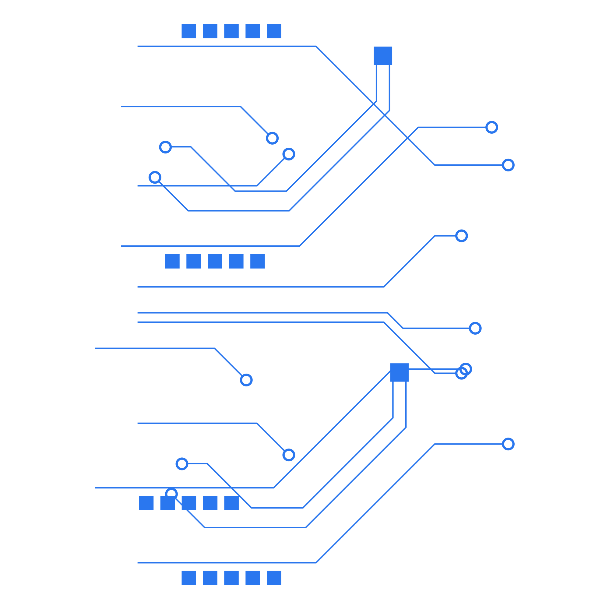


矽典微 | 南京 上海 苏州

[www.iclegend.com](http://www.iclegend.com) | [support@iclegend.com](mailto:info@iclegend.com)

**矽典微雷达驱动使用手册**



**关于本文档**

本文档主要描述矽典微雷达驱动的使用方法，文档内容仅供相关嵌入式驱动工程师，嵌入式算法工程师参考。

**修订历史**

版本变更记录

|  |  |  |
| --- | --- | --- |
| 日期 | 版本号 | 记录 |
| 2021.01.15 | 0.1 | 初版 |
| 2021.03.06 | 0.2 | 支持EVBSN02\_GE |
| 2021.03.25 | 0.3 | 支持XenD101 |
|  |  |  |

**免责说明**

1. 您理解并同意：本产品硬件及相关文档内容按“原样”和“现状”提供，我们不确保本文档中的信息、文本、图形、链接、项目描述的完整性和时效性，准确信息请以实际产品为准。

2. 您购买的产品、服务或特性等应受矽典微公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，矽典微公司对本文档内容不做任何明示或默示的声明或保证。

3. 由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，所有陈述、信息和建议不构成任何明示或暗示的担保。

**目录**

[1 简介 3](#_Toc61525679)

[2 使用方法 4](#_Toc61525680)

[2.1 工程编译 4](#_Toc61525681)

[2.2 系统配置 4](#_Toc61525682)

[2.3 雷达芯片配置 4](#_Toc61525683)

[2.4 算法处理入口 4](#_Toc61525684)

[2.5 其他说明 5](#_Toc61525685)

[3 数据输出格式 6](#_Toc61525686)

[3.1 FFT数据格式 6](#_Toc61525687)

[3.2 DFFT数据格式 6](#_Toc61525688)

[3.3 DFFT PEAK数据格式 7](#_Toc61525689)

[4 RVA算法介绍 8](#_Toc61525690)

[4.1 基于FFT的RVA计算 8](#_Toc61525691)

[4.2 基于DFFT的RVA计算 8](#_Toc61525692)

[4.3 基于DFFT PEAK的RV计算 9](#_Toc61525693)

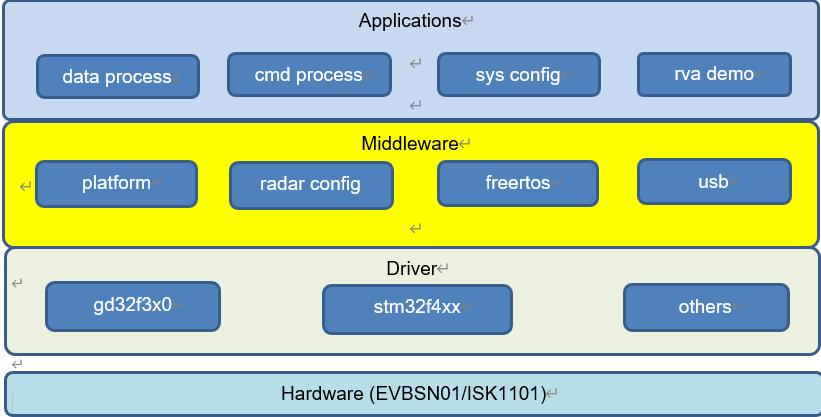
# 简介

ICLM Radar Driver是矽典微基于Keil MDK V5开发的，搭配矽典微硬件开发平台EVBSN01/EVBSN02/ ISK1101/XenD101等使用，默认提供数据透传功能，输出格式支持FFT、DFFT (EVBSN01/EVBSN02平台）、DFFT PEAK三种数据格式。

代码总体分为三层：

1. Applications: 平台无关, 处理具体应用（数据透传，算法开发等）
2. Middleware: 适配不同平台，还包含第三方代码库(FREERTOS, USB等)
3. Driver: MCU平台官方驱动以及外设驱动（EVBSN01/EVBSN02\_GE：STM32F429平台, ISK1101：GD32F350平台，XenD101: MM32F0010平台等）

系统框图如下：



代码顶层目录介绍：

1. App：数据处理模块，命令处理模块，系统配置模块，算法模块
2. Config：全局功能开关，雷达芯片参数配置
3. Doc：说明文档
4. Driver：MCU平台官方驱动模块，外设驱动模块
5. Middle：平台适配模块，雷达芯片配置模块，第三方代码库
6. Project：各硬件平台工程目录
7. Script：脚本相关文件

# 使用方法

## 工程编译

1. EVBSN01工程文件:  Project/EVBSN01/EVBSN01.uvprojx，编译生成文件：Project\EVBSN01\Objects
2. EVBSN02\_GE工程文件:  Project/ EVBSN02\_GE/ EVBSN02\_GE.uvprojx，编译生成文件：Project\ EVBSN02\_GE\Objects
3. ISK1101工程文件: Project/ISK1101/ISK1101.uvprojx，编译生成文件：Project\ISK1101\Objects
4. XenD101工程文件: Project/ISK1101/ISK1101.uvprojx，编译生成文件：Project\XenD101\Objects

## 系统配置

系统配置文件：Config/global\_conf.h

1. 系统功能配置：

SUPPORT\_DATA\_PASSTHROUGH---数据透传模式

关闭SUPPORT\_DATA\_PASSTHROUGH，则可以在MCU处理自己实现的算法

1. 系统参数配置：

UPLOAD\_SAMPLE\_RATE---数据上传到上位机的采样率

RADAR\_DATA\_MAX\_LEN---系统支持的最大数据长度

DEBUG\_MODE\_DEFAULT---系统默认是否开启debug模式

1. 系统debug开关：

CONFIG\_DEBUG---是否编译debug代码

## 雷达芯片配置

雷达芯片配置文件：Config/radarPara/XXX/regs.txt，该文件用来配置相应工程的雷达参数（可以直接拷贝寄存器工具生成的参数），部分文件描述如下：

regs.txt---默认FFT透传模式

DFFT.txt---DFFT透传模式

DFFT\_PEAK.txt---DFFT PEAK透传模式

注意：

Config/radarPara/XXX/regs.txt 中XXX目录必须与工程同名，例如Config/radarPara/EVBSN01/regs.txt，对应Project/EVBSN01工程；

XXX目录下DFFT.txt需要重命名为regs.txt，DFFT配置才可以在对应工程中生效

## 算法处理入口

算法入口函数StartAlgorithm( )位于App/common/src/dataprocess.c，可以在此函数内部做算法的具体实现（需要关闭宏SUPPORT\_DATA\_PASSTHROUGH），函数说明如下：

|  |  |
| --- | --- |
| 函数名 | void StartAlgorithm (uint8\_t\* dataBuf, uint16\_t dataLen, uint8\_t channel, uint16\_t index) |
| 说明 | 驱动解析完数据，会调用该函数做具体算法处理 |
| 参数说明 | * + dataBuf: 驱动解析得到的雷达数据   + dataLen: 数据长度   + channel: 数据对应的通道   + index: 数据的序列号，FFT---chirp index；DFFT---frame count；DFFT PEAK--无意义 |
| 返回值 | 无 |

## 其他说明

1. EVBSN01 LED定义如下：

LED0闪烁：正常收到雷达数据

LED1闪烁：雷达数据index错误

LED2闪烁：雷达数据接收溢出

LED3闪烁：雷达数据发送溢出

LED4闪烁：MCU接收上位机命令溢出，来不及处理命令

所有LED同时亮起，则表示系统不能正常运行，可能的原因：

1. 错误配置导致，此时系统会自动恢复出厂配置，手动重启一下板子即可
2. 某些硬件单元无法正常启动，可以通过UART查看具体错误信息；

正常情况下，应该只有LED0在闪烁

1. EVBSN02\_GE LED定义如下：

LED0闪烁：正常收到雷达数据

LED1闪烁：雷达数据index错误

LED2闪烁：雷达数据接收溢出

所有LED同时亮起，则表示系统不能正常运行，可能的原因：

1. 错误配置导致，此时系统会自动恢复出厂配置，手动重启一下板子即可
2. 某些硬件单元无法正常启动，可以通过UART查看具体错误信息；

正常情况下，应该只有LED0在闪烁

1. ISK1101/XenD101不支持DFFT模式

# 数据输出格式

## FFT数据格式

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DW | Item | 31:24 | 23:22 | 21:20 | 19:16 | 15:11 | 10:9 | 8:6 | 5:0 |
| DW0 | FFT\_HEAD | 0xAA | FFT\_ID | 3 | fft\_chirp\_cnt[8:0] [1] | | 0 | cfg\_fft\_tx\_max[8:0] [2]+1 | |
| **DW** | **Item** | **31:16** | | | | **15:0** | | | | |
| DW1  ：  DWm-1 | FFT\_DATA | fft real data 0[15:0] | | | | fft image data 0 [15:0] | | | | |
| fft real data 1[15:0] | | | | fft image data 1 [15:0] | | | | |
| … | | | | … | | | | |
| fft real data n[15:0] | | | | fft image data n [15:0] | | | | |
| **DW** | **Item** | **31:16** | | | | **15:14** | **13:12** | **11:8** | **7:0** |
| DWm | FFT\_TAIL | check\_sum | | | | 0/1 | 3 | fft\_chirp\_cnt[3:0] | 0x55 |

[1] fft\_chirp\_cnt: chirp number

[2] cfg\_fft\_tx\_max: N points FFT transferred

[3] if MOSI[0], Head[23:22]=0 & Tail[15:14]=0; if MOSI[1], Head[23:22]=1 & Tail[15:14]=1

FFT\_ID: 0 – FFT0, 1 – FFT1

## DFFT数据格式

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DW | Item | 31:24 | 23:22 | 21:20 | 19:16 | 15:11 | | 10:9 | | 8:6 | | 5:0 |
| DW0 | DFFT\_HEAD | 0xAA | DFFT\_ID | 3 | F | dpl\_frame\_cnt[15:0][1] | | | | | | |
| **DW** | | **Item** | **31:16** | | | | **15:0** | | | | | | |
| DW1  ：  DWm-1 | | DFFT\_DATA | dfft real data 0[15:0] | | | | dfft image data 0 [15:0] | | | | | | |
| dfft real data 1[15:0] | | | | dfft image data 1 [15:0] | | | | | | |
| … | | | | … | | | | | | |
| dfft real data N[15:0] | | | | dfft image data N [15:0] | | | | | | |
| **DW** | | **Item** | **31:16** | | | | **15:14** | **13:12** | | **11:8** | | **7:0** | |
| DWm | | DFFT\_TAIL | check\_sum[2] | | | | 0x5555 | | | | | | |

[1] dpl\_frame\_cnt: doppler frame number

[2] check\_sum: sum of all data，which is used for redundancy check optionally

DFFT\_ID : 2’b10 – DFFT0 2’b11- DFFT1

## DFFT PEAK数据格式

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DW | Item | 31:24 | 23:22 | 21:20 | 19:16 | 15:11 | 10:9 | 8:6 | 5:0[1] |
| DW0 | DFFT\_PEAK\_HEAD | 0xAA | 1 | 0 | 0 | 0 | | | cfg\_len\_rpt[4:0]+1 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Data[2][3][4]** | **31** | **30:24** | **23** | **22:16** | **15** | **14:8** | **7** | **6:0** |
| 1 | 0 | dfft0\_max0\_didx | 0 | dfft0\_max0\_ridx | 0 | dfft1\_max0\_didx | 0 | dfft1\_max0\_ridx |
| 2 | dfft0\_max0\_value | | | | | | | |
| 3 | dfft1\_max0\_value | | | | | | | |
| 4 | 0 | dfft0\_max1\_didx | 0 | dfft0\_max1\_ridx | 0 | dfft1\_max1\_didx | 0 | dfft1\_max1\_ridx |
| 5 | dfft0\_max1\_value | | | | | | | |
| 6 | dfft1\_max1\_value | | | | | | | |
| 7 | 0 | dfft0\_max2\_didx | 0 | dfft0\_max2\_ridx | 0 | dfft1\_max2\_didx | 0 | dfft1\_max2\_ridx |
| 8 | dfft0\_max2\_value | | | | | | | |
| 9 | dfft1\_max2\_value | | | | | | | |
| 10 | 0 | dfft0\_max3\_didx | 0 | dfft0\_max3\_ridx | 0 | dfft1\_max3\_didx | 0 | dfft1\_max3\_ridx |
| 11 | dfft0\_max3\_value | | | | | | | |
| 12 | dfft1\_max3\_value | | | | | | | |
| 13 | 0 | dfft0\_max4\_didx | 0 | dfft0\_max4\_ridx | 0 | dfft1\_max4\_didx | 0 | dfft1\_max4\_ridx |
| 14 | dfft0\_max4\_value | | | | | | | |
| 15 | dfft1\_max4\_value | | | | | | | |
| 16 | 0 | dfft0\_max5\_didx | 0 | dfft0\_max5\_ridx | 0 | dfft1\_max5\_didx | 0 | dfft1\_max5\_ridx |
| 17 | dfft0\_max5\_value | | | | | | | |
| 18 | dfft1\_max5\_value | | | | | | | |
| 19 | 0 | dfft0\_max6\_didx | 0 | dfft0\_max6\_ridx | 0 | dfft1\_max6\_didx | 0 | dfft1\_max6\_ridx |
| 20 | dfft0\_max6\_value | | | | | | | |
| 21 | dfft1\_max6\_value | | | | | | | |
| 22 | 0 | dfft0\_max7\_didx | 0 | dfft0\_max7\_ridx | 0 | dfft1\_max7\_didx | 0 | dfft1\_max7\_ridx |
| 23 | dfft0\_max7\_value | | | | | | | |
| 24 | dfft1\_max7\_value | | | | | | | |
| 25 | 0 | dfft0\_max8\_didx | 0 | dfft0\_max8\_ridx | 0 | dfft1\_max8\_didx | 0 | dfft1\_max8\_ridx |
| 26 | dfft0\_max8\_value | | | | | | | |
| 27 | dfft1\_max8\_value | | | | | | | |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| DW | Item | 31:16 | 15:14 | 13:12 | 11:8 | 7:0 |
| DW28 | DFFT\_PEAK\_ TAIL | check\_sum | 1 | 0 | 0 | 0x55 |

[1] cfg\_len\_rpt: length of reported data

[2] dfft\*\_max\*\_didx: dfft peak location, doppler fft index

[3] dfft\*\_max\*\_ridx: dfft peak location, range fft index

[4] dfft\*\_max\*\_value: dfft peak value

# RVA算法介绍

计算RVA需要关闭宏SUPPORT\_DATA\_PASSTHROUGH，并且打开宏SUPPORT\_RVA\_CALC，目前仅在EVBSN01平台做了实现。入口函数RVA\_CalcRva，函数说明如下：

|  |  |
| --- | --- |
| 函数名 | void RVA\_CalcRva(uint8\_t \*pbuf, uint16\_t bufLen, uint8\_t channel, uint16\_t index, uint8\_t dataType) |
| 说明 | 根据不同的数据类型，计算RVA |
| 参数说明 | * + pBuf: 驱动解析得到的雷达数据   + bufLen: 数据长度   + channel: 数据对应的通道   + index: 数据的序列号，FFT---chirp index；DFFT---frame count；DFFT PEAK--无意义   + dataType：数据类型 （FFT/DFFT/DFFT PEAK） |
| 返回值 | 无 |

## 基于FFT的RVA计算

1. 把驱动接收的FFT数据进行缓存，当缓存两个通道完整的frame之后进行DFFT处理（当前配置每个frame包含32个chirp）
2. DFFT处理：对每个距离门做32点FFT运算，再执行FFTSHIFT将FFT的DC分量移到频谱中心，然后重组成DFFT数据（32\*32的矩阵）
3. ClusterRemove()：动态去杂波
4. complex2abs2(): 把0通道的DFFT复数值转换成能量值，生成RDMap(dataAbs2)
5. 开机阶段（前30个frame）基于RDMap计算当前环境的背景值，在后续提取RVA时先做去背景处理
6. findMaxPos\_2d(): 基于RDMap，寻找最大值以及最大值对应的行索引和列索引，其中行索引对应速度，列索引对应距离
7. 如果RDMap最大值大于给定的阈值，则认为存在有效目标，可以进行RVA计算
8. R计算：列索引 \* 距离分辨率，距离分辨率计算公式为：，其中c表示光速，B表示有效扫频带宽
9. V计算：（行索引 – DC分量索引）\* 速度分辨率，速度分辨率计算公式为：，其中表示波长，T表示frame时长
10. A计算：通过最大值索引找到两通道对应的复数值，再做共轭相乘得到相位差。基于相位差计算角度的公式为：，其中表示波长，表示相位差，表示接收天线间距

## 基于DFFT的RVA计算

1. 把驱动接收的DFFT数据进行缓存，当缓存两个通道完整的frame之后进行后续处理（每个frame为32\*32的矩阵）
2. complex2abs2(): 把0通道的DFFT复数值转换成能量值，生成RDMap(dataAbs2)
3. 开机阶段（前30个frame）基于RDMap计算当前环境的背景值，在后续提取RVA时先做去背景处理
4. findMaxPos\_2d(): 基于RDMap，寻找最大值以及最大值对应的行索引和列索引，其中行索引对应速度，列索引对应距离
5. 如果RDMap最大值大于给定的阈值，则认为存在有效目标，可以进行RVA计算
6. R计算：列索引 \* 距离分辨率，距离分辨率计算公式为：，其中c表示光速，B表示有效扫频带宽
7. V计算：（行索引 – DC分量索引）\* 速度分辨率，速度分辨率计算公式为：，其中表示波长，T表示frame时长
8. A计算：通过最大值索引找到两通道对应的复数值，再做共轭相乘得到相位差。基于相位差计算角度的公式为：，其中表示波长，表示相位差，表示接收天线间距

## 基于DFFT PEAK的RV计算

1. 把驱动接收的DFFT PEAK数据进行解析得到各PEAK点的速度索引，距离索引和PEAK值
2. 对所有PEAK值进行比较，找到最大值。如果最大值大于给定阈值，则认为存在有效目标，可以进行RV计算
3. R计算：距离索引 \* 距离分辨率，距离分辨率计算公式为：，其中c表示光速，B表示有效扫频带宽
4. V计算：（速度索引 – DC分量索引）\* 速度分辨率，速度分辨率计算公式为：，其中表示波长，T表示frame时长