*Technical Article*

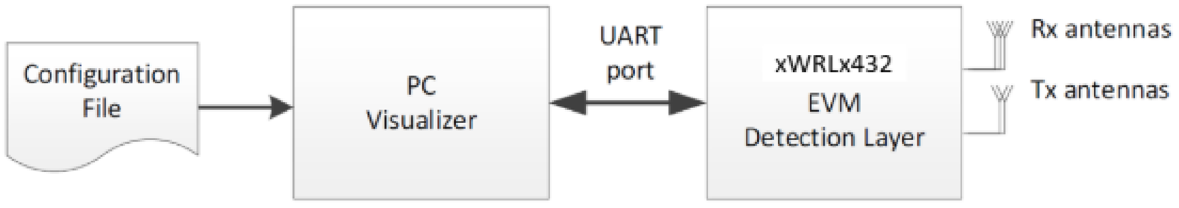
# *The finalized configuration for xwrLx432 motion/presence detection demo and custom output of detection results*

*This document was translated from a simplified Chinese source.* (ZHCTA05)

xwrLx432, as a low-power and low-cost mmwave radar SoC, offers exceptional value for money and is gaining increasing favor among customers. The motion/presence detection demo comes with the TI SDK, which provides full functionality with exceptional flexibility. During the customer evaluation phase, transmission configurations can be flexibly adjusted via PC GUI, with detection results visible in real time. For mass production, parameter configurations must be finalized and results are output based on customer requirements. This document details how to finalize parameters and modify code to output desired results, enabling the software to run automatically upon loading and deliver the required output.

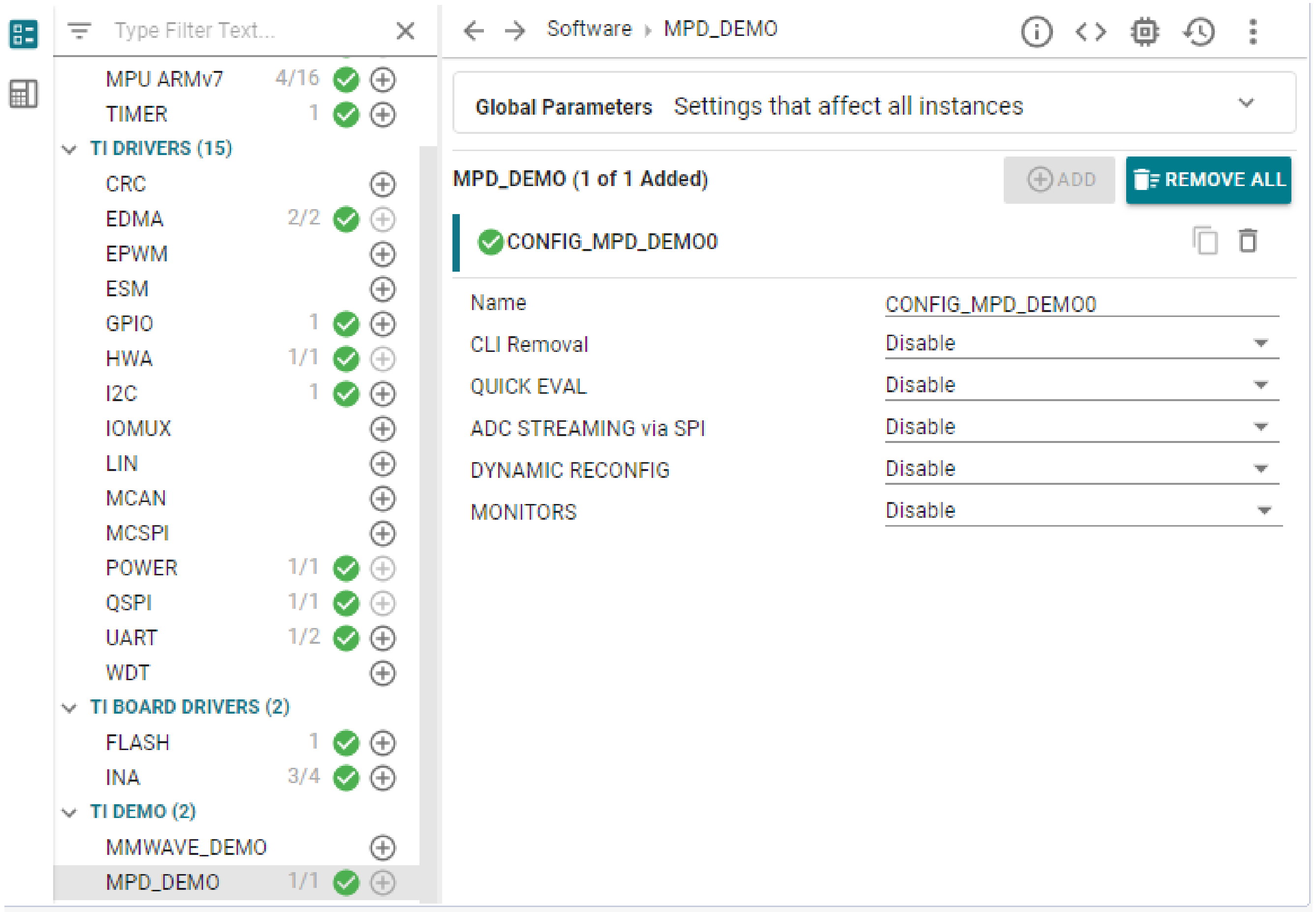
## 1. Quick start for motion/presence detection demo

The demo operates as follows: the EVM communicates with the PC via UARTB. After powering the board and loading the program, the PC GUI sends configurations via the UARTB serial port. Upon receiving these, the EVM configures the RF front-end and application processing, then outputs detection results through the same UARTB serial port. By default, the demo outputs results determined by the configuration in a fixed format. This can be found in the toolbox motion/presence detection demo user guide.



## 2. Quick finalization of configuration

This demo supports two methods for parameter finalization: CLI REMOVAL and QUICK EVAL. Configuration is accessible via the syscfg interface below. By default, both functions are disabled and require configuration initiation through the PC GUI. However, the use of CLI REMOVAL imposes certain restrictions. If CLI REMOVAL is enabled, other features such as ADC streaming, Dynamic reconfig, and MONITORS cannot be activated. However, in practical applications, it is often necessary for these features to coexist. QUICK EVAL does not impose this restriction. Therefore, it is recommended to enable QUICK EVAL to finalize parameters.



| | |
|---|---|
| Name | CONFIG_MPD_DEMO0 |
| CLI Removal | Disable |
| QUICK EVAL | Disable |
| ADC STREAMING via SPI | Disable |
| DYNAMIC RECONFIG | Disable |
| MONITORS | Disable |

After enabling QUICK EVAL in syscfg, the following macro definition is generated in the auto-generated ti_cli_mpd_demo_config.h that is compiled.

```
#define QUICK_START 1
    The following code is related to saving the configuration and requires no modifications from
the user.
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
```

It is modified as follows and the configuration parameters are updated to the array below according to the actual platform:

char* radarCmdStringAOP[MAX_RADAR_CMD_AOP] for xwrL6432AOP

char* radarCmdString[MAX_RADAR_CMD] for xwrLx432 NON AOP

Note that the macro definitions must be modified based on the actual configuration parameter types, ensuring the number of actual parameter types is less than the macro size.

#define MAX_RADAR_CMD 20

#define MAX_RADAR_CMD_AOP 45

## 3. Detection result output

The default demo outputs the corresponding data based on the following guimonitor configuration, and the specific parameters are defined by referring to Motion_Presence_Detection_Demo_Tuning_Guide.pdf. The default output data format can be found in the toolbox motion/presence detection demo user guide. During evaluation, outputs can be directly received via TI GUI such as industry visualizer to parse and display detection results.

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

During mass production, each customer may customize output data without relying on guimonitor. The detected point cloud and the target results are stored in gMmwMssMCU.dpcResult, in the format defined below. Note that the first parameter PointCloud in guiMonitor must be set to 1 and thus the detected point cloud results can be stored in objOut and objOutSideInfo as shown below.

```
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;
```

The custom data output code can be added to the task mmwDemo_TransmitProcessedOutputTask. The reference code below may be adjusted and modified according to actual requirements.

The reference code below divides the detection zone into three equal zones (left, center, right), outputting the number of detected targets in each zone and the position of each target, supporting up to 5 targets. Add the corresponding macro OUT_NO_GUI to control the function switch.

```
#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 //output custom format data via serial port
      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
```

// The serial port outputs ASCII codes, which can display the results of printed characters directly on the serial debug assistant, such as putty.

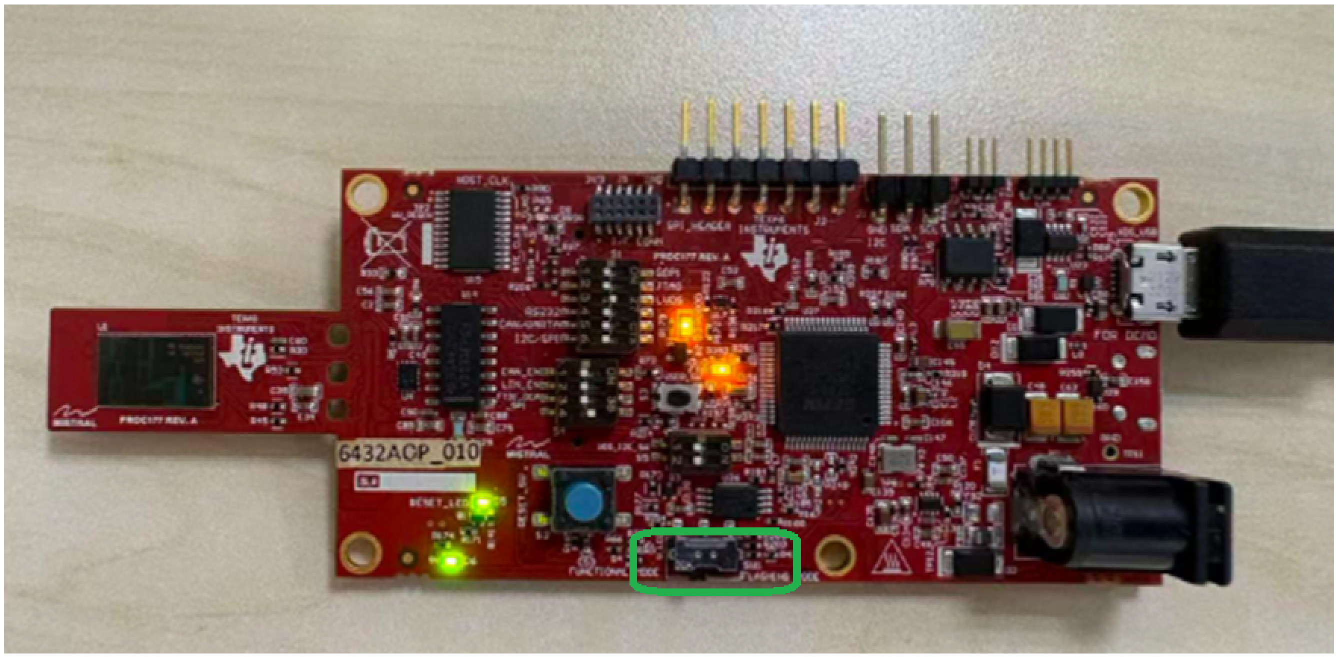
```
        CLI_write("\r\n");
        CLI_write("Frame No: (%d), \r\n", gMmwMssMCB.stats.frameStartIntCounter);
        CLI_write("No of Detected Tracks (%d), \r\n", numTargets);
        //output tracker
        for(tlvIdx = 0; tlvIdx < numTargets ; tlvIdx++)
        {
              track_x = (float) *(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--;
……
}
```

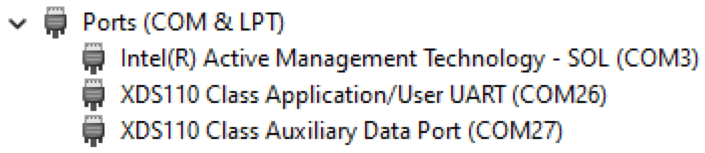
## 4. Test results

After finalizing the configuration, modify the code according to the above output format, compile to generate the image, and flash it onto the EVM's flash memory. Upon powering the board, the program loads automatically and configures and then starts the RF and application processing based on the finalized parameters. The output results are displayed via the serial port debugging assistant. The following is a description of the hardware operation and output for reference.

Hardware board operations:

Connect the xds-USB port to a PC via the EVM's integrated micro USB cable to power on via USB. Two new ports will appear in the pc's device manager. Program burning, configuration, and result output are conducted through the user port. Program burning and loading are toggled via the SW1 switch highlighted in green below. The flash mode corresponds to burning the program to the on-board flash via the user port, while the function mode corresponds to loading the program from flash and starting it upon power-up. Note that mode changes require a power cycle to take effect. For the other DIP switches, please refer to the diagram below. For specific details, consult the IWRL6432AOP EVM user guide.

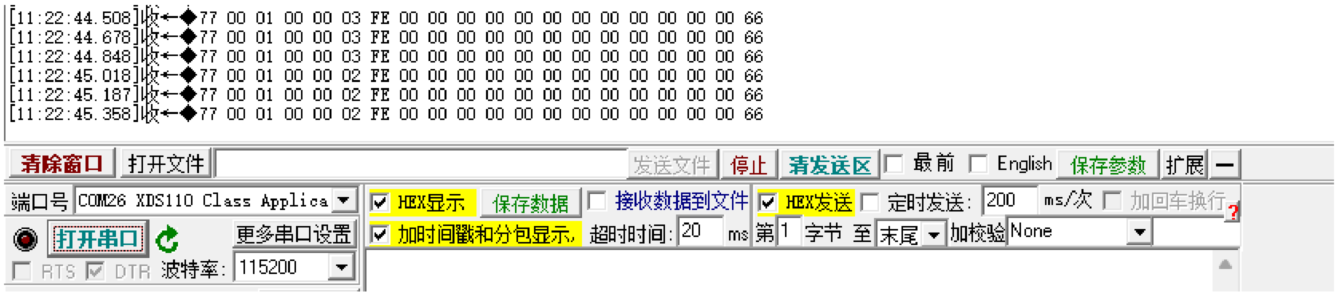




Ports (COM & LPT)
Intel(R) Active Management Technology - SOL (COM3)
XDS110 Class Application/User UART (COM26)
XDS110 Class Auxiliary Data Port (COM27)

The default motion/presence detection demo can be configured through a low Power visualizer or an industry visualizer to parse the data and observe the detection results. For details, you can refer to 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.

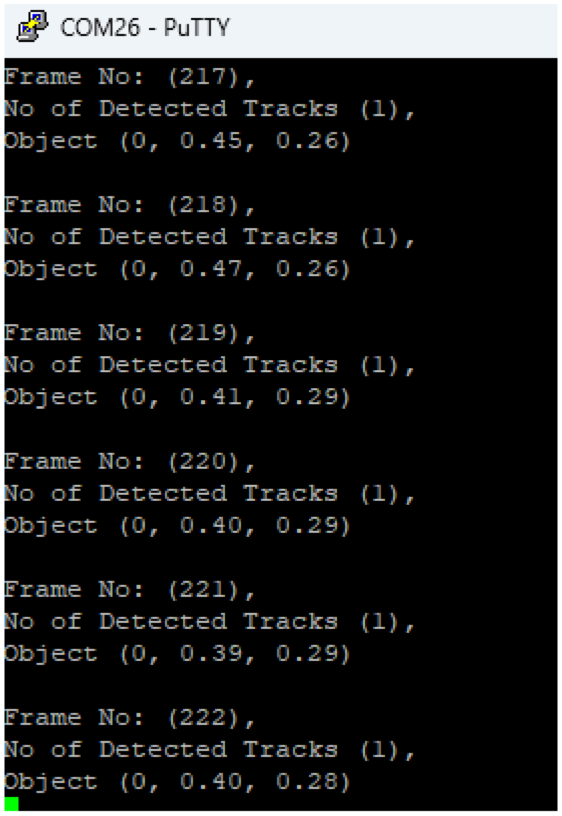
As outlined in Section 3 above, the customized formats output the detection results. Below is the data received and parsed in real time using the serial port debug assistant.



```
[11:22:44.508]收←◆77 00 01 00 00 03 FE 00 00 00 00 00 00 00 00 00 00 00 00 66
[11:22:44.678]收←◆77 00 01 00 00 03 FE 00 00 00 00 00 00 00 00 00 00 00 00 66
[11:22:44.848]收←◆77 00 01 00 00 03 FE 00 00 00 00 00 00 00 00 00 00 00 00 66
[11:22:45.018]收←◆77 00 01 00 00 02 FE 00 00 00 00 00 00 00 00 00 00 00 00 66
[11:22:45.187]收←◆77 00 01 00 00 02 FE 00 00 00 00 00 00 00 00 00 00 00 00 66
[11:22:45.358]收←◆77 00 01 00 00 02 FE 00 00 00 00 00 00 00 00 00 00 00 00 66
```

端口号: COM26 XDS110 Class Applica
[✓] HEX显示
保存数据
☐ 接收数据到文件
[✓] HEX发送
☐ 定时发送: 200 ms/次
☐ 加回车换行
打开串口
更多串口设置
[✓] 加时间戳和分包显示, 超时时间: 20 ms 第 1 字节 至 末尾 加校验 None
☐ RTS [✓] DTR 波特率: 115200

The following is a diagrammatic representation of the real-time output string.



```
Frame No: (217),
No of Detected Tracks (1),
Object (0, 0.45, 0.26)

Frame No: (218),
No of Detected Tracks (1),
Object (0, 0.47, 0.26)

Frame No: (219),
No of Detected Tracks (1),
Object (0, 0.41, 0.29)

Frame No: (220),
No of Detected Tracks (1),
Object (0, 0.40, 0.29)

Frame No: (221),
No of Detected Tracks (1),
Object (0, 0.39, 0.29)

Frame No: (222),
No of Detected Tracks (1),
Object (0, 0.40, 0.28)
```

## 5. References

- 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