame Check Sequence，简称（FCS），数据校验，由 DWIC 自动计算完成。

4.1.7 Tdoa Message

4.1.7.1 Poll Message

1字节 9	1字节 10	1字节 11	1字节 12
Function Code	Range Number	SOS 标签报警上传	BATTERY 电池电量
0x21	0-255	1/0	0-100%

4.1.7.2 Sync Message

1字节 9	1字节 10	2字节 11-12
Function Code	Range Number	Sleep Correction
0x10	0-255	-

5 标签时序管理

HR-RTLS4 系统内默认由 A0 基站进行标签时序管理，A0 基站按系统内标签最大容量分配若干个 SLOT，SLOT 译为“时间槽”或“时隙”，每个 SLOT 内预留一定的保护时间（guard time），用于接收其他标签的定位请求，A0 基站监控每个标签发起测距请求的时间，并通过 resp 消息内的 Sleep Correction 将标签按 ID 分配到相应 SLOT 内，使标签有序工作，如图 5-1 所示。

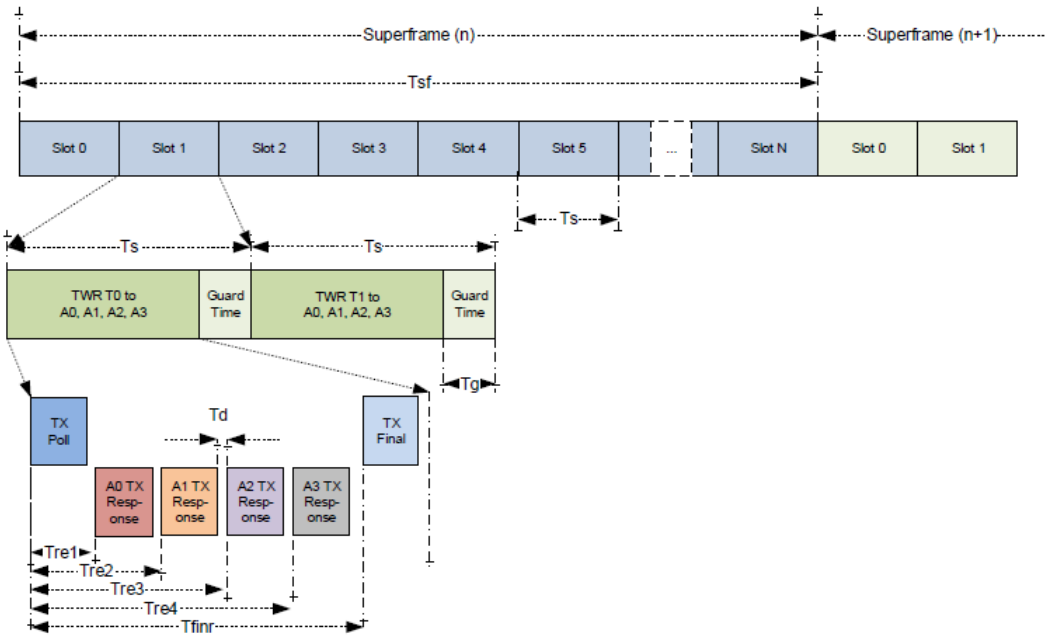

图 5-1 SLOT 分配

6 嵌入式软件流程

6.1 主程序工作流程

主程序位于 Src/application/dw_main.c，主要完成设备参数初始化和按拨码开关状态进行基站或标签状态机运行，最后进行串口数据打包发送。HR-RTLS4 嵌入式软件将基站和标签维护在同一套工程内，通过上电读取拨码开关来判断执行

相应的角色，大大减小了系统维护的复杂度，仅需一套程序可以适配所有的基站和标签模块。主程序工作流程图如图 6-1 所示。

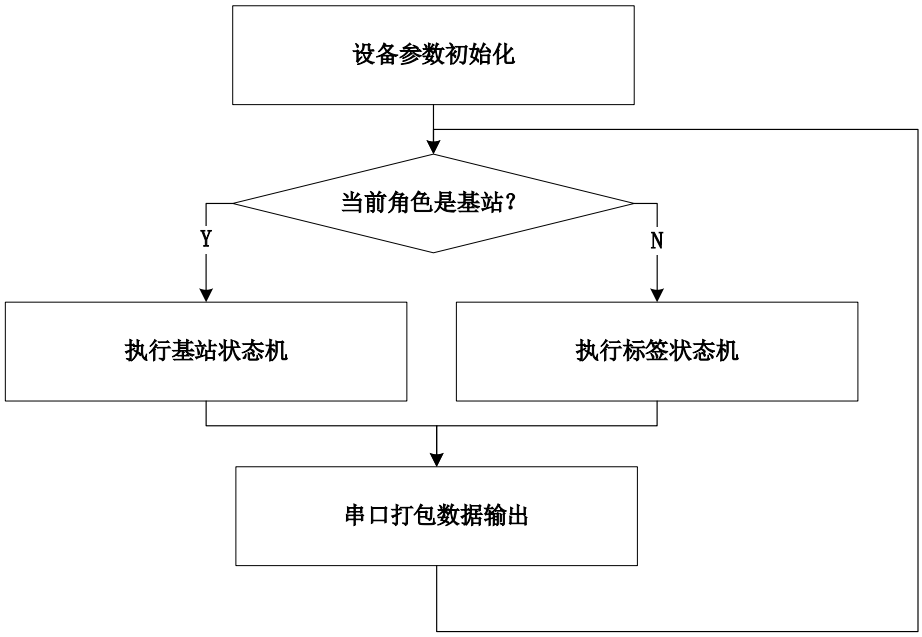

图 6-1 主程序工作流程图

6.2 标签工作流程

标签部分代码位于 Src/application/instance_tag.c，由状态机控制整个工作流程。主要工作流程如图 6-2 所示，其中以蓝色 STA 开头的是程序内状态机标识符，建议结合源码进行同步学习。

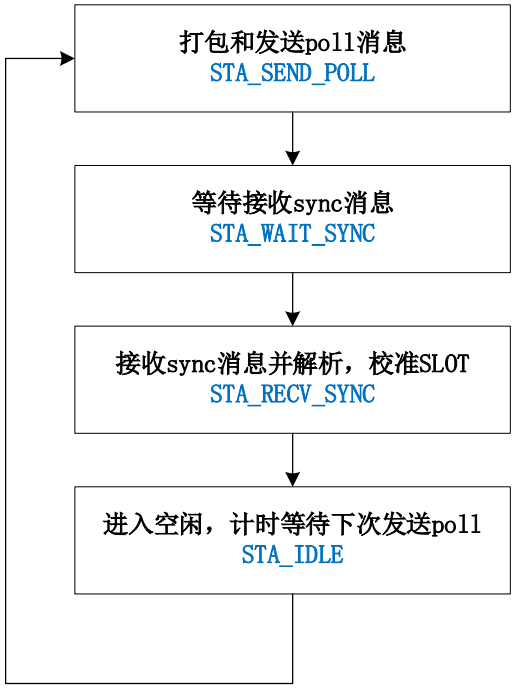

图 6-2 标签工作流程图

6.3 基站工作流程

基站部分代码位于 Src/application/instance_anchor.c，由状态机控制整个工作流程。主要工作流程如图 6-3 所示，其中以蓝色 STA 开头的是程序内状态机标识符，建议结合源码进行同步学习。

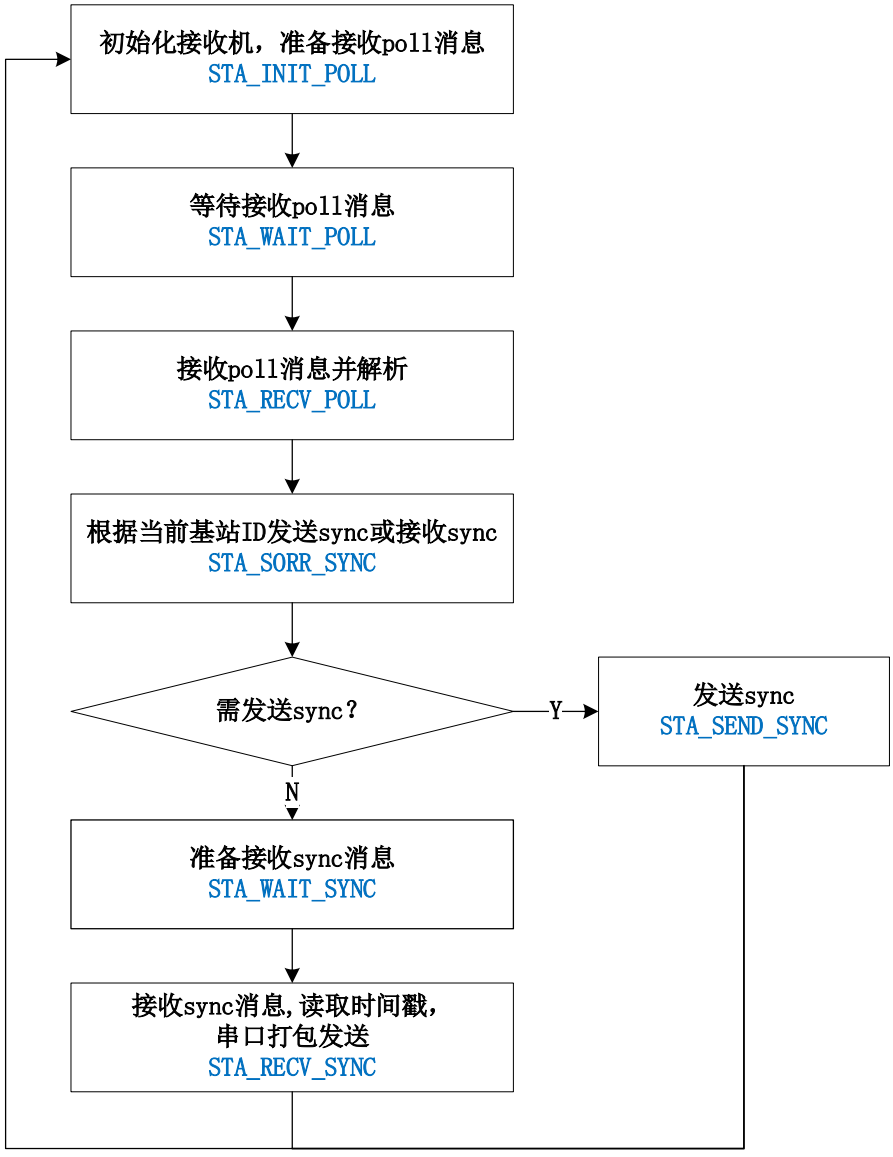

图 6-3 基站工作流程图

7 开发环境说明

使用 CUBEmx 配置系统参数和底层参数，随代码工程提供 CUBE 配置 IOC 文件，如只对 UWB 模块进行应用层二次开发，基本不用修改 CUBEmx 配置，CUBEmx 属可视化操作，较为简单，此处不再赘述。开发 IDE 使用 KEIL-MDK，

我公司开发时使用的版本为 V5.25.2，该版本自带 ARM CompilerV6.9，理论上 V5.25.2 版本以上自带 ARM CompilerV6.9 以上版本都可向下兼容，如编译和烧录出厂程序后出错，请降级至 ARM CompilerV6.9 使用。

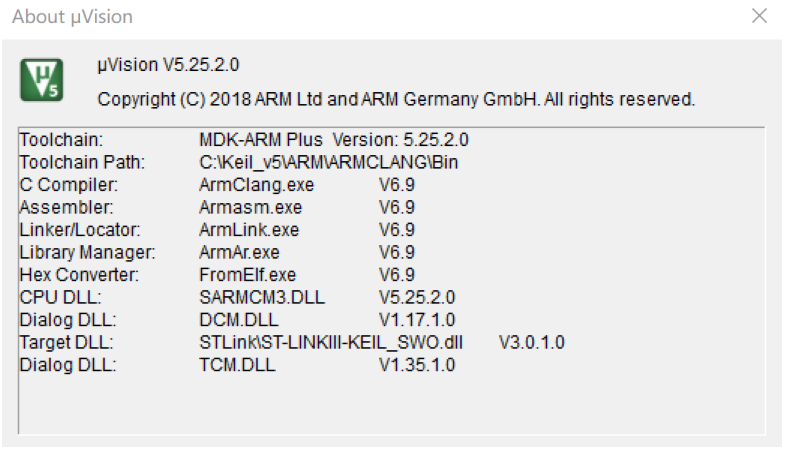

图 7-1 KEIL-MDK 版本

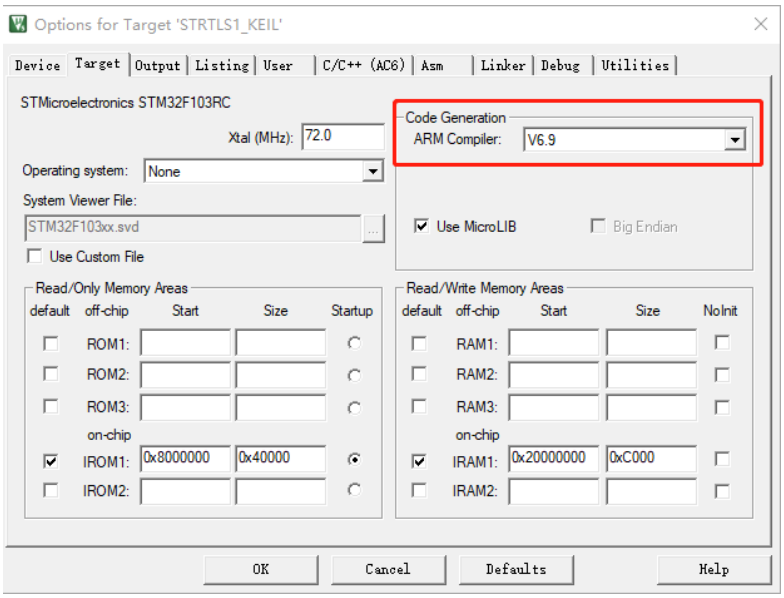

图 7-2 ARM Compiler 版本

另需注意工程配置 Options for Target 内的 c/c++内按如下配置：

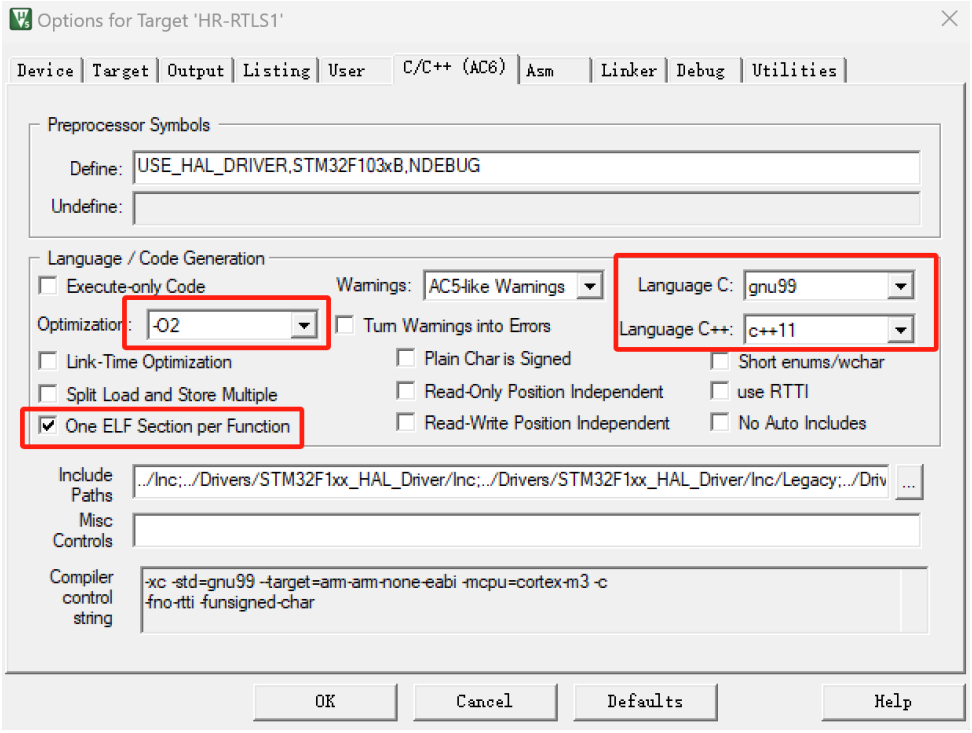

图 7-3 Options for Target 配置

8 串口通信协议

HR-RTLS4 系列基站串口数据可通过板载 USB 接口连接上位机通过串口助手接收查看串口数据，同时通过板载 ESP8266 模块连接 wifi 并通过以太网 UDP 方式透传到主机，格式与串口一致。

串口波特率 115200bps，8 位数据，无奇偶校验，1 位停止位，字符串 ASSCII 格式，相邻字段用逗号隔开。

例：mt,0,1,33,10872698843,10948064768,0,0,-78.16,-82.27,144

mt,1,2,33,610103618676,0,610179018053,-148,-78.27,-85.11,176

表 8-1 串口通信协议说明

内容	例子	功能
HEAD	mt	消息头，固定为 mt
ANCID	0	该条消息的基站 ID
TAGID	1	接收到定位请求的标签 ID
RN	33	Range number 不断累积增加范围 0~255
POLL_RX_TIME	10872698843	该基站收到标签发送的 POLL 消息时间戳
SYNC_TX_TIME	10948064768	主基站发送 SYNC 消息时间戳，只有主基站

		A0 这个字段有效，其他从基站该字段为 0 无效
SYNC_RX_TIME	0	从基站接收主基站 SYNC 消息时间戳，只有从基站 A1,A2,A3…有效，主基站 A0 该字段为 0 无效
CLOCKOFF	0	从基站接收主基站 SYNC 消息频偏，只有从基站 A1,A2,A3…有效，主基站 A0 该字段为 0 无效
RX_PWR	-78.16	该基站接收标签消息的接收总功率，单位 dBm
FP_PWR	-82.27	该基站接收标签消息的第一路径功率，单位 dBm
STD_NOISE	144	该基站接收标签消息的噪声标准差 0~65535

9 常用 API 说明

请结合官方《DW1000/3000_Software_API_Guide.pdf》一起参考学习与使用。

9.1 dwt_writetxdata

int dwt_writetxdata(uint16_t txBufferLength, uint8_t *txBuffer, uint16_t txBufferOffset);

该 API 用于将发送消息数据写入 DWIC 的内部发送缓冲区。

type	name	description
uint16_t	txBufferLength	写入 TX 缓冲区的总长度，它可以包含两字节 CRC 的空间，但不一定要包含。设备将自动在发送的 TX 数据的最后两个字节中放入两字节 CRC。要传输的数据长度由 dwt_writetxfctrl()中的 txFrameLength 参数指定
uint8_t*	txBuffer	发送数据指针
uint16_t	txBufferOffset	开始写入数据的偏移量

9.2 dwt_writetxfctrl

void dwt_writetxfctrl(uint16_t txFrameLength, uint16_t txBufferOffset, uint8_t ranging);

该 API 用于配置 TX 帧控制寄存器。

type	name	description
uint16_t	txFrameLength	帧总长度，需包括 2 个字节的 CRC
uint16_t	txBufferOffset	开始写入数据的偏移量
uint8_t	ranging	如 1 是测距帧，0 是普通数据帧

9.3 dwt_starttx

int dwt_starttx(uint8_t mode);

该 API 用于启动帧数据发送。

type	name	description
uint8_t	mode	DWT_START_TX_IMMEDIATE 立即发送 DWT_START_TX_DELAYED 延时发送，通过 dwt_setdelayedtrxtime()设置发送时间 DWT_RESPONSE_EXPECTED 发送后打开接收机自动启动接收

9.4 dwt_setdelayedtrxtime

void dwt_setdelayedtrxtime (uint32_t starttime);

该 API 设置延迟发送模式中延迟的发送时间或延迟接收模式中接收机开启的时间。在调用 dwt_starttx()或 dwt_rxenable()以延迟为参数时，应调用此函数以设置所需的发送或开启接收机的时间。

type	name	description
uint32_t	starttime	发送或接收开始的时间，参数为系统时间戳的高 32 位，用于发送消息或开启接收器

9.5 dwt_setrxtimeout

void dwt_setrxtimeout (uint32_t time);

该 API 用于在指定时间内未收到任何帧时，将接收器设置为超时。应在 wt_rxenable()之前调用此函数。

type	name	description
uint32_t	time	超时时间以微秒为单位。如果该值为 0，则将禁用超时。最大值为 0xFFFFF。

9.6 dwt_rxenable

int dwt_rxenable(int mode);

此 API 函数用于打开接收器以等待接收数据帧，在延迟接收模式下，接收器开启直到通过 dwt_setdelayedtrxtime()设置的超时时间后结束。

type	name	description
int	mode	DWT_START_RX_IMMEDIATE 立即开启接收机 DWT_START_RX_DELAYED 延迟开启接收机，开始时间在 dwt_setdelayedtrxtime()中设置