Technical Article

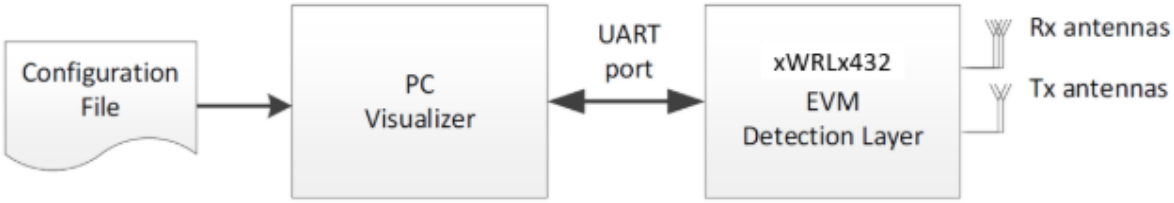
适用于 xwrLx432 motion/presence detection demo 的固化配置及检测结果自定义输出方式

TEXAS INSTRUMENTS

xwrLx432 作为 TI 低功耗低成本的毫米波雷达 SoC，具有很高的性价比，得到越来越多客户的青睐。TI SDK 自带的 motion/presence detection demo 提供了完备的功能，具有非常高的灵活性，在客户评估阶段可以通过 PC GUI 灵活调整发送配置，并实时查看检测结果。量产阶段，需要将参数配置固化并按照客户需求输出结果，本文介绍参数的固化，及如何修改代码输出客户想要的结果，便于软件在加载后自动运行并输出所需的结果。

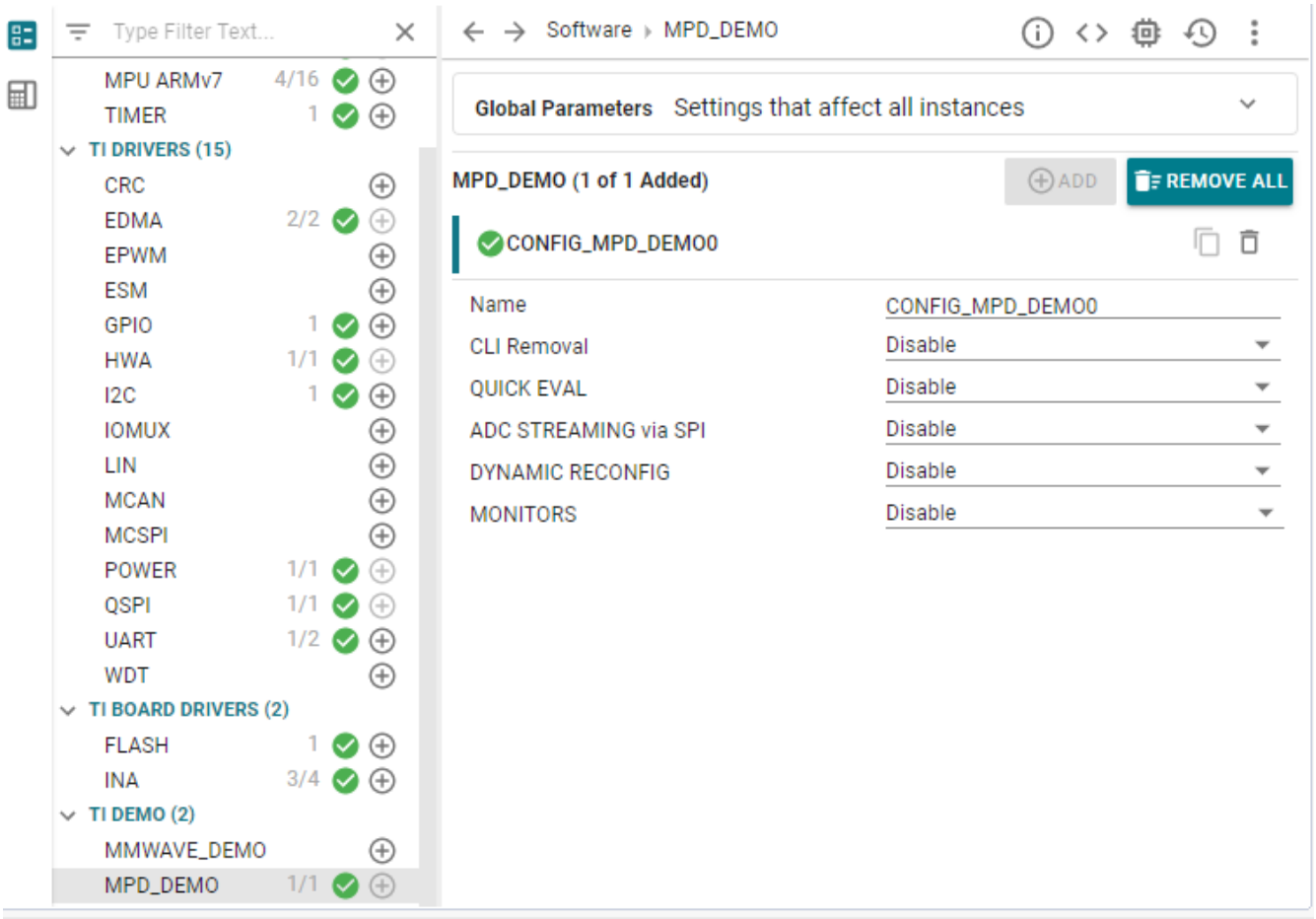
1. motion/presence detection demo 快速上手

该 demo 的运行环境如下，EVM 与 PC 通过 UARTB 进行通信，在板子上电加载程序后，需要 PC GUI 通过串口 UARTB 发送配置，EVM 收到配置后配置启动射频前端及应用程序处理，通过相同的串口 UARTB 输出检测的结果。默认 demo 输出的结果是由配置决定，并输出固定的格式。具体可参考 toolbox motion/presence detection demo user guide。



2. 快速固化配置

该 demo 提供两种方式支持参数的固化，CLI REMOVAL 及 QUICK EVAL， 可以通过如下 syscfg 界面进行配置，默认这两个功能是 disable，需要通过 PC GUI 发送配置启动。但是 CLI REMOVAL 的使用有一定的限制，如果 enable CLI REMOVAL, 则如下其他的功能如 ADC streaming，Dynamic reconfig 和 MONITORS 都不能使能，但实际应用中往往需要其并存。QUICK EVAL 则没有这个限制，因此这里建议通过 enable QUICK EVAL，将参数固化。



在 syscfg 中 enable QUICK EVAL 之后，编译自动生成的 ti_cli_mpd_demo_config.h 中会将生成如下的宏定义。

```
#define QUICK_START 1
与配置固化相关的代码如下，不需要客户做任何修改。
Motion_detect.c:
#if (CLI_REMOVAL == 0 && QUICK_START == 1)
      // Task function
      void CLI_defaultcfg_task(void *args);
#endif

#if (CLI_REMOVAL == 0 && QUICK_START == 1)
      // Create a Task for running default configuration
      gDefCfgTask = xTaskCreateStatic(CLI_defaultcfg_task, /* Pointer to the function that
implements the task. */
            "Run_Defaultcfg", /* Text name for the task. This is to facilitate debugging only. */
            DEFAULT_CFG_TASK_SIZE, /* Stack depth in units of StackType_t typically uint32_t on 32b
CPUs */
            NULL, /* We are not using the task parameter. */
            DEFAULT_CFG_TASK_PRI, /* task priority, 0 is lowest priority, configMAX_PRIORITIES-1 is
highest */
            gDefCfgTaskStack, /* pointer to stack base */
            &gDefCfgTaskObj); /* pointer to statically allocated task object memory */
#endif

Mmw_cli.c:
#if (CLI_REMOVAL == 0 && QUICK_START == 1)
extern uint8_t gIsDefCfgUsed;
void CLI_defaultcfg_task(void *args)
{
   gIsDefCfgUsed = 1;
   CLI_write("\r\n Starting the Demo with Default Configurations...\r\n");
   CLI_ByPassApi(&gCLI.cfg, 0);
   CLI_write("Running the demo...\r\n");
   vTaskDelete(NULL);
}
```

```
#endif
```

修改如下，根据实际的平台将配置参数更新到如下数组：

char* radarCmdStringAOP[MAX_RADAR_CMD_AOP] for xwrL6432AOP

char* radarCmdString[MAX_RADAR_CMD] for xwrLx432 NON AOP

注意需要根据实际配置参数种类，修改宏定义，保证实际的参数种类小于宏的大小。

#define MAX_RADAR_CMD 20

#define MAX_RADAR_CMD_AOP 45

3. 检测结果输出

默认 demo 根据如下 guimonitor 配置，输出相应的数据，具体参数的定义参考 Motion_Presence_Detection_Demo_Tuning_Guide.pdf。默认输出数据格式见 toolbox motion/presence detection demo user guide。评估阶段可以直接用 TI GUI 如 industry visualizer 接收输出，解析显示检测结果。

guiMonitor 0 0 0 0 0 1 1 0 1 0 0

量产阶段每个客户可以自定义输出数据，不需要依赖 guimonitor。检测的点云及目标结果存储在 gMmwMssMCB.dpcResult，如下定义格式。注意，需要配置 guiMonitor 的第一个参数 PointCloud 为 1，检测的点云结果才会存储在如下 objOut 及 objOutSideInfo。

```
typedef struct DPC_ObjectDetection_ExecuteResult_t
{
   /*! @brief Number of detected objects - Major motion */
      uint16_t numObjOutMajor;
   /*! @brief Number of detected objects - Minor motion */
      uint16_t numObjOutMinor;
   /*! @brief Total Number of detected objects */
      uint32_t numObjOut;
   /*! @brief Detected objects output list of @ref numObjOut elements */
      DPIF_PointCloudCartesian *objOut;
   /*! @brief Detected objects side information (snr + noise) output list,* of @ref numObjOut
elements */
      DPIF_PointCloudSideInfo *objOutSideInfo;
   /*! @brief Range Azimuth Heatmap [0] - for Major and [1] - for Minor Motion*/
      uint32_t *rngAzHeatMap[2];
   /*! @brief Tracker output */
      DPU_TrackerProc_OutParams trackerOutParams;
   /*! @brief Tracker output, includes pointers */
      DPU_uDopProc_OutParams microDopplerOutParams;
} DPC_ObjectDetection_ExecuteResult;
```

自定义数据输出代码可以添加在任务 mmwDemo_TransmitProcessedOutputTask，如下为参考代码，可根据实际需求调整修改。

如下参考代码，将检测区域等分成左中右三个区域，输出每个区域的检测目标数，及每个目标的位置，最大支持 5 个目标。增加相应的宏 OUT_NO_GUI 控制该功能开关。

```
#ifdef OUT_NO_GUI
typedef struct TARGET_POS_T
{
    //decimeters
    int8_t x;
    int8_t y;
    int8_t z;
}TARGET_POS;
typedef struct OUT_TOHOST_T
{
    uint8_t hdr; //0x77
    uint8_t peopleCnt[3]; //max 3regions
    TARGET_POS targetList[5]; //maximum total 5person
    uint8_t crc;
}OUT_TOHOST;
#endif
```

```
void mmwDemo_TransmitProcessedOutputTask()
{
   UART_Handle uartHandle = gUartHandle[0];
  I2C_Handle i2cHandle = gI2cHandle[CONFIG_I2C0];
   DPC_ObjectDetection_ExecuteResult *result = &gMmwMssMCB.dpcResult;
    ……
   uint8_t *tIndex;
#ifdef OUT_NO_GUI
    float *trackerList;
    float track_x;
    float track_y;
    float track_z;
    OUT_TOHOST outToHost_info;
#endif

 uint8_t *uDopplerData;
    ……
    mpdEnabled = gMmwMssMCB.isMotionPresenceDpuEnabled;
    while(true)
    {
     SemaphoreP_pend(&gMmwMssMCB.tlvSemHandle, SystemP_WAIT_FOREVER);
/
**************************************************************************************************
*/

      tlvIdx = 0;
   #ifdef OUT_NO_GUI
      numTargets = result->trackerOutParams.numTargets;
      numIndices = result->trackerOutParams.numIndices;
      tList = (uint8_t *)result->trackerOutParams.tList;
      tIndex = (uint8_t *)result->trackerOutParams.targetIndex;
      trackerList = (float *)result->trackerOutParams.tList;
   #if 1 //通过串口输出自定义格式数据
      memset((void *)&outToHost_info, 0, sizeof(OUT_TOHOST));
      outToHost_info.hdr = 0x77;
      outToHost_info.crc += outToHost_info.hdr;
      for(tlvIdx = 0; tlvIdx < numTargets && tlvIdx < 5; tlvIdx++) //maximum 5person
      {
            track_x = (float) *(trackerList +1);
            if(track_x >= -2 && track_x <= -0.67)
      {
            outToHost_info.peopleCnt[0] +=1;
      }
            else if(track_x >= -0.66 && track_x <= 0.66)
      {
            outToHost_info.peopleCnt[1] +=1;
      }
            else if(track_x >= 0.67 && track_x <= 2)
      {
            outToHost_info.peopleCnt[2] +=1;
      }
      track_y = (float) *(trackerList +2);
      track_z = (float) *(trackerList +3);
      outToHost_info.targetList[tlvIdx].x = (int8_t)(track_x * 10);
      outToHost_info.targetList[tlvIdx].y = (int8_t)(track_y * 10);
      outToHost_info.targetList[tlvIdx].z = (int8_t)(track_z * 10);
      trackerList = trackerList + (sizeof(trackerProc_Target)>>2);
      }
      outToHost_info.crc = 0x66;
      mmw_UartWrite(uartHandle, (uint8_t*)&outToHost_info, sizeof(OUT_TOHOST));
      UART_flushTxFifo(uartHandle);
   #else
```

//串口输出 ASCII 码，可以直接在串口调试助手如 putty 上显示打印字符结果。

```
      CLI_write("\r\n");
      CLI_write("Frame No: (%d), \r\n", gMmwMssMCB.stats.frameStartIntCounter);
      CLI_write("No of Detected Tracks (%d), \r\n", numTargets);
      //output tracker
      for(tlvIdx = 0; tlvIdx < numTargets ; tlvIdx++)
      {
            track_x = (float) *(trackerList +1);
            track_y = (float) *(trackerList +2);
            trackerList = trackerList + (sizeof(trackerProc_Target)>>2);
            CLI_write("Object (%d, %.2f, %.2f) \r\n", tlvIdx, track_x, track_y);
```

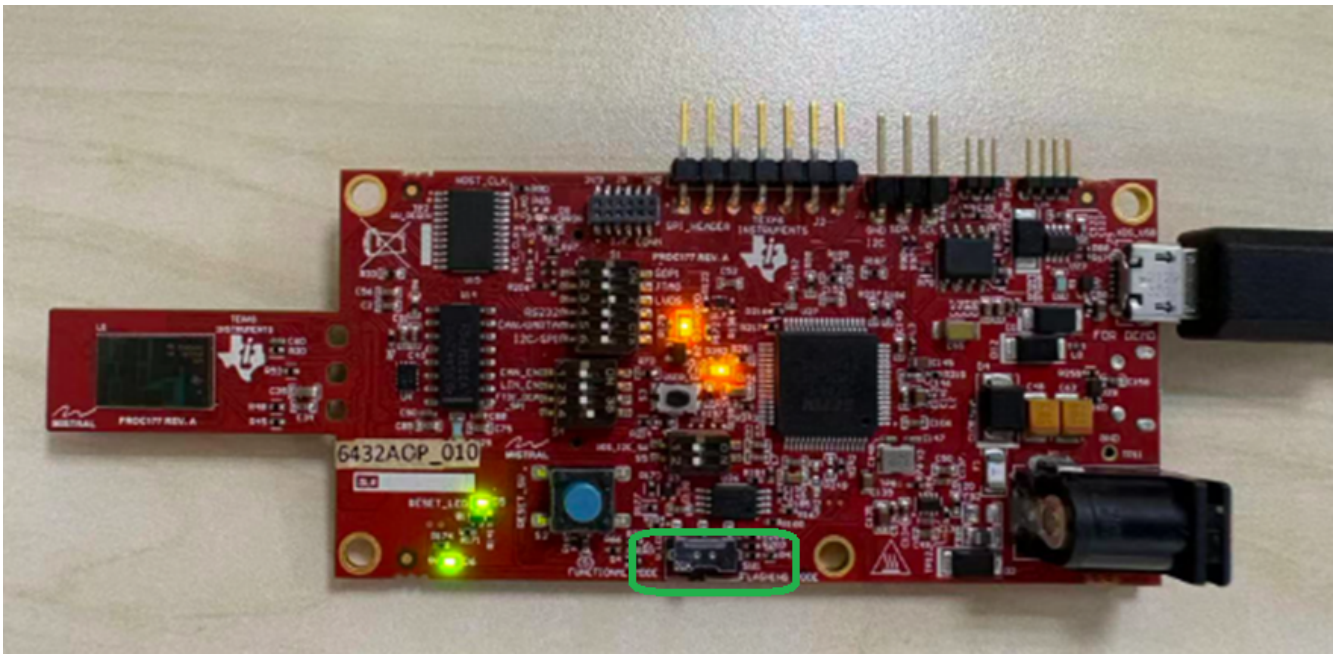
```
      }
#endif
#else
//comment all other default output
#endif
//keep below default code
//Interframe processing and UART data transmission completed
      gMmwMssMCB.interSubFrameProcToken--;
……
}
```

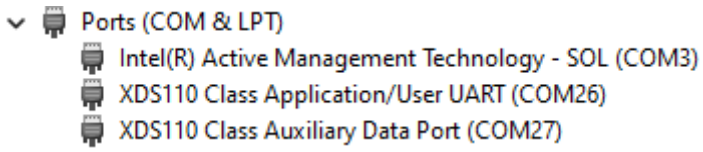
4. 测试结果

将配置固化，按上述输出格式，修改代码，编译生成 image，烧录在 EVM flash，板子上电后加载程序自动根据固化参数，配置启动射频，及应用程序处理，输出结果在串口调试助手。如下是硬件操作及输出说明，供参考。

硬件板子操作：

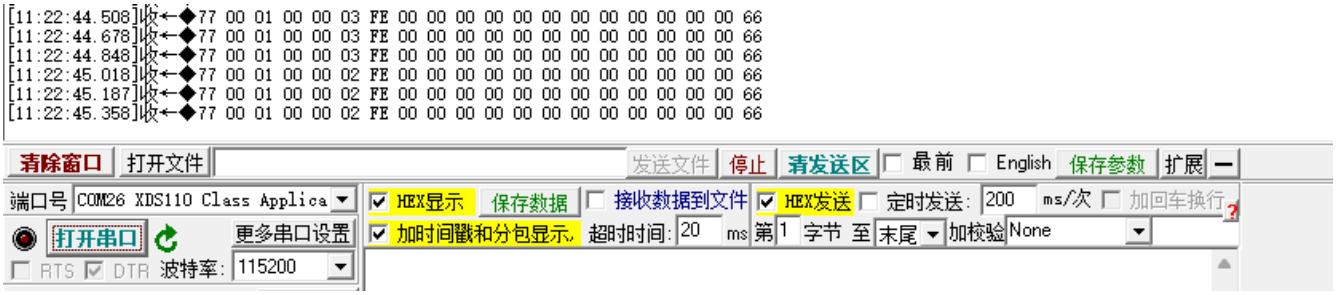
如下使用 EVM 自带的 micro USB 将 xds-USB 口与 PC 连接，即可通过 USB 上电，并在 pc device manager 上看到如下新增两个 port，程序的烧录、配置及结果输出都是通过 user port。程序的烧录及加载通过如下绿色标注的 SW1 切换，flash mode 对应通过 user port 将程序烧录到板载 flash，function mode 对应上电从 flash 加载程序并启动，注意模式的切换之后需要掉电重新上电生效。其他的拨码开关参考如下图示即可，具体可以参考 IWRL6432AOP EVM user guide。



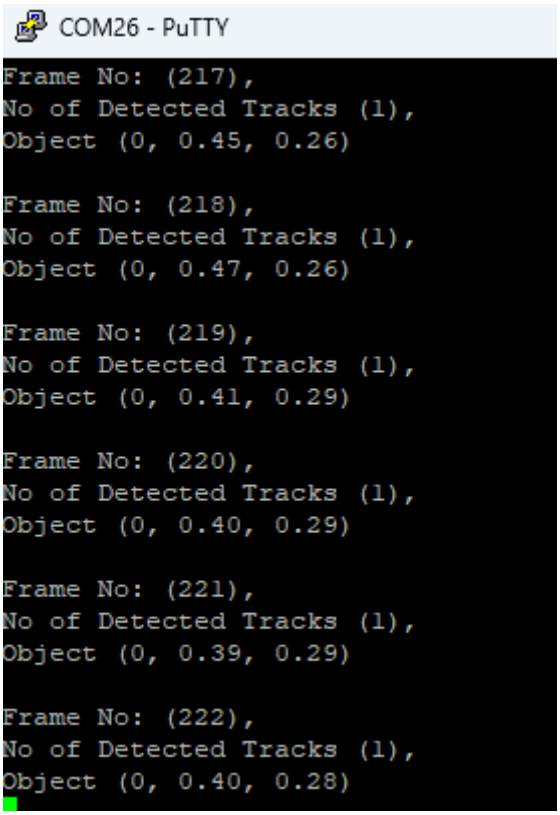


默认的 motion/presence detection demo 可以通过 low Power visualizer 或者 industry visualizer 进行配置，解析数据并观察检测结果。具体可参考 radar_toolbox_path\source\ti\examples\Industrial_and_Personal_Electronics\Motion_and_Presence_Detection\Motion_and_Presence_Detection\docs\motion_and_presence_detection_users_guide.html.

按上述第 3 节所述，自定义格式输出检测结果，如下是使用串口调试助手实时接收解析的数据。



如下是实时输出字符串的图示结果。



5. 参考资料

- MMWAVE_L_SDK_install_path/docs/api_guide_xwrL64xx/MOTION_AND_PRESENCE_DETECTION_DEMO.html
- MMWAVE_L_SDK_install_path\docs\Low_Power_Visualizer_User_Guide.pdf\
- MMWAVE_L_SDK_install_path\docs\Motion_Presence_Detection_Demo_Tuning_Guide.pdf
- radar_toolbox_3_20_00_04\source\ti\examples\Industrial_and_Personal_Electronics\Motion_and_Presence_Detection\Motion_and_Presence_Detection\docs\motion_and_presence_detection_users_guide.html
- IWRL6432AOPEVM user guide SPRU620B

重要通知和免责声明

TI“按原样”提供技术和可靠性数据（包括数据表）、设计资源（包括参考设计）、应用或其他设计建议、网络工具、安全信息和其他资源，不保证没有瑕疵且不做出任何明示或暗示的担保，包括但不限于对适销性、与某特定用途的适用性或不侵犯任何第三方知识产权的暗示担保。

这些资源可供使用 TI 产品进行设计的熟练开发人员使用。您将自行承担以下全部责任：(1) 针对您的应用选择合适的 TI 产品，(2) 设计、验证并测试您的应用，(3) 确保您的应用满足相应标准以及任何其他安全、安保法规或其他要求。

这些资源如有变更，恕不另行通知。TI 授权您仅可将这些资源用于研发本资源所述的 TI 产品的相关应用。严禁以其他方式对这些资源进行复制或展示。您无权使用任何其他 TI 知识产权或任何第三方知识产权。对于因您对这些资源的使用而对 TI 及其代表造成的任何索赔、损害、成本、损失和债务，您将全额赔偿，TI 对此概不负责。

TI 提供的产品受 TI 销售条款)、TI 通用质量指南 或 ti.com 上其他适用条款或 TI 产品随附的其他适用条款的约束。TI 提供这些资源并不会扩展或以其他方式更改 TI 针对 TI 产品发布的适用的担保或担保免责声明。 除非德州仪器 (TI) 明确将某产品指定为定制产品或客户特定产品，否则其产品均为按确定价格收入目录的标准通用器件。

TI 反对并拒绝您可能提出的任何其他或不同的条款。

最后更新日期：2025 年 10 月