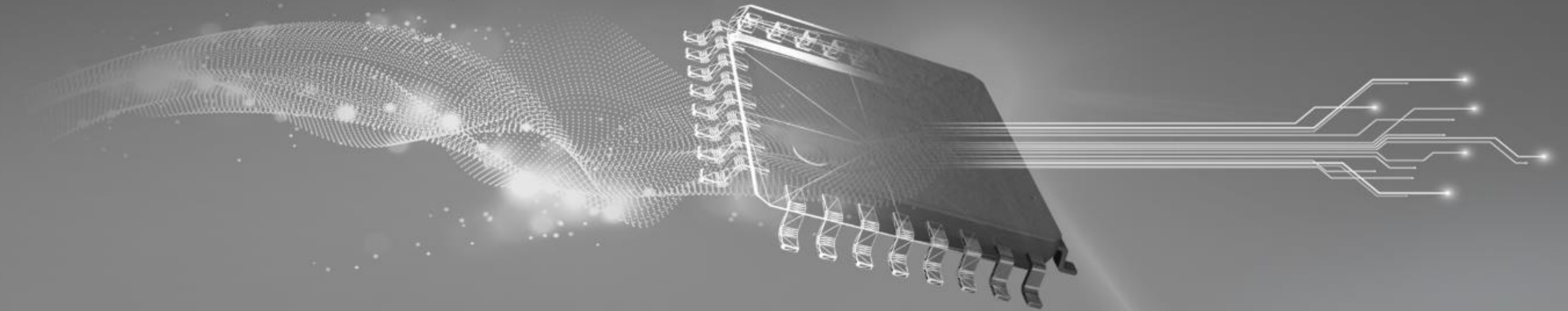


# TI TECH DAYS



## **MMWAVE-SDK deep dive Easy evaluation and development of mmWave systems with software development kit**

**Nitin Sakhuja - Industrial mmWave Radar Applications**

# MMWAVE-SDK deep dive

TI's MMWAVE-SDK (MilliMeter Wave Software Development Kit) is a unified software platform for the TI mmWave Sensing Portfolio, providing easy setup and fast out-of-the-box access to evaluation and development.

This training provides an overview of the MMWAVE-SDK 3.x architecture and the various building blocks such as Data Processing Units (DPUs) and Data Processing Chains(DPCs). It also provides a deeper look into the components with software execution flows accompanied with source code references from the MMWAVE-SDK Out of Box Demo point cloud processing chain.

We also present some example applications where the Out of box point cloud detection chain is extended to develop more complex mmWave applications such as Long Range People Detection and Tracking Demo, Traffic Monitoring Demo and Area Scanner Demo.

## What you'll learn:

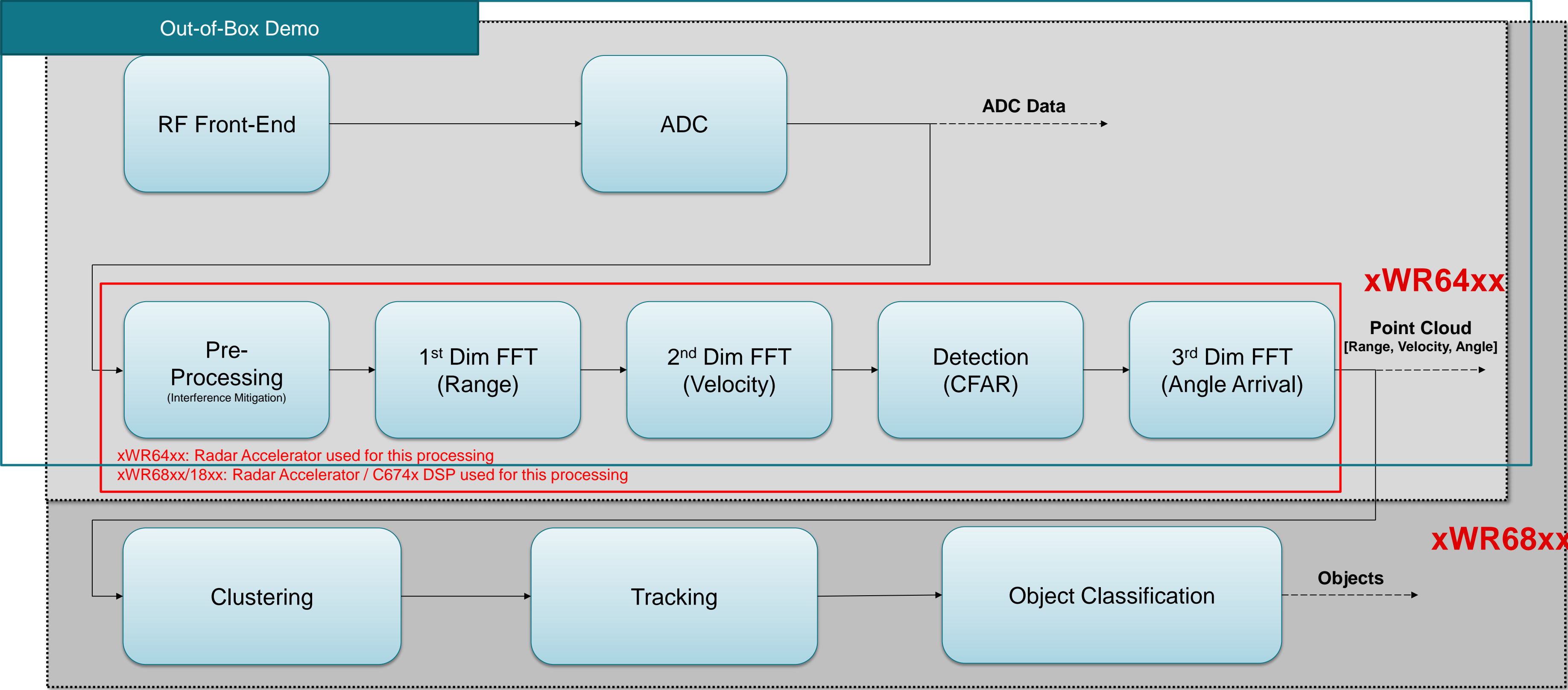
- TI MMWAVE-SDK architecture and it's various building blocks such as DPCs and DPUs
- Understand DPM, DPC and DPU execution flows e.g. initialization and runtime operation using source code references
- Developing custom components to extend the out of box processing chain and available examples.

# Agenda

- MMWAVE-SDK
  - Architecture overview
  - Data path design
- Data path deep dive
  - Initialization
  - Configuration
  - Execution (Runtime view)
- DPUs and DPCs in MMWAVE-SDK 3.x
  - DPUs: Range, Static-Clutter removal, Doppler, CFAR-CA and AoA
  - DPCs: HWA and DSP based object detection chains
- Software development and debugging
  - Development resources
  - MMWAVE-SDK debugging
- Extending SDK architecture for advanced applications
  - Considerations for developing custom DPUs and DPCs
  - Custom DPUs and DPCs in Industrial Toolbox
  - Demo: Long Range People Tracking and, Traffic Monitoring
  - Demo: Area Scanner and, Automated Doors and Gates

# MMWAVE-SDK - Architecture Overview

# mmWave signal processing



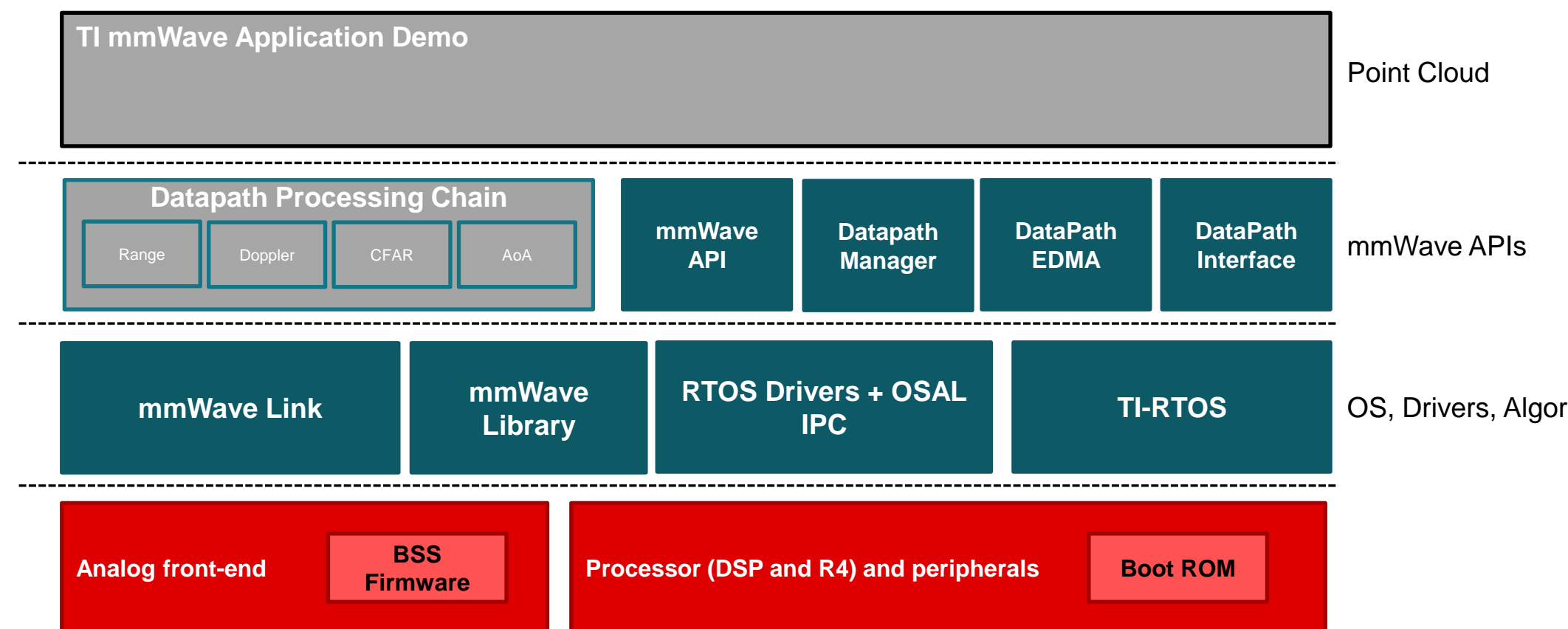
# mmWave SDK contents

- Building blocks
  - RTOS, Drivers, and RadarSS firmware
  - Scalable data processing blocks and chains to work on HWA or DSP
  - Layered / API based Radar analog front end (AFE) programming
  - Pre-built software blocks and chains for basic FMCW Radar signal processing
  - Catalog of mmWave signal processing algorithms optimized for C674x DSPs including tracker
  - Package for high-security (HS) devices to enable programming encryption keys and encrypt/authenticate program binaries
- Demonstrations and examples
  - TI RTOS based
  - Out of box demo with easy configurability via TI Cloud-based or offline GUI
  - Representation of point cloud and benchmarking data from demo via GUI
- Documentation
  - Associated tools: [Code Composer Studio](#), [TI-RTOS](#), [Uniflash](#)
  - Available at <http://www.ti.com/tool/MMWAVE-SDK>

# mmWave SDK architecture

## Main highlights:

- Foundational components for SOC enablement – RTOS, Drivers, mmWaveLink, mmWaveLib
- RF Front-end completely abstracted using mmWaveLink
- mmWave API simplifies device integration of mmWaveLink
- Data path layer is an abstraction over existing driver APIs in the data flow
- Separation of data processing units and chain from the application
- Simpler application that does the instantiation of the Datapath layer

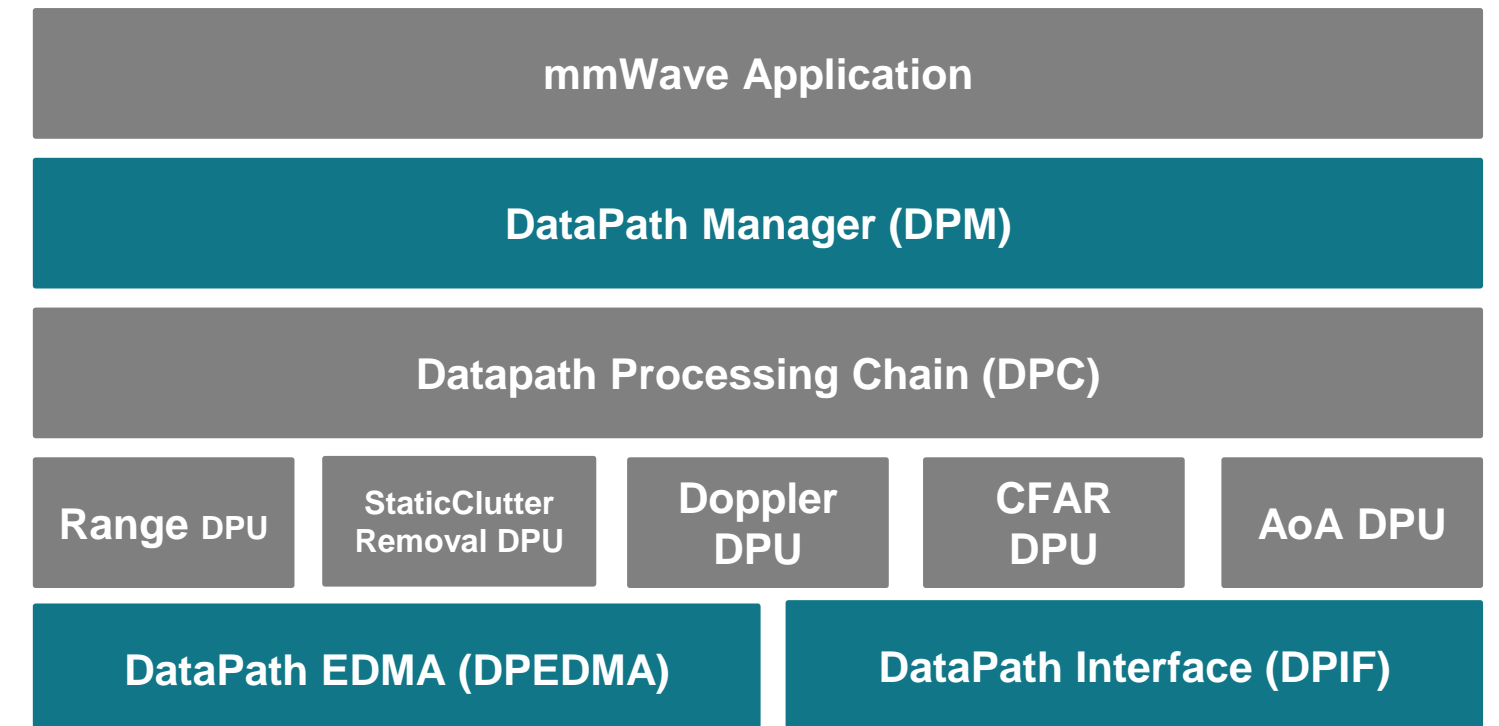


# MMWAVE-SDK – Datapath Design



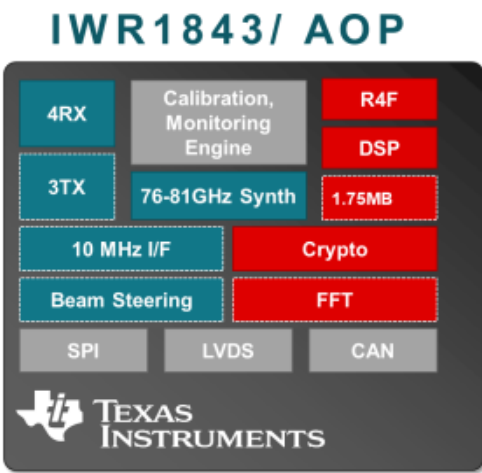
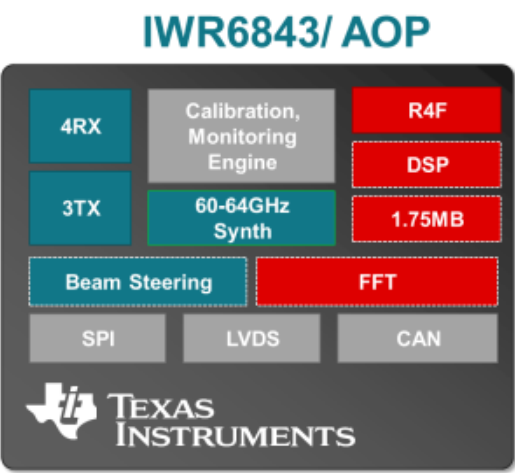
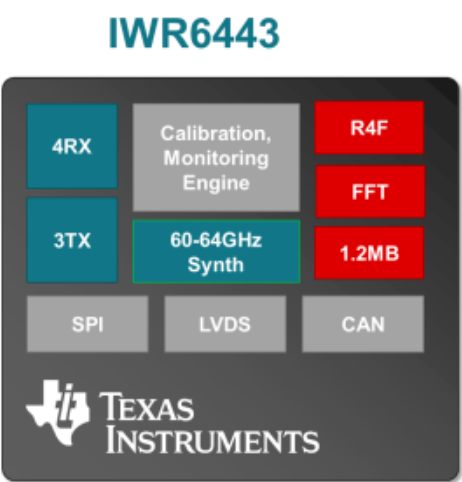
# Datapath Layer Design

- **DPM: Datapath manager**
  - Foundation layer that enables the "scalability" aspect of the architecture.
- **DPIF: Standard Interface points in the Detection chain are defined**
  - Input ADC data, Radar Cube, Detection Matrix, Point cloud
- **DPU: Data Translating function(s) from one interface point to the other are called “Data Processing Units”**
  - Range Processing (ADC data to Radar Cube)
  - Doppler Processing (Radar Cube to Detection Matrix)
  - CFAR and AoA (Detection Matrix to Point Cloud)
- **DPC: Data Processing Chain**
  - Chain of “data processing units” is called a data processing Chain.  
Ex: Detection DPC (ADC to Point Cloud).
  - This conforms to the DPM dictated API definition

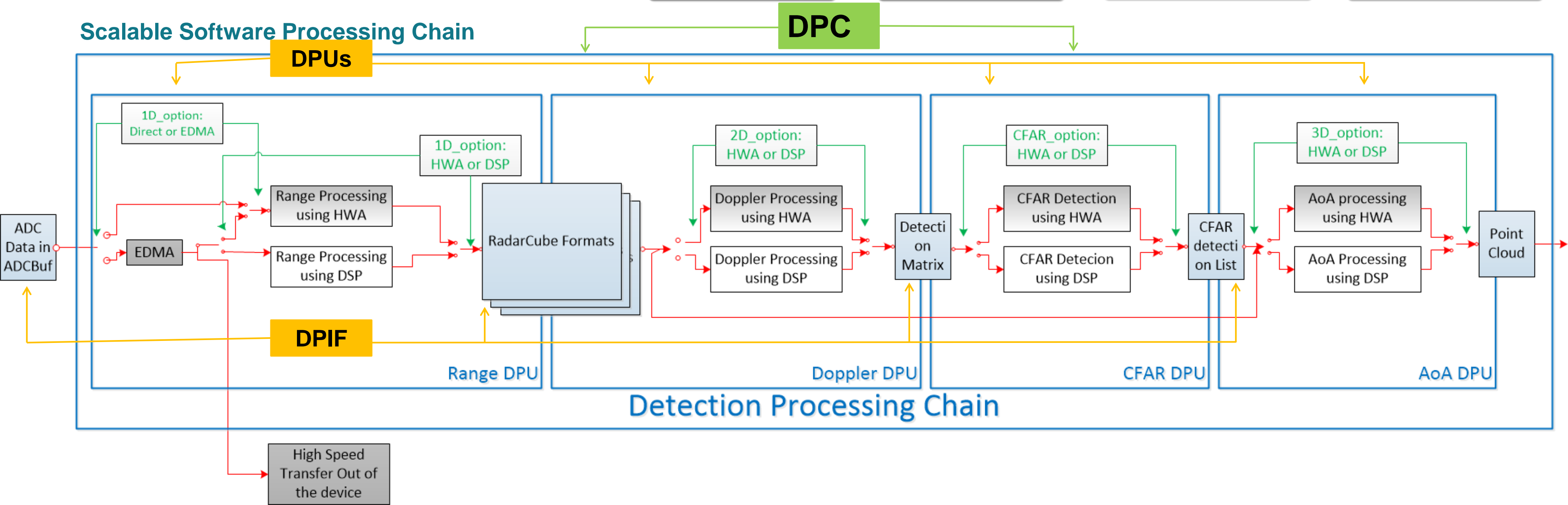


# Scalable SW Chain

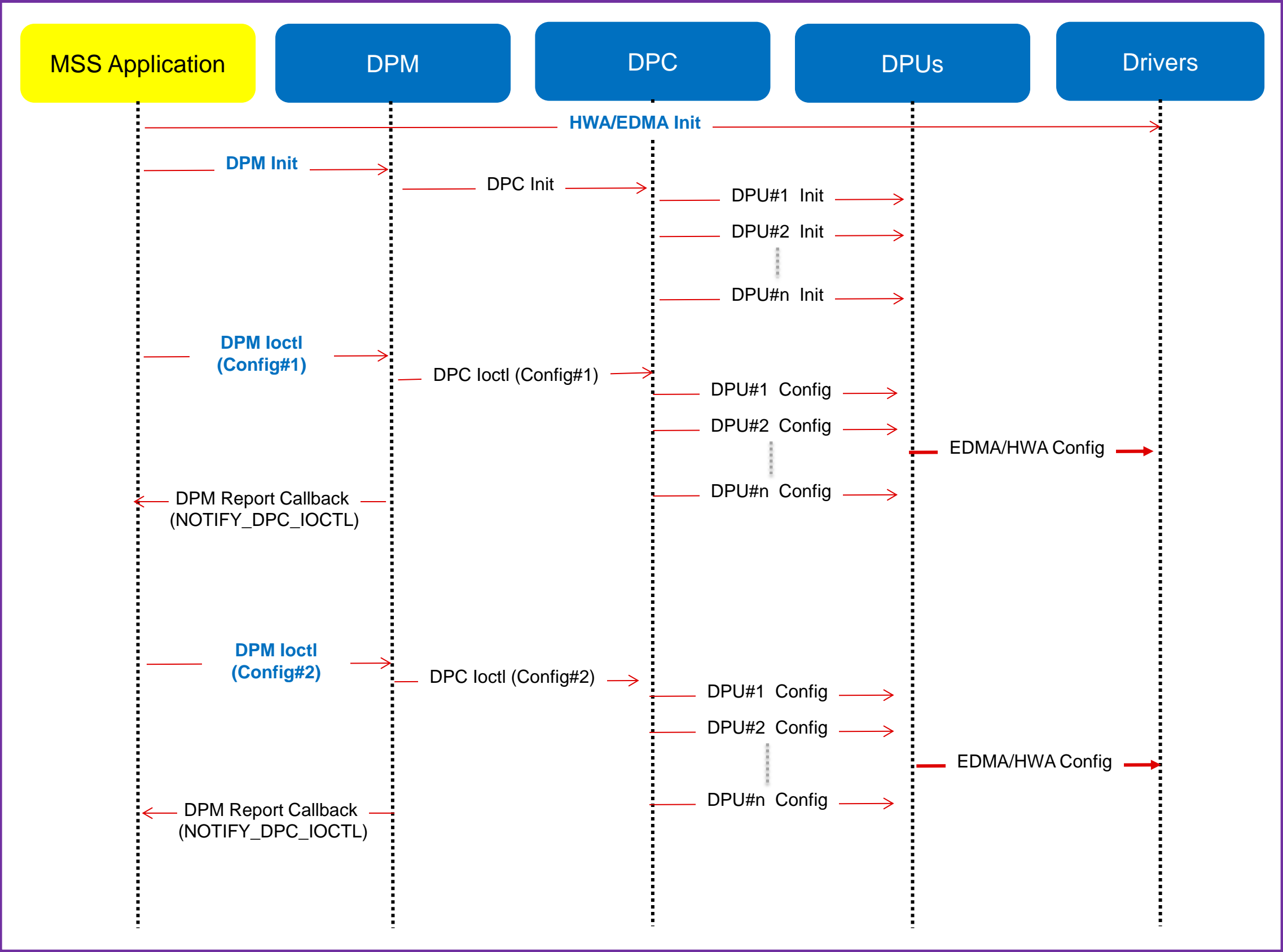
## TI mmWave Sensor Portfolio



## Scalable Software Processing Chain

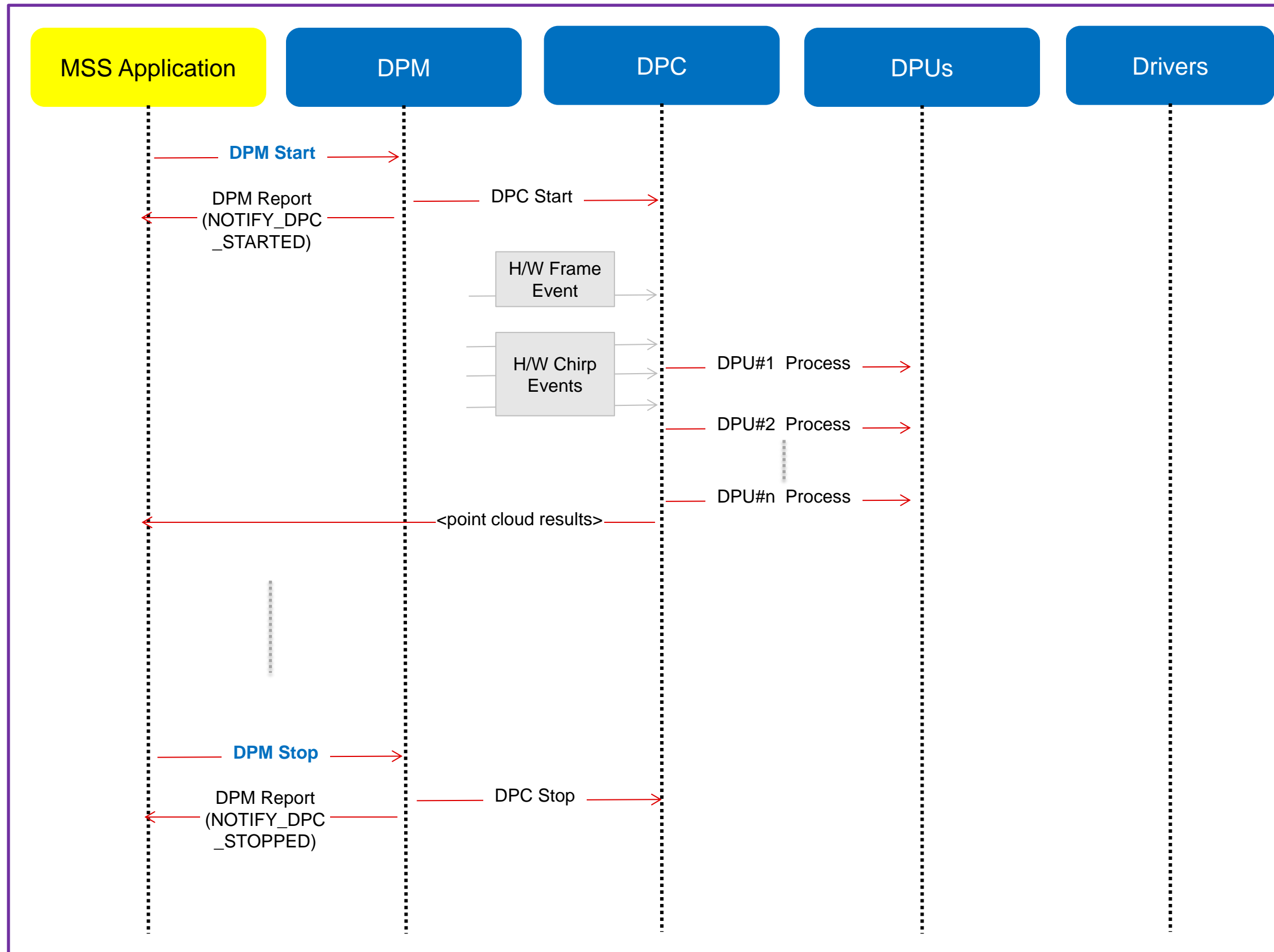


# Typical call flow (1/2)



Config#1 and Config#2 are shown as examples here on how a config can be split into multiple shorter structures as needed by a DPC. It is not mandatory to split into two. On the contrary, config can be split into even more smaller structures.

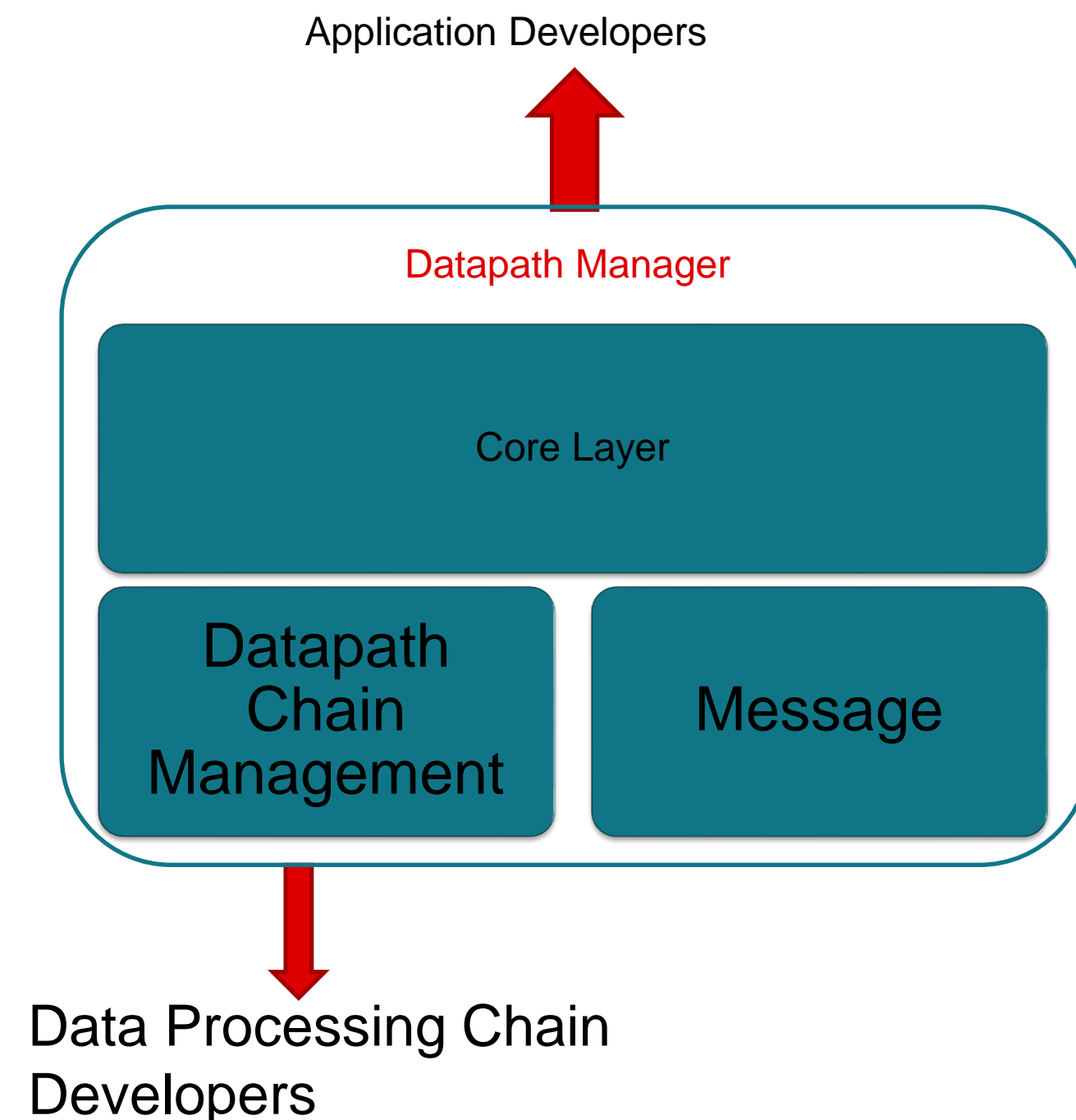
# Typical call flow (2/2)



# Data path deep dive

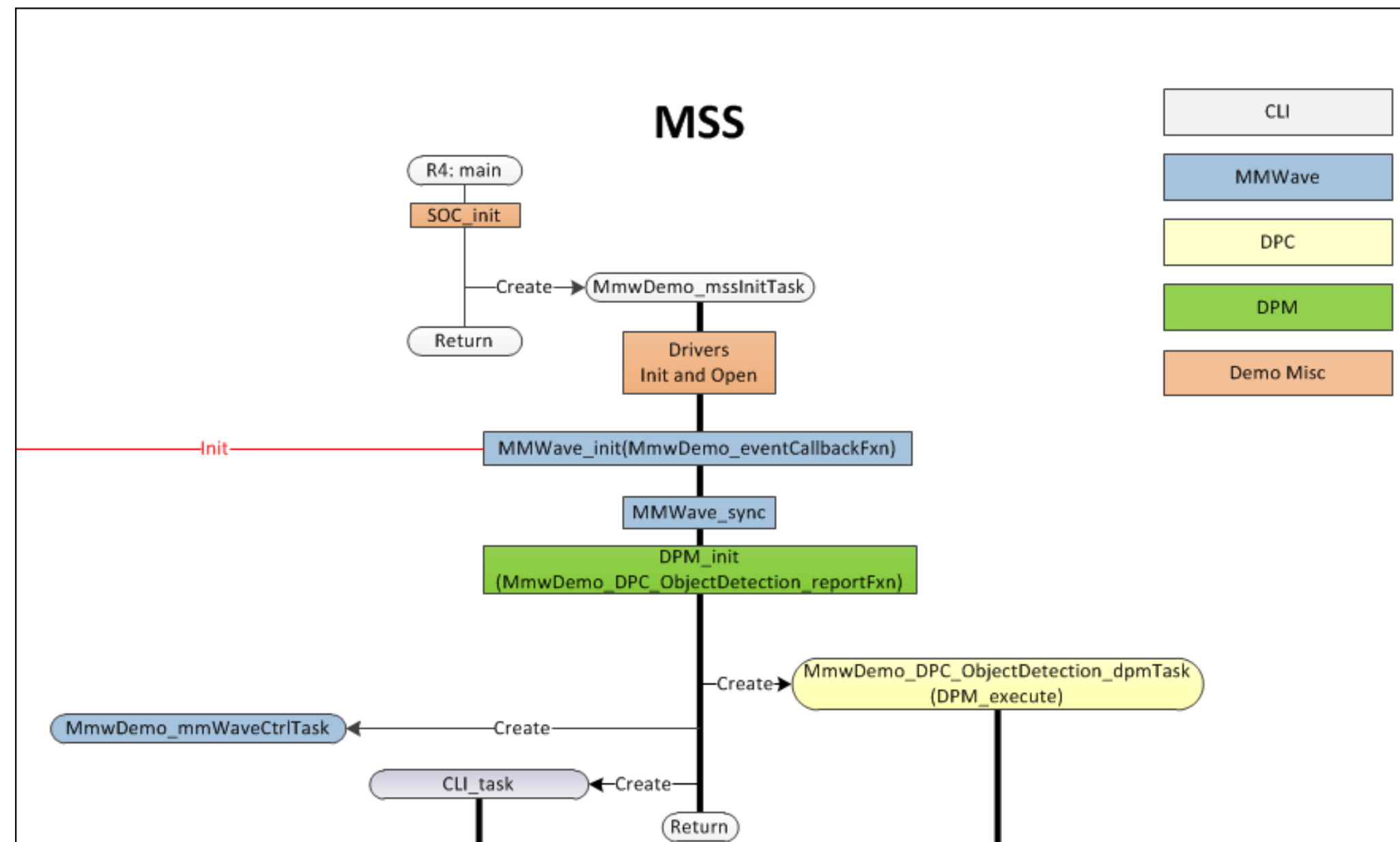
# DPM: Datapath Manager

- Modular SW Architecture which provides an abstraction between the “Datapath Processing Chain” and the customer application.
  - Main task context which encapsulates the execution of DPC and DPUs
  - Application code instantiates DPM at start-up and registers a DPC.
- Provides a well-defined API
  - Exposed to the application to interface with the DPM
  - Exposed to the “*Data processing chain*” developers to be able to write their own code.
- Messaging mechanism
  - Send/Receive Configuration
  - Extends to Multiple-Thread/Core
  - Synchronized execution (No critical section required)
  - Response Mechanism with error code passing
- Reporting mechanism which allows applications to be notified about the status of the DPM/Datapath Processing Chain.



# DPM Initialization

- Application code creates and initializes a DPM instance using the DPM\_init function
- Application also creates a DPM task



Source: [MMWAVE-SDK HTML documentation](#)

C:\ti\mmwave\_sdk\_03\_xx\_xx\_xx\docs\mmwave\_sdk\_module\_documentation.html



# DPM Initialization

- Application code creates and initializes a DPM instance using the DPM\_init function
- Application also creates a DPM task

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM
File Edit Tools Syntax Buffers Window Help
3230 /* Launch the DPM Task */
3231 Task_Params_init(&taskParams);
3232 taskParams.priority = MMWDEMO_DPC_OBJDET_DPM_TASK_PRIORITY;
3233 taskParams.stackSize = 4*1024;
3234 gMmwMCB.taskHandles.objDetDpmTask = Task_create(MmwDemo_DPC_ObjectDetection_dpmTask, &taskParams,
3235
3236 /******
```

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM
File Edit Tools Syntax Buffers Window Help
2732 * @b Description
2733 * @n
2734 * DPM Execution Task. DPM execute results are processed here:
2735 * a) Transmits results through UART port.
2736 * b) Updates book-keeping code for timing info.
2737 * c) Notifies DPC that results have been exported (using DPC IOCTL command)
2738 *
2739 * @retval
2740 * Not Applicable.
2741 */
2742 static void MmwDemo_DPC_ObjectDetection_dpmTask(UArg arg0, UArg arg1)
2743 {
2744     int32_t retVal;
2745     DPM_Buffer resultBuffer;
2746     DPC_ObjectDetection_ExecuteResultExportedInfo exportInfo;
2747     DPC_ObjectDetection_ExecuteResult *result;
2748
2749     while (1)
2750     {
2751         /* Execute the DPM module: */
2752         retVal = DPM_execute (gMmwMCB.dataPathObj.objDetDpmHandle, &resultBuffer);
2753         if (retVal < 0) {
```

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM
File Edit Tools Syntax Buffers Window Help
3185 /******
3186 * Initialization of the DPM Module:
3187 ******
3188 memset ((void *)&objDetInitParams, 0, sizeof(DPC_ObjectDetection_InitParams));
3189
3190 /* Note this must be after MmwDemo_dataPathOpen() above which opens the hwa
3191 * and edma drivers */
3192 objDetInitParams.hwaHandle = gMmwMCB.dataPathObj.hwaHandle;
3193 for (edmaCCIdx = 0; edmaCCIdx < EDMA_NUM_CC; edmaCCIdx++)
3194 {
3195     objDetInitParams.edmaHandle[edmaCCIdx] = gMmwMCB.dataPathObj.edmaHandle[edmaCCIdx];
3196 }
3197
3198 /* Memory related config */
3199 objDetInitParams.L3ramCfg.addr = (void *)&gMmwL3[0];
3200 objDetInitParams.L3ramCfg.size = sizeof(gMmwL3);
3201 objDetInitParams.CoreLocalRamCfg.addr = &gDPC_ObjDetTCM[0];
3202 objDetInitParams.CoreLocalRamCfg.size = sizeof(gDPC_ObjDetTCM);
3203
3204 /* Call-back config */
3205 objDetInitParams.processCallBackCfg.processFrameBeginCallBackFxn =
3206     MmwDemo_DPC_ObjectDetection_processFrameBeginCallBackFxn;
3207 objDetInitParams.processCallBackCfg.processInterFrameBeginCallBackFxn =
3208     MmwDemo_DPC_ObjectDetection_processInterFrameBeginCallBackFxn;
3209
3210 memset ((void *)&dpmInitCfg, 0, sizeof(DPM_InitCfg));
3211
3212 /* Setup the configuration: */
3213 dpmInitCfg.socHandle = gMmwMCB.socHandle;
3214 dpmInitCfg.ptrProcChainCfg = &gDPC_ObjectDetectionCfg;
3215 dpmInitCfg.instanceId = 0xFEEDFEED;
3216 dpmInitCfg.domain = DPM_Domain_LOCALIZED;
3217 dpmInitCfg.reportFxn = MmwDemo_DPC_ObjectDetection_reportFxn;
3218 dpmInitCfg.arg = &objDetInitParams;
3219 dpmInitCfg.argSize = sizeof(DPC_ObjectDetection_InitParams);
3220
3221 /* Initialize the DPM Module: */
3222 gMmwMCB.dataPathObj.objDetDpmHandle = DPM_init (&dpmInitCfg, &errCode);
3223 if (gMmwMCB.dataPathObj.objDetDpmHandle == NULL)
3224 {
3225     System_printf ("Error: Unable to initialize the DPM Module [Error: %d]\n", errCode);
3226     MmwDemo_debugAssert (0);
3227     return;
```



# DPC: Data Processing Chain

## DPC\_xxx\_APIS

- **DPC\_xxx\_init**
- **DPC\_xxx\_execute**
- **DPC\_xxx\_ioctl**
- **DPC\_xxx\_start**
- **DPC\_xxx\_injectData**
- **DPC\_xxx\_stop**
- **DPC\_xxx\_deinit**

## DPC\_xxx\_Callbacks

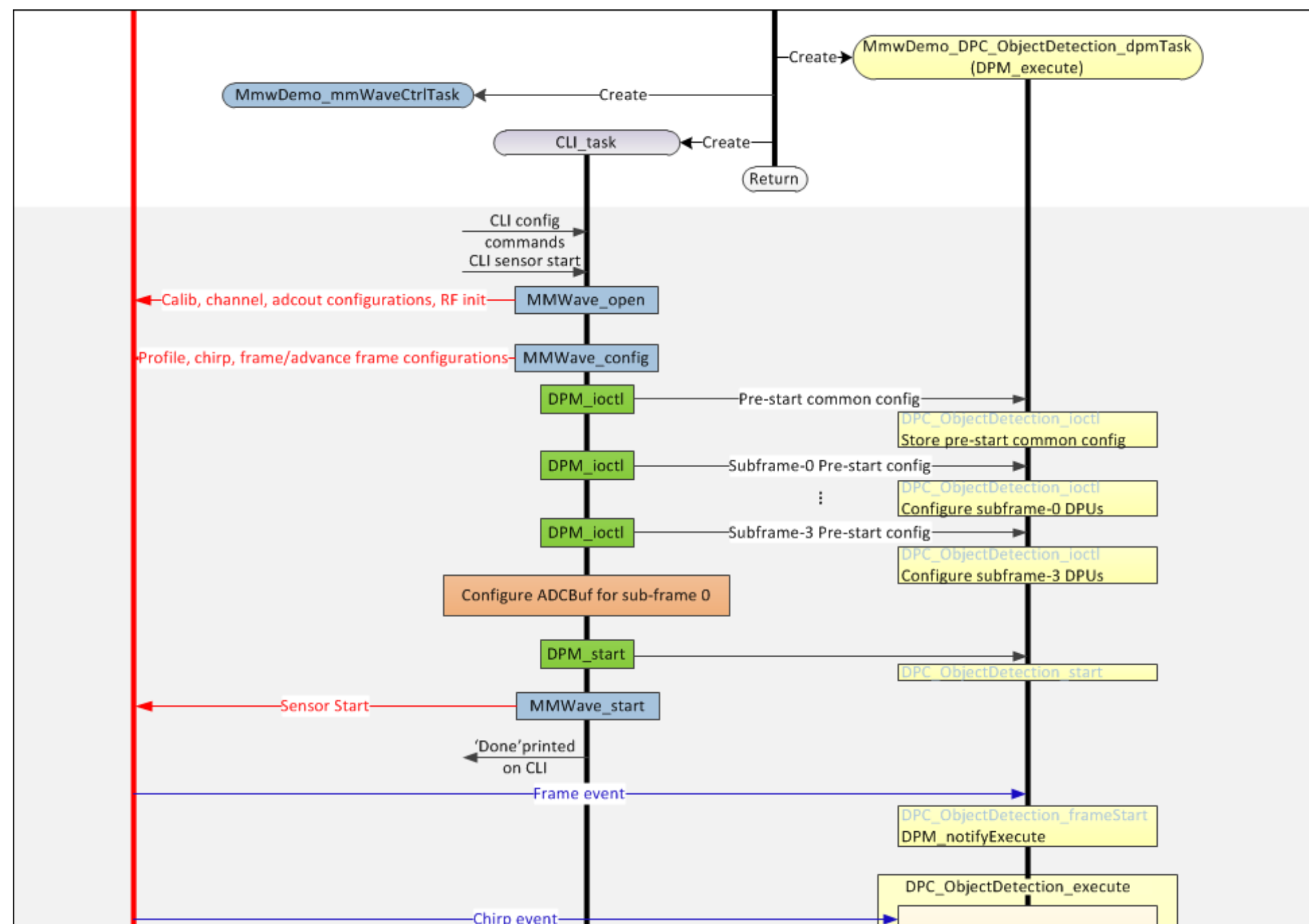
- **DPC\_xxx\_cbChirpAvailable**
- **DPC\_xxx\_cbFrameStart**

- All external DPC APIs starts with DPC\_. DPC unique name follows next (follows coding guidelines).
  - DPC\_ObjectDetection\_Init
- External Mandatory APIs follows the prototype defined by the “Datapath Manager”
- DPCs have flexibility in defining their own content within the individual structure
- DPC that is split between MSS and DSS will expose two set of APIs - one for MSS and one for DSS. Depending on the functionality split between the two domains, not all APIs need to be implemented on the two domains

For more details, refer to docs folder in each of the DPCs

# DPC Initialization

- Application registers the DPC with DPM during DPM creation.
- Application calls DPM init which calls the DPC's registered init function
- Application calls DPM\_ioctl with different message types
- DPM\_ioctl invokes the DPC's registered ioctl function with:
  - Pre-start common config message (common to all sub-frames) and
  - Sub-frame specific Pre-start config messages.
- DPC handles the messages in the ioctl function and performs the corresponding configuration.

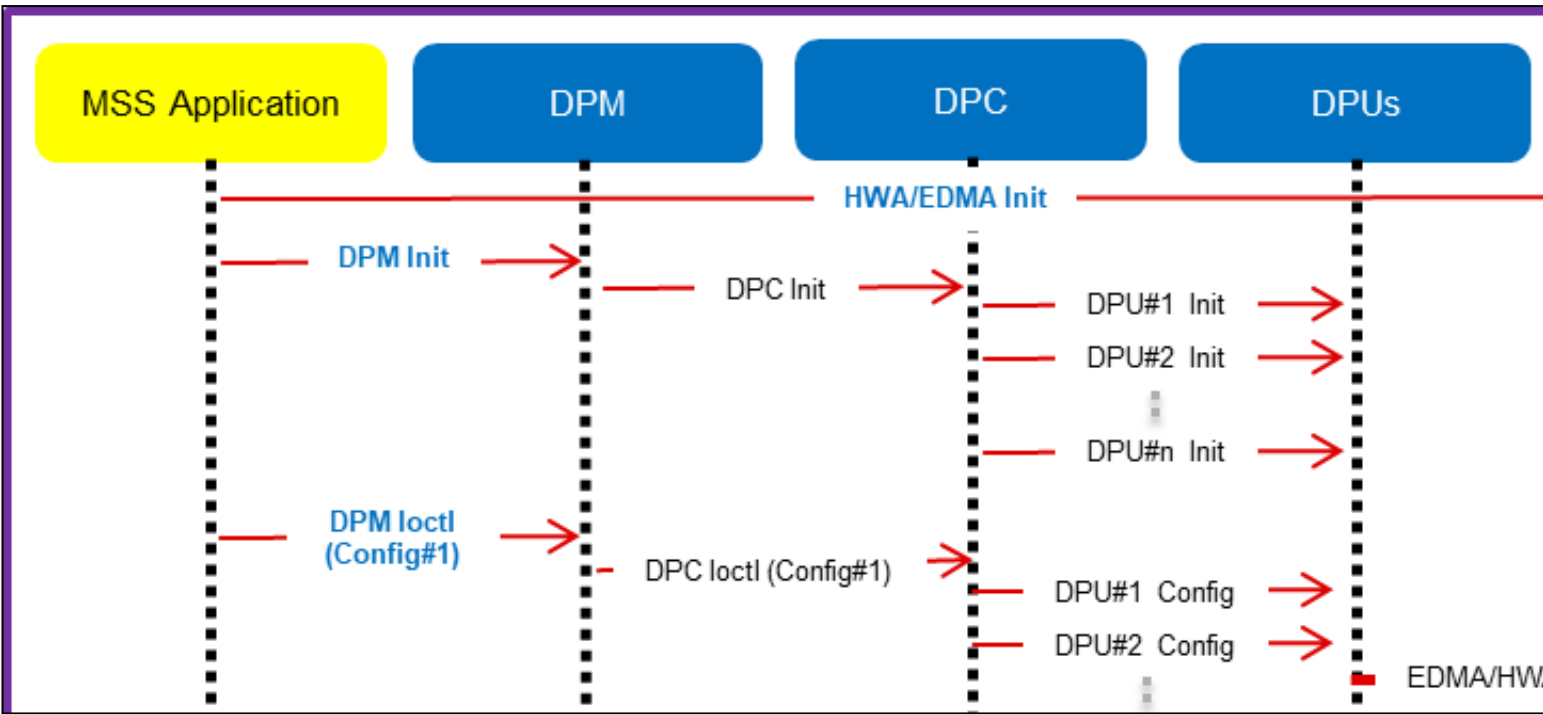


Source: MMWAVE-SDK HTML documentation

C:\ti\mmwave\_sdk\_03\_xx\_xx\_xx\docs\mmwave\_sdk\_module\_documentation.html

# DPC Initialization

- Application registers the DPC with DPM during DPM creation.
- Application calls DPM init which calls the DPC's registered init function
- Application calls DPM\_ioctl with different message types
- DPM\_ioctl internally invokes the DPC's registered ioctl function and passes the message to it
- DPC handles the message in the ioctl function and performs the corresponding configuration
- Refer to the call flow below

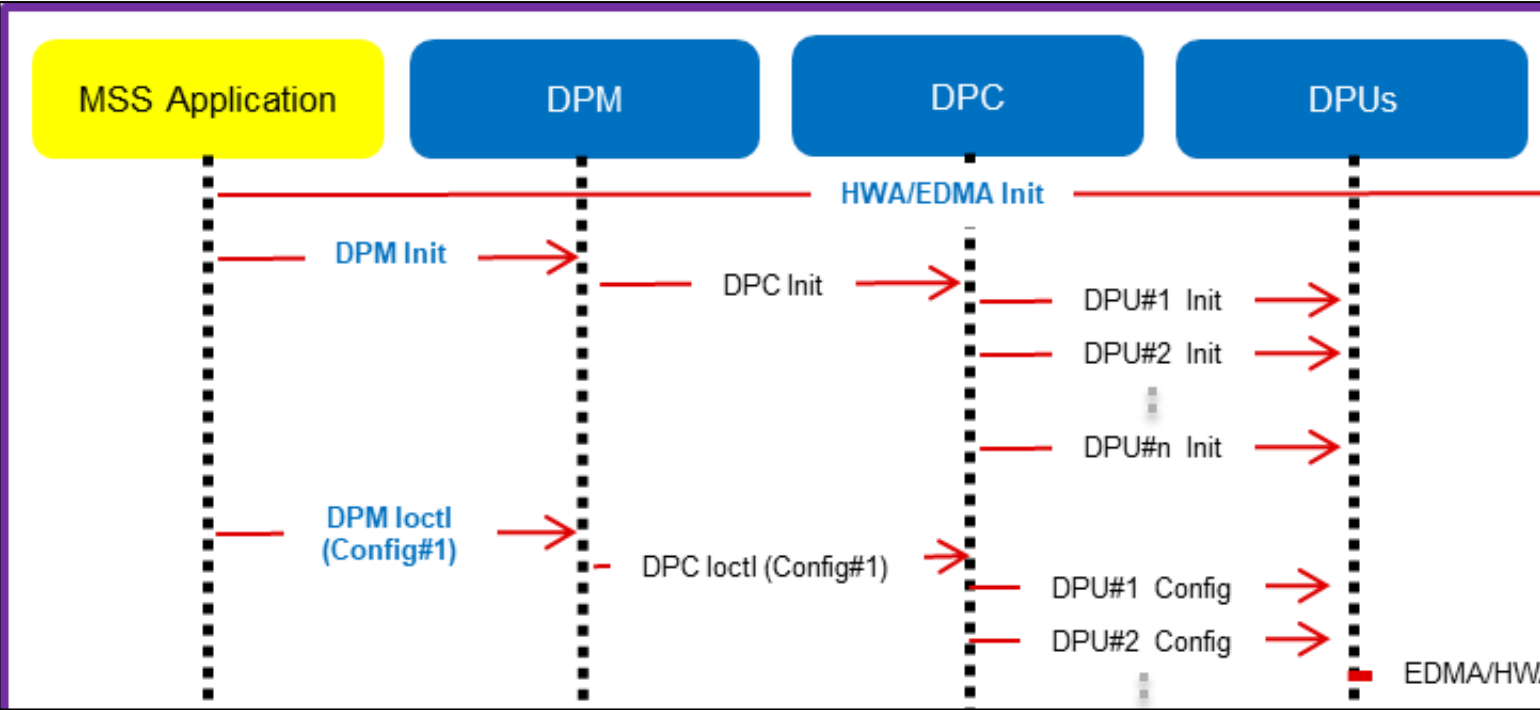


```
objectdetection.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\datapath\dpc\objectdetection\objdethwa\src) - GVIM
File Edit Tools Syntax Buffers Window Help
357  * @brief Global used to register Object Detection DPC in DPM
358  */
359  DPM_ProcChainCfg gDPC_ObjectDetectionCfg =
360  {
361      DPC_ObjectDetection_init,          /* Initialization Function:
362      DPC_ObjectDetection_start,         /* Start Function:
363      DPC_ObjectDetection_execute,       /* Execute Function:
364      DPC_ObjectDetection_ioctl,         /* Configuration Function:
365      DPC_ObjectDetection_stop,          /* Stop Function:
366      DPC_ObjectDetection_deinit,        /* Deinitialization Function:
367      NULL,                              /* Inject Data Function:
368      NULL,                              /* Chirp Available Function:
369      DPC_ObjectDetection_frameStart     /* Frame Start Function:
370  };
371
372  /* @} */
```

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM
File Edit Tools Syntax Buffers Window Help
3207  objDetInitParams.processCallBackCfg.processInterFrameBeginCallB
3208      MmwDemo_DPC_ObjectDetection_processInterFrameBeginCallB
3209
3210  memset ((void *)&dpmInitCfg, 0, sizeof(DPM_InitCfg));
3211
3212  /* Setup the configuration: */
3213  dpmInitCfg.socHandle      = gMmwMCB.socHandle;
3214  dpmInitCfg.ptrProcChainCfg = &gDPC_ObjectDetectionCfg;
3215  dpmInitCfg.instanceId     = 0xFEEDFEED;
3216  dpmInitCfg.domain         = DPM_Domain_LOCALIZED;
3217  dpmInitCfg.reportFxn       = MmwDemo_DPC_ObjectDetection_reportFxn;
3218  dpmInitCfg.arg             = &objDetInitParams;
3219  dpmInitCfg.argSize         = sizeof(DPC_ObjectDetection_InitParams);
3220
3221  /* Initialize the DPM Module: */
3222  gMmwMCB.dataPathObj.objDetDpmHandle = DPM_init (&dpmInitCfg, &errCode);
```

# DPC Configuration

- Application registers the DPC with DPM during DPM creation.
- Application calls DPM init which calls the DPC's registered init function
- Application calls DPM\_ioctl with different message types (i.e. commands)
- DPM\_ioctl internally invokes the DPC's registered ioctl function and passes the message to it
- DPC handles the message in the ioctl function and performs the corresponding configuration
- Refer to the call flow below



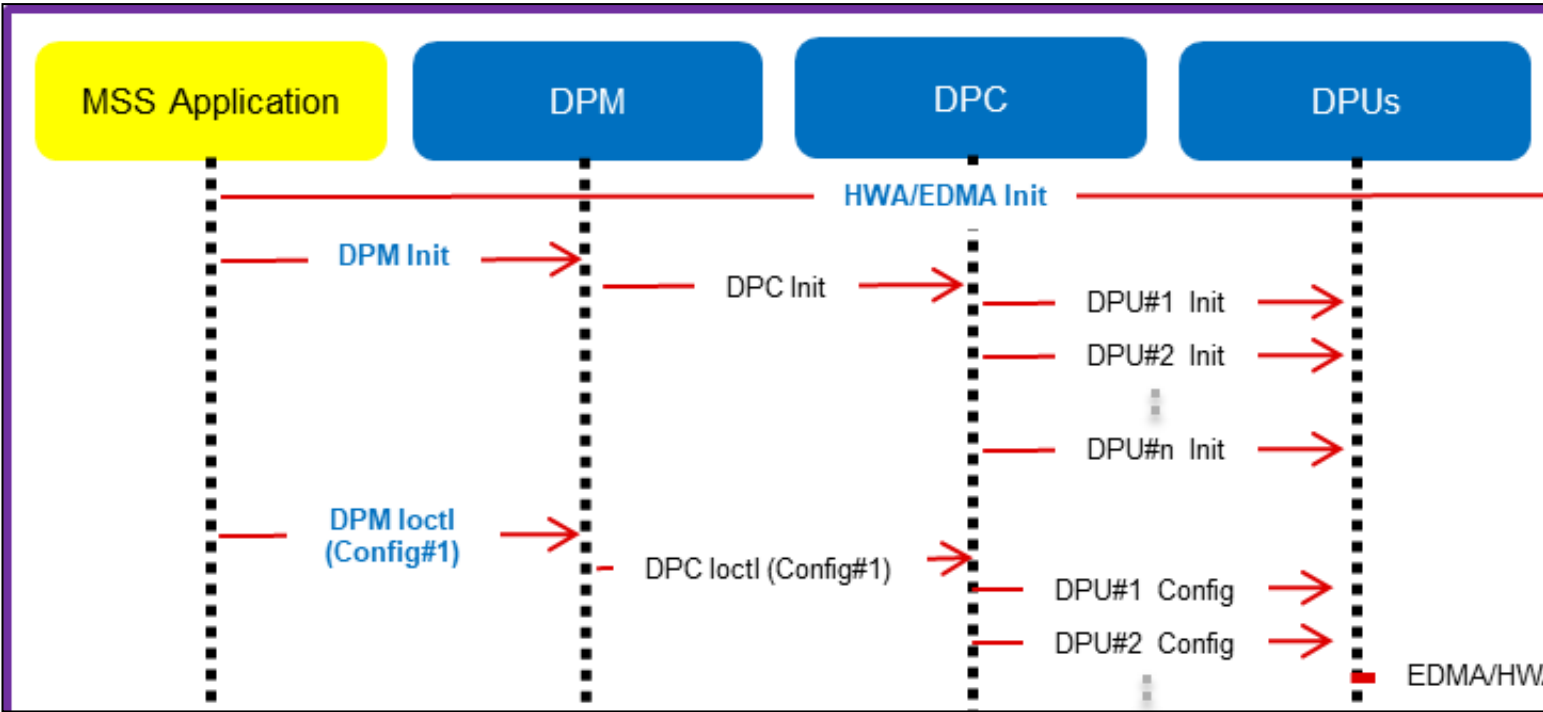
```
mmw_cli.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM1
File Edit Tools Syntax Buffers Window Help
125 static int32_t MmwDemo_CLISensorStart (int32_t argc, char* argv[])
126 {
127     bool doReconfig = true;
128     int32_t retVal = 0;
129
304     /* if MmwDemo_openSensor has non-first time related processing, call here ag
305     /* call sensor config */
306     CLI_getMMWaveExtensionConfig (&gMmwMCB.cfg.ctrlCfg);
307     retVal = MmwDemo_configSensor();
308     if(retVal != 0)
```

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM
File Edit Tools Syntax Buffers Window Help
1206 System_printf ("Error: mmWave Config failed [Error code: %d Subsystem: %d]\n
1207 mmWaveErrorCode, subsysErrorCode);
1208 goto exit;
1209 }
1210 else
1211 {
1212     errCode = MmwDemo_dataPathConfig();
1213     goto exit;
1214 }
1215
1572 /* DPC pre-start common config */
1573 errCode = DPM_ioctl (dataPathObj->objDetDpmHandle,
1574 DPC_OBJDET_IOCTL_STATIC_PRE_START_COMMON_CFG,
1575 &dataPathObj->objDetCommonCfg.preStartCommonCfg,
1576 sizeof (DPC_ObjectDetection_PreStartCommonCfg));
1577
1578 if (errCode < 0)
1579 {
1580     System_printf ("Error: Unable to send DPC_OBJDET_IOCTL_STATIC_PRE_START_COM
1581     goto exit;
1582 }
```



# DPC Configuration (continued)

- Application registers the DPC with DPM during DPM creation.
- Application calls DPM init which calls the DPC's registered init function
- Application calls DPM\_ioctl with different message types
- DPM\_ioctl internally invokes the DPC's registered ioctl function and passes the message to it
- DPC handles the message in the ioctl function and performs the corresponding configuration
- Refer to the call flow below



5

```
objectdetection.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\datapath\dpc\objectdetection\objdethwa\src) - GVIM
File Edit Tools Syntax Buffers Window Help
2350 */
2351 static int32_t DPC_ObjectDetection_ioctl
2352 (
2353     DPM_DPCHandle    handle,
2354     uint32_t          cmd,
2355     void*             arg,
2356     uint32_t          argLen
2357 )
2358 {
2359     ObjDetObj *objDetObj;
2360     SubFrameObj *subFrmObj;
2361     int32_t retVal = 0;
2362
2363     /* Get the DSS MCB: */
2364     objDetObj = (ObjDetObj *) handle;
2365     DebugP_assert(objDetObj != NULL);
2366
2367     /* Process the commands. Process non sub-frame specific ones first
2368      * so the sub-frame specific ones can share some code. */
2369     if (cmd == DPC_OBJDET_IOCTL_TRIGGER_FRAME)
2370     {
2371         DPC_ObjectDetection_frameStart(handle);
2372     }
2373     else if (cmd == DPC_OBJDET_IOCTL_STATIC_PRE_START_COMMON_CFG)
2374     {
2375         DPC_ObjectDetection_PreStartCommonCfg *cfg;
2376         int32_t indx;
```

6

```
objectdetection.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\datapath\dpc\objectdetection\objdethwa\src) - GVIM
File Edit Tools Syntax Buffers Window Help
2415     }
2416 }
2417 else if (cmd == DPC_OBJDET_IOCTL_DYNAMIC_COMP_RANGE_BIAS_AND_RX_CHAN_PH
2418 {
2419     DPU_AoAProc_compRxChannelBiasCfg *inpCfg;
2420     DPU_AoAProc_compRxChannelBiasCfg outCfg;
2421     int32_t i;
2422
2423     DebugP_assert(argLen == sizeof(DPU_AoAProc_compRxChannelBiasCfg));
2424
2425     inpCfg = (DPU_AoAProc_compRxChannelBiasCfg*)arg;
2426
2427     for(i = 0; i < objDetObj->commonCfg.numSubFrames; i++)
```

# DPU: Data Processing Units

## DPU\_xxx\_init

- DPU\_xxx\_InitParams\_t
- errCode
- Handle

## DPU\_xxx\_config

- DPU\_Handle
- DPU\_xxx\_Config\_t
  - H/W Resources (EDMA, HWA, I/O buffer pointers, Scratch buffer pointers)
  - Frame/Sub-frame DPU Static Config (Ex: Num ADC Samples, Chirps/Frame, ADCBuf Config, Data Interface Desc)
  - Frame/Sub-frame DPU Dynamic Config (Ex: DC Range Calibration)
- errCode

## DPU\_xxx\_process

- DPU\_Handle
- DPU\_xxx\_OutParams\_t
- DPU\_xxx\_Stats\_t
- DPU optional specific Params (Ex: isLastChirp)
- errCode

## DPU\_xxx\_control

- DPU\_Handle
- cmd
- args
- argSize
- errCode

## DPU\_xxx\_deinit

- DPU\_handle
- errCode

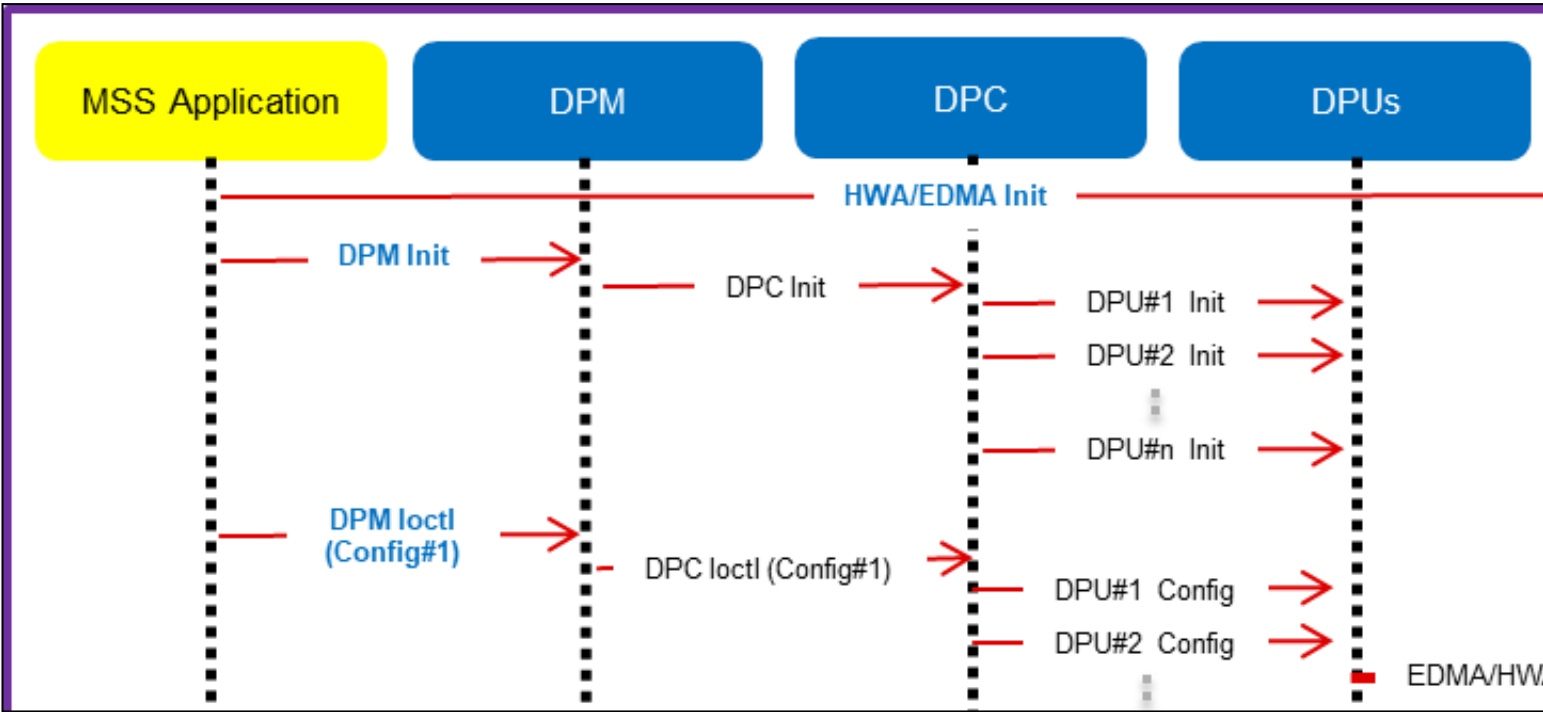
- All external DPU APIs start with the prefix DPU\_. DPU unique name follows next..Ex: DPU\_RangeProchWA\_init
- Standard external APIs:
  - **Init**: one time initialization of DPU
  - **Config**: complete configuration of the DPU: hardware resources, static and dynamic (if supported by DPU)
    - **static config**: config that is static during ongoing frames
    - **dynamic config**: config that can be changed from frame to frame but only when process is not ongoing - ideally interframe time after DPC has exported the results for the frame
  - **Process**: the actual processing function of the DPU
  - **Control**: ioctl interface that allows higher layer to switch dynamic configuration during interframe time
  - **De-init**: de-initialization of DPU
- All memory allocations for I/O buffers and scratch buffers are outside the DPU since mmWave applications rely on memory overlay technique for optimization and that is best handled at application level
- All H/W resources must be allocated by application and passed to the DPU. This helps in keeping DPU platform agnostic as well as allows application to share the resources across DPU when DPU processing doesn't overlap in time.
- DPUs are OS agnostic and use OSAL APIs for needed OS services.

For more details, refer to docs folder in each of the DPUs

# DPU Initialization

Recall from DPC Configuration...

- Application calls DPM init which calls the DPC's registered init function
- DPC\_init calls the various DPU init functions in sequence, to create separate DPU instances for the number of sub-frames configured.
- Refer to the call flow below

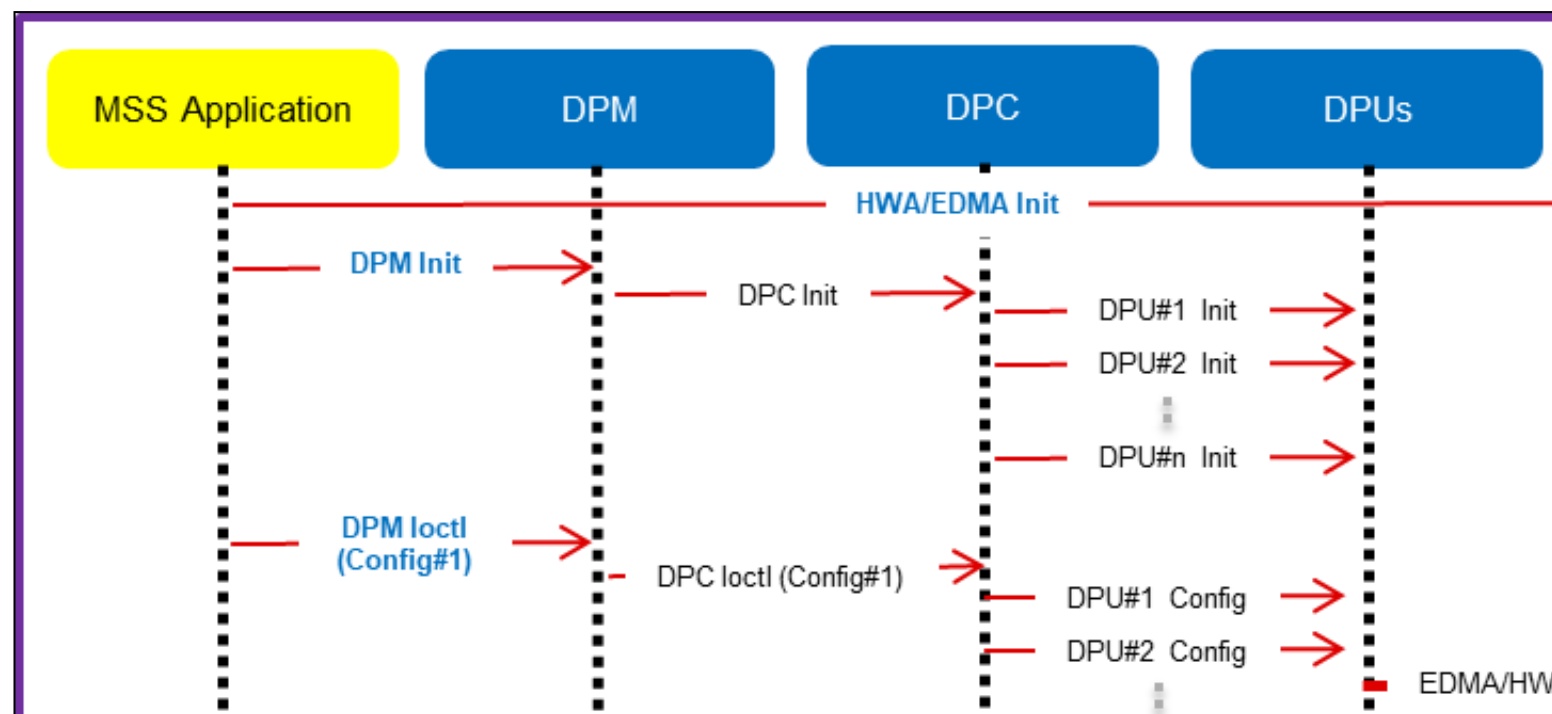


```
objectdetection.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\datapath\dpc\objectdetection\objdethwa\src) - GVIM
File Edit Tools Syntax Buffers Window Help
2825 *      Error      -      <0
2826 */
2827 static DPM_DPCHandle DPC_ObjectDetection_init
2828 {
2829     DPM_Handle      dpmHandle,
2830     DPM_InitCfg*    ptrInitCfg,
2831     int32_t*        errCode
2832 }
2833 {
2834     int32_t i;
2835     ObjDetObj      *objDetObj = NULL;
2836     SubFrameObj     *subFrmObj;
2837     DPC_ObjectDetection_InitParams *dpcInitParams;
2838     DPU_RangeProcHWA_InitParams rangeInitParams;
2839     DPU_AoAProcHWA_InitParams aoaInitParams;
2840     DPU_CFARCAProcHWA_InitParams cfarInitParams;
2841     DPU_DopplerProcHWA_InitParams dopplerInitParams;
2842     HWA_MemInfo     hwaMemInfo;
2843
2844     rangeInitParams.hwaHandle = dpcInitParams->hwaHandle;
2845     aoaInitParams.hwaHandle = dpcInitParams->hwaHandle;
2846     cfarInitParams.hwaHandle = dpcInitParams->hwaHandle;
2847     dopplerInitParams.hwaHandle = dpcInitParams->hwaHandle;
2848
2849     for(i = 0; i < RL_MAX_SUBFRAMES; i++)
2850     {
2851         subFrmObj = &objDetObj->subFrameObj[i];
2852         subFrmObj->dpuRangeObj = DPU_RangeProcHWA_init(&rangeInitParams, errCode);
2853
2854         if (*errCode != 0)
2855         {
2856             goto exit;
2857         }
2858
2859         subFrmObj->dpuStaticClutterObj = DPU_StaticClutterProc_init(errCode);
2860
2861         if (*errCode != 0)
2862         {
2863             goto exit;
2864         }
2865
2866         subFrmObj->dpuCFARCAObj = DPU_CFARCAProcHWA_init(&cfarInitParams, errCode);
2867
2868         if (*errCode != 0)
2869         {
2870             goto exit;
2871         }
2872     }
2873 }
```

# DPU Configuration

Recall from DPC Configuration...

- Application calls DPM\_ioctl with different message types
- DPM\_ioctl internally invokes the DPC's registered ioctl function and passes the message to it
- DPC performs DPU configuration when handling the PRE\_START\_CFG command
  - The PRE\_START\_CFG handler calls wrapper functions for each DPU, e.g. DPC\_ObjDet\_rangeConfig
  - The wrapper allocates the resources required for the DPU and calls the corresponding DPU config function e.g. DPU\_RangeProchWA\_config.
- Refer to the call flow below



```

objectdetection.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\datapath\dpc\objectdetection\objdethwa\src) - GVIM
File Edit Tools Syntax Buffers Window Help
2737
2738     /* Related to pre-start configuration */
2739     case DPC_OBJDET_IOCTL_STATIC_PRE_START_CFG:
2740     {
2741         DPC_ObjectDetection_PreStartCfg *cfg;
2742         DPC_ObjectDetection_DPC_IOCTL_preStartCfg_memUsage *memUsage;
2743         MemoryP_Stats statsStart;
2744         MemoryP_Stats statsEnd;
2745
2746         /* Pre-start common config must be received before pre-start configs
2763         memUsage->CoreLocalRamTotal = objDetObj->CoreLocalRamObj.cfg.size;
2764         retVal = DPC_ObjDet_preStartConfig(subFrmObj,
2765                                             &objDetObj->commonCfg, &cfg->staticCfg, &cfg->dynCfg,
2766                                             &objDetObj->edmaHandle[0],
2767                                             &objDetObj->L3RamObj,
2768                                             &objDetObj->CoreLocalRamObj,
2769                                             &objDetObj->hwaMemBankAddr[0],
2232
2233         retVal = DPC_ObjDet_rangeConfig(obj->dpuRangeObj, &obj->staticCfg, &obj->dynCfg,
2234                                         edmaHandle[DPC_OBJDET_DPU_RANGEPROC_EDMA_INST_ID],
2235                                         &radarCube, CoreLocalRamObj, &hwaWindowOffset,
2236                                         &rangeCoreLocalRamScratchUsage, &obj->dpuCfg.rangeCfg);
2237         if (retVal != 0)
1408         hwaCfg->paramSetStartIdx = DPC_OBJDET_DPU_RANGEPROC_PARAMSET_START_IDX;
1409
1410         retVal = DPU_RangeProchWA_config(dpuHandle, &rangeCfg);
1411         if (retVal != 0)
1412         {
1413             goto exit;
1414         }
1415
1416         /* store configuration for use in intra-sub-frame processing and
1417         * inter-sub-frame switching, although window will need to be regenerated and
1418         * dc range sig should not be reset. */
1419         rangeCfg.staticCfg.resetDcRangeSigMeanBuffer = 0;
1420         *cfgSave = rangeCfg;
1421
    
```



# DPIF: Datapath Interface

## Input ADC data

- Property
  - numADCsSamples
  - RX interleaved/non-interleaved
  - Complex/Real
- Buffer Pointer

## Radar Cube

- Property
  - Layout – RADAR CUBE RANGE DOPPLER RX TX (1)
- Buffer Pointer

## Detection Matrix

- Property
  - Layout – DET MATRIX RANGE DOPPLER (1)
- Buffer Pointer

## Point cloud

- Property
  - Format – XYZV, RAEV
- Float

## Point Cloud SidelInfo

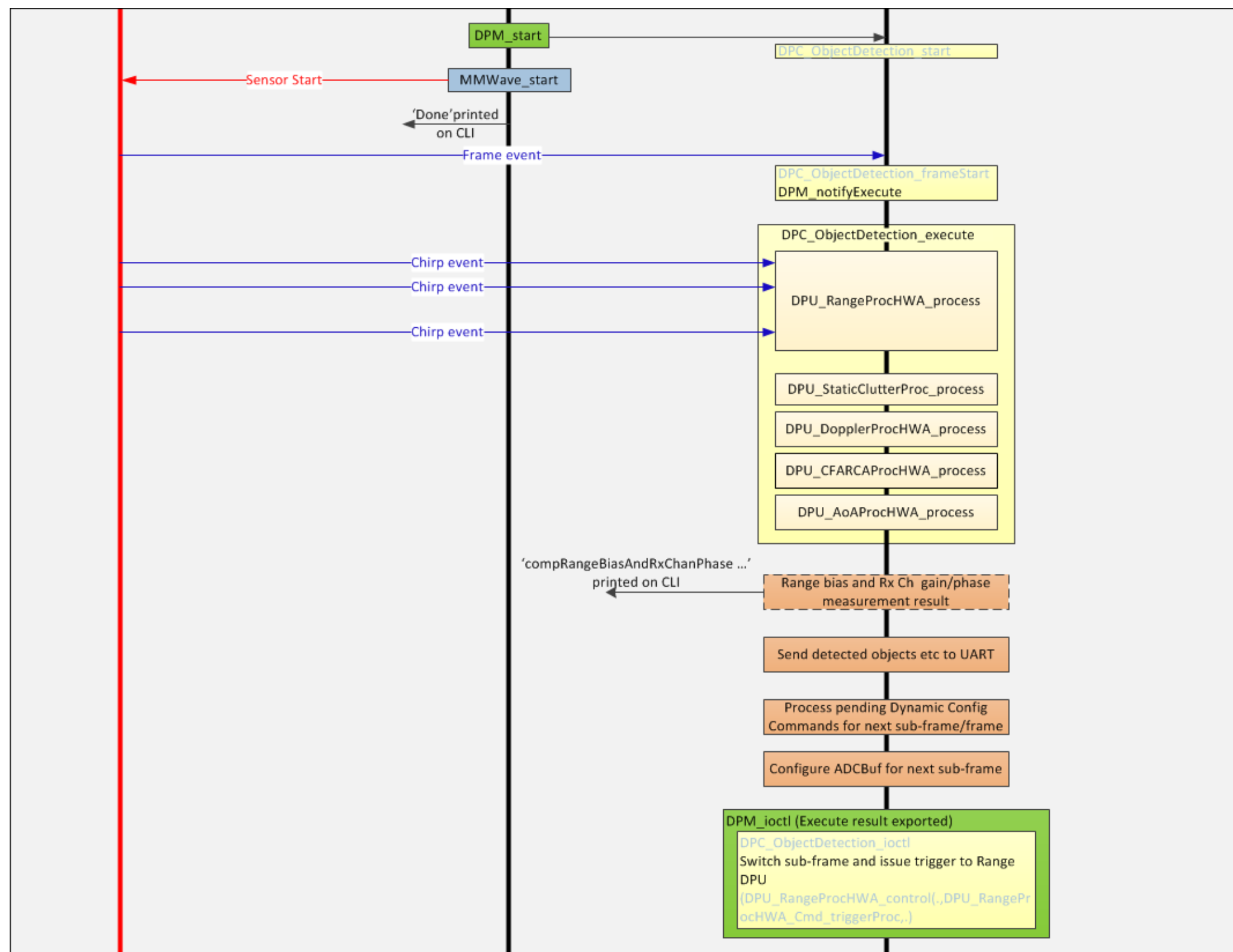
- Property
  - snr
  - noiseVal

```
dpif_adcdata.h (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\datapath\dpif) - GVIM
File Edit Tools Syntax Buffers Window Help
108 /**
109  * @brief
110  *   ADC Data buffer definition
111  *
112  * @details
113  *   The structure defines the ADC data buffer ,including data property, data size and
114  */
115 typedef struct DPIF_ADCBufData_t
116 {
117     /*! @brief ADCBuf data property */
118     DPIF_ADCBufProperty    dataProperty;
119
120     /*! @brief ADCBuf  buffer size in bytes */
121     uint32_t                dataSize;
122
123     /*! @brief ADCBuf data pointer */
124     void                    *data;
125 }DPIF_ADCBufData;
```

```
dpif_pointcloud.h (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\datapath\dpif) - GVIM1
File Edit Tools Syntax Buffers Window Help
56 * Point cloud definition in Cartesian coordinate system
57 */
58 typedef struct DPIF_PointCloudCartesian_t
59 {
60     /*! @brief x - coordinate in meters. This axis is parallel to the sensor plane
61      *        and makes the azimuth plane with y-axis. Positive x-direction is right
62      *        in the azimuth plane when observed from the sensor towards the scene
63      *        and negative is the opposite direction. */
64     float x;
65
66     /*! @brief y - coordinate in meters. This axis is perpendicular to the
67      *        sensor plane with positive direction from the sensor towards the scene
68      */
69     float y;
70
71     /*! @brief z - coordinate in meters. This axis is parallel to the sensor plane
72      *        and makes the elevation plane with the y-axis. Positive z direction
73      *        is above the sensor and negative below the sensor */
74     float z;
75
76     /*! @brief Doppler velocity estimate in m/s. Positive velocity means target
77      *        is moving away from the sensor and negative velocity means target
78      *        is moving towards the sensor. */
79     float velocity;
80 }DPIF_PointCloudCartesian;
```

# Datapath Execution – Runtime View

- **At sensor start:** Application calls DPM\_start which calls the DPC's pre-registered start function DPC\_ObjectDetection\_start.
  - The mmw demo does this in the sensorStart CLI command handler function.
- **At every frame interrupt:** The DPC's pre-registered frame start call back function, DPC\_ObjectDetection\_frameStart is invoked.
  - **Recall from DPC\_init:** The DPC's frameStart callback is registered with the DPM during DPC initialization
- This invokes the DPC's registered execute function, e.g. DPC\_ObjectDetection\_execute
  - **Under the hood:** The Frame Start handler calls DPM\_notifyExecute which posts the semaphore for DPM\_execute
  - DPM\_execute then calls the DPC's pre-registered execute function.
- DPC\_ObjectDetection\_execute performs the frame processing and populates the output in DPC\_ObjectDetection\_ExecuteResult structure and returns.
- Application reads the output structure and sends it over UART. This completes the frame processing.



# Datapath Execution – Runtime View

- **At sensor start:** Application calls DPM\_start which calls the DPC's pre-registered start function DPC\_ObjectDetection\_start.
  - The mmw demo does this in the sensorStart CLI command handler function.
- **At every frame interrupt:** The DPC's pre-registered frame start call back function, DPC\_ObjectDetection\_frameStart is invoked.
  - **Recall from DPC\_init:** The DPC's frameStart callback is registered with the DPM during DPC initialization
- This invokes the DPC's registered execute function, e.g. DPC\_ObjectDetection\_execute
  - **Under the hood:** The Frame Start handler calls DPM\_notifyExecute which posts the semaphore for DPM\_execute
  - DPM\_execute then calls the DPC's pre-registered execute function.
- DPC\_ObjectDetection\_execute performs the frame processing and populates the output in DPC\_ObjectDetection\_ExecuteResult structure and returns.
- Application reads the output structure and sends it over UART. This completes the frame processing.

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM1
File Edit Tools Syntax Buffers Window Help
1223 *      mmw demo helper Function to start sensor.
1224 *
1225 * @retval 0 if no error, -1 if error
1226 */
1227 int32_t MmwDemo_startSensor(void)
1228 {
1229     int32_t    errCode;
1230     MMWave_CalibrationCfg  calibrationCfg;
1231
1232     /*****
1233      * Data path :: start data path first - this will pend for DPC to ack
1234      *****/
1235     MmwDemo_dataPathStart();
```

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM
File Edit Tools Syntax Buffers Window Help
1786 void MmwDemo_dataPathStart (void)
1787 {
1788     int32_t errCode;
1789
1790     DebugP_log0("App: Issuing DPM_start\n");
1791
1792     /* Configure HW LVDS stream for the first sub-frame that will start upon
1793      * start of frame */
1794     if (gMmwMCB.subFrameCfg[0].lvdsStreamCfg.dataFmt != MMW_DEMO_LVDS_STREAM_CFG_DAT
1795     {
1796         MmwDemo_configLVDSHwData(0);
1797     }
1798
1799     /* Start the DPM Profile: */
1800     if ((errCode = DPM_start(gMmwMCB.dataPathObj.objDetDpmHandle)) < 0)
1801     {
```

# Datapath Execution – Runtime View

- **At sensor start:** Application calls DPM\_start which calls the DPC's pre-registered start function DPC\_ObjectDetection\_start.
  - The mmw demo does this in the sensorStart CLI command handler function.
- **At every frame interrupt:** The DPC's pre-registered frame start call back function, DPC\_ObjectDetection\_frameStart is invoked.
  - **Recall from DPC\_init:** The DPC's frameStart callback is registered with the DPM during DPC initialization
- This invokes the DPC's registered execute function, e.g. DPC\_ObjectDetection\_execute
  - **Under the hood:** The Frame Start handler calls DPM\_notifyExecute which posts the semaphore for DPM\_execute
  - DPM\_execute then calls the DPC's pre-registered execute function.
- DPC\_ObjectDetection\_execute performs the frame processing and populates the output in DPC\_ObjectDetection\_ExecuteResult structure and returns.
- Application reads the output structure and sends it over UART. This completes the frame processing.

```
objectdetection.c (C:\ti\mmwave_sdk_03_04_00_03\packa...ti\datapath\dpc\objectdetection\objdethwa\src) - GVIM1
File Edit Tools Syntax Buffers Window Help
417 */
418 static void DPC_ObjectDetection_frameStart (DPM_DPCHandle handle)
419 {
420     ObjDetObj *objDetObj = (ObjDetObj *) handle;
421     objDetObj->stats.frameStartTimeStamp = Cycleprofiler_getTimeStamp();
422     DebugP_log2("ObjDet DPC: Frame Start, frameIndx = %d, subFrameIndx = %d\n",
423         objDetObj->stats.frameStartIntCounter, objDetObj->subFrameIndx);
424     /* Check if previous frame (sub-frame) processing has completed */
425     DPC_Objdet_Assert(objDetObj->dpmHandle, (objDetObj->interSubFrameProcToken == 0)
426         objDetObj->interSubFrameProcToken++);
427     /* Increment interrupt counter for debugging and reporting purpose */
428     if (objDetObj->subFrameIndx == 0)
429     {
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871
2872
2873
2874
2875
2876
2877
2878
2879
2880
2881
2882
2883
2884
2
```



# Datapath Execution – Runtime View

- **At sensor start:** Application calls DPM\_start which calls the DPC's pre-registered start function DPC\_ObjectDetection\_start.
  - The mmw demo does this in the sensorStart CLI command handler function.
- **At every frame interrupt:** The DPC's pre-registered frame start call back function, DPC\_ObjectDetection\_frameStart is invoked.
  - **Recall from DPC\_init:** The DPC's frameStart callback is registered with the DPM during DPC initialization
- This invokes the DPC's registered execute function, e.g. DPC\_ObjectDetection\_execute
  - **Under the hood:** The Frame Start handler calls DPM\_notifyExecute which posts the semaphore for DPM\_execute
  - DPM\_execute then calls the DPC's pre-registered execute function.
- DPC\_ObjectDetection\_execute performs the frame processing and populates the output in DPC\_ObjectDetection\_ExecuteResult structure and returns.
- Application reads the output structure and sends it over UART. This completes the frame processing.

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM
File Edit Tools Syntax Buffers Window Help
2735 *      a) Transmits results through UART port.
2736 *      b) Updates book-keeping code for timing info.
2737 *      c) Notifies DPC that results have been exported (using DPC IOCTL command)
2738 *
2739 *      @retval
2740 *      Not Applicable.
2741 */
2742 static void MmwDemo_DPC_ObjectDetection_dpmTask(UArg arg0, UArg arg1)
2743 {
2744     int32_t      retVal;
2745     DPM_Buffer    resultBuffer;
2746     DPC_ObjectDetection_ExecuteResultExportedInfo exportInfo;
2747     DPC_ObjectDetection_ExecuteResult *result;
2748
2749     while (1)
2750     {
2751         /* Execute the DPM module: */
2752         retVal = DPM_execute (gMmwMCB.dataPathObj.objDetDpmHandle, &resultBuffer);
2753
2754         MmwDemo_transmitProcessedOutput(gMmwMCB.loggingUartHandle, result,
2755                                         &currSubFrameStats->outputStats);
2756
2757         /* Wait until s/w session is complete. We expect the LVDS transmission
2758          * s/w session to be completed by now because the UART transmission
2759          * Doing the wait immediately after starting the transmission above
2760          * will serialize the LVDS and UART transfers so it is better to do
2761          * transmission (which is blocking call i.e UART transmission is complete)
2762          */
2763     }
2764 }
```

# Datapath Execution – Runtime View

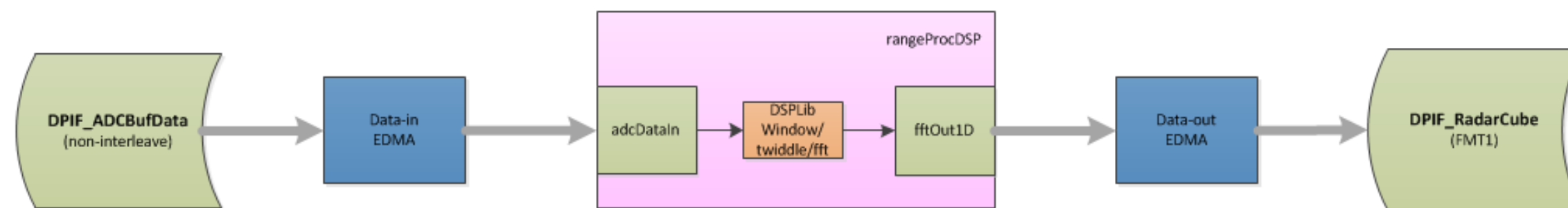
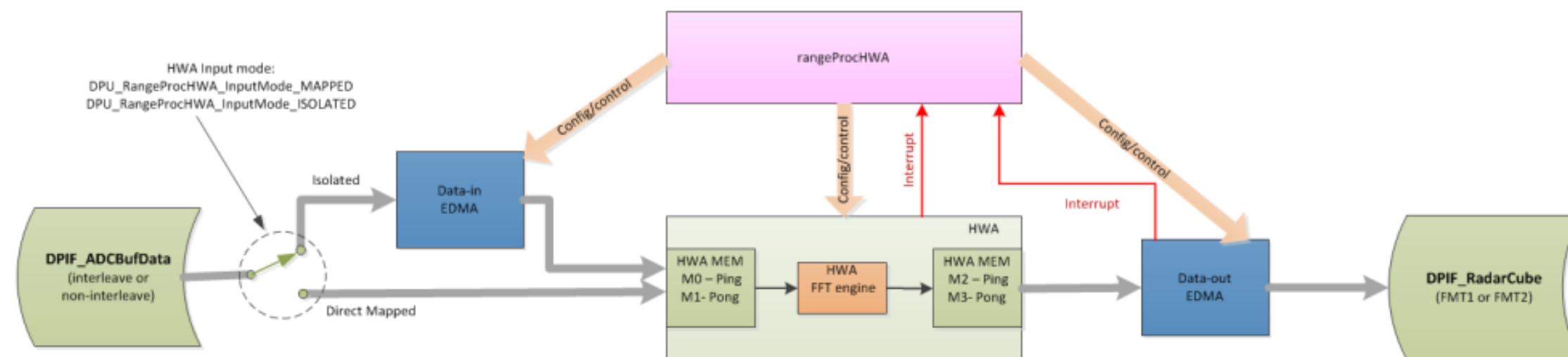
- **At sensor start:** Application calls DPM\_start which calls the DPC's pre-registered start function DPC\_ObjectDetection\_start.
  - The mmw demo does this in the sensorStart CLI command handler function.
- **At every frame interrupt:** The DPC's pre-registered frame start call back function, DPC\_ObjectDetection\_frameStart is invoked.
  - **Recall from DPC\_init:** The DPC's frameStart callback is registered with the DPM during DPC initialization
- This invokes the DPC's registered execute function, e.g. DPC\_ObjectDetection\_execute
  - **Under the hood:** The Frame Start handler calls DPM\_notifyExecute which posts the semaphore for DPM\_execute
  - DPM\_execute then calls the DPC's pre-registered execute function.
- DPC\_ObjectDetection\_execute performs the frame processing and populates the output in DPC\_ObjectDetection\_ExecuteResult structure and returns.
- Application reads the output structure and sends it over UART. This completes the frame processing.

```
main.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\demo\xwr64xx\mmw) - GVIM
File Edit Tools Syntax Buffers Window Help
2174 */
2175 void MmwDemo_transmitProcessedOutput(UART_Handle uartHandle,
2176                                     DPC_ObjectDetection_ExecuteResult *result,
2177                                     MmwDemo_output_message_stats *timingInfo)
2178 {
2179     MmwDemo_output_message_header header;
2180     MmwDemo_GuiMonSel *pGuiMonSel;
2181     MmwDemo_SubFrameCfg *subFrameCfg;
2182     uint32_t tlvIdx = 0;
2183     uint32_t i;
2184     uint32_t numPaddingBytes;
2185     uint32_t packetLen;
2186     uint8_t padding[MMWDEMO_OUTPUT_MSG_SEGMENT_LEN];
2187     uint16_t *detMatrix = (uint16_t *)result->detMatrix.data;
2188
2189     MmwDemo_output_message_tlv tlv[MMWDEMO_OUTPUT_MSG_MAX];
2190
2191     tlvIdx = 0;
2192     /* Send detected Objects */
2193     if ((pGuiMonSel->detectedObjects == 1) || (pGuiMonSel->detectedObjects == 2) &&
2194         (result->numObjOut > 0))
2195     {
2196         UART_writePolling (uartHandle,
2197                           (uint8_t*)&tlv[tlvIdx],
2198                           sizeof(MmwDemo_output_message_tlv));
2199
2200         /*Send array of objects */
2201         UART_writePolling (uartHandle, (uint8_t*)result->objOut,
2202                           sizeof(DPIF_PointCloudCartesian) * result->numObjOut);
2203
2204         tlvIdx++;
2205     }
2206
2207     /* Send detected Objects Side Info */
```

# DPUUs and DPCs in SDK 3.x

# RangeProc DPU

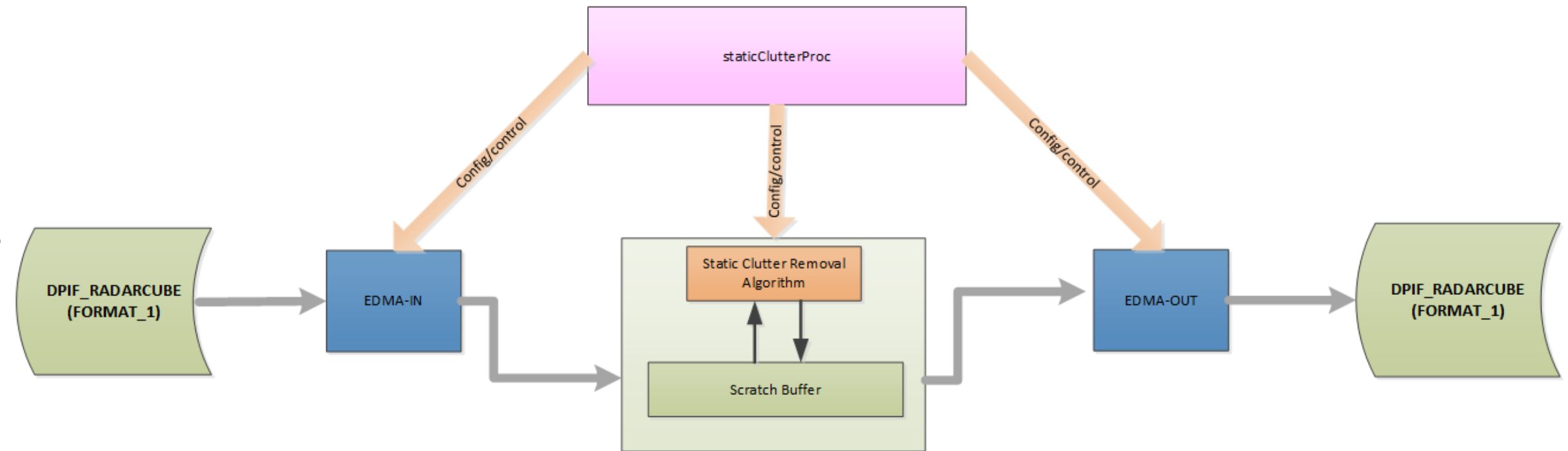
- Purpose: (1D FFT+ DC Range Calib) processing during active frame.
  - Takes ADCBuf as input
    - interleaved or nonInterleaved format
    - Single chirp or multichirp (DSP mode only)
    - Either direct access (HWA mode only) or via EDMA
  - Produces RadarCube in L3 in user requested format (fixed set of formats described in data interface).
  - Performs FFT using HWA or DSP based on configuration.
  - Performs DC range calibration either inline (DSP mode) or at the end of all chirps (HWA mode).
- Supported architecture
  - R4F, C674x
  - Different files for HWA based and S/W based implementation





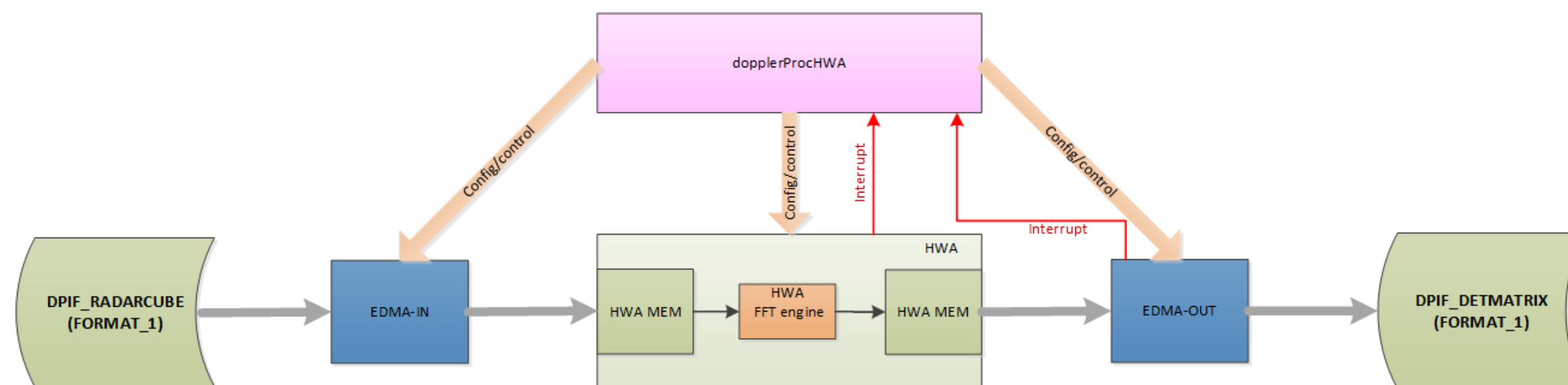
# Static Clutter DPU

- Purpose: (Clutter Removal) processing during inter frame.
  - Takes non-transposed formatted 1DOUT Radarcube as input
  - Updates the RadarCube in L3 (keeping the same format)
  - S/W based only
- Supported architecture
  - R4F, C674x



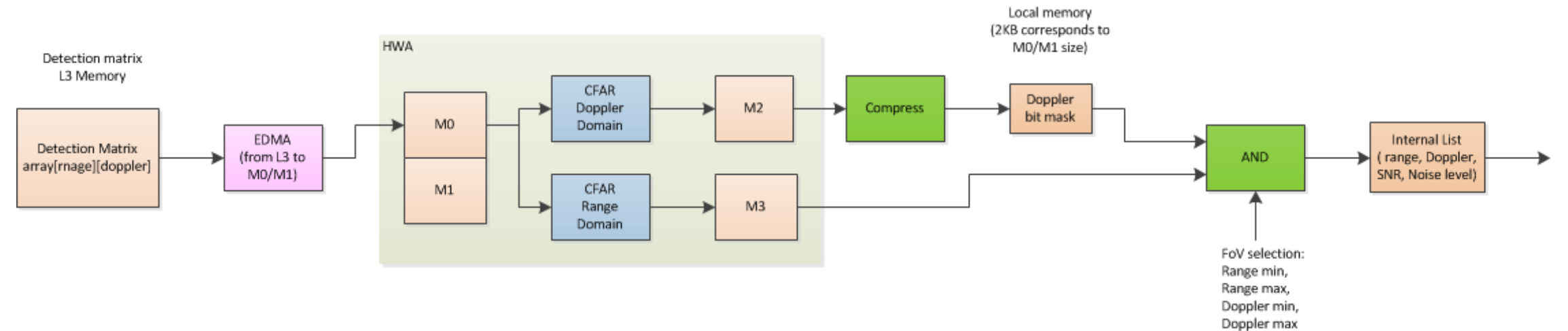
# Doppler DPU

- Purpose: (2D FFT + Energy Sum) processing during inter frame.
  - Takes non-transposed formatted 1DOUT Radarcube as input
  - Produces Detection Matrix in L3 in fixed format
  - Performs FFT and Energy Sum using H/W(HWA)
    - S/W(DSP) based implementation would be added in the SDK in future.
- Supported architecture
  - R4F, C674x



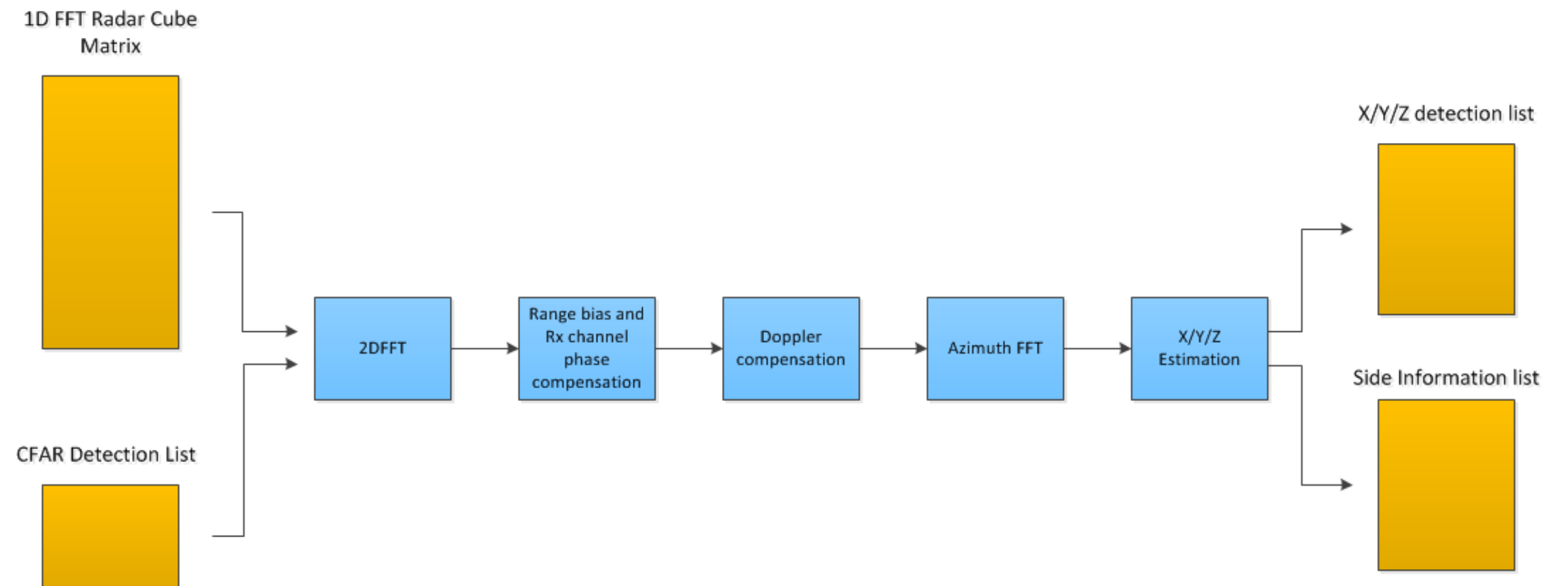
# CFAR-CA DPU

- Purpose: (CFAR-CA +peak grouping) processing during inter frame.
  - User can choose between various CFAR-CA implementation: CFAR-CA, CFAR-CASO, CFAR-CAGO
  - Fixed Point implementation
  - 2 pass implementation: first pass (optional) in Doppler direction and then second pass in Range direction
  - Performs CFAR and PeakGrouping using H/W(HWA)
    - S/W(DSP) based implementation would be added in SDK in future
  - Takes Detection Matrix as input
  - Produces bitmask of peaks and SNR information for AoA.
- Supported architecture
  - R4F, C674x



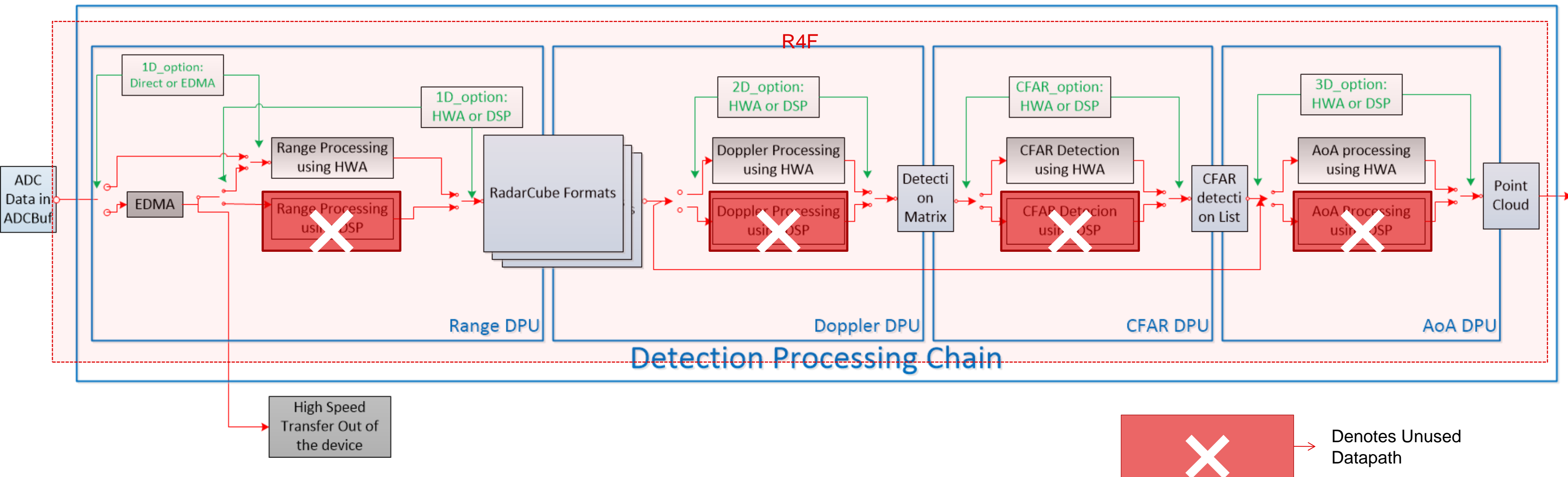
# AoA DPU

- Purpose: (Range/Phase/Doppler compensation + *Near field correction* + *Max Velocity enhancement* + AoA + FoV filter) processing during inter frame.
  - Takes non-transposed formatted 1DOUT Radarcube, detection matrix and peak bitmask as input
  - Produces Point Cloud
  - Performs any FFT ops using H/W(HWA)
    - S/W(DSP) based implementation will be added in SDK in future
  - All other processing operations are done using S/W.
    - For R4F based implementation, near field and max velocity algos will not be offered
  - All the processing ops other than AoA FFT is optional and driven by user input
- Supported architecture
  - R4F, C674x



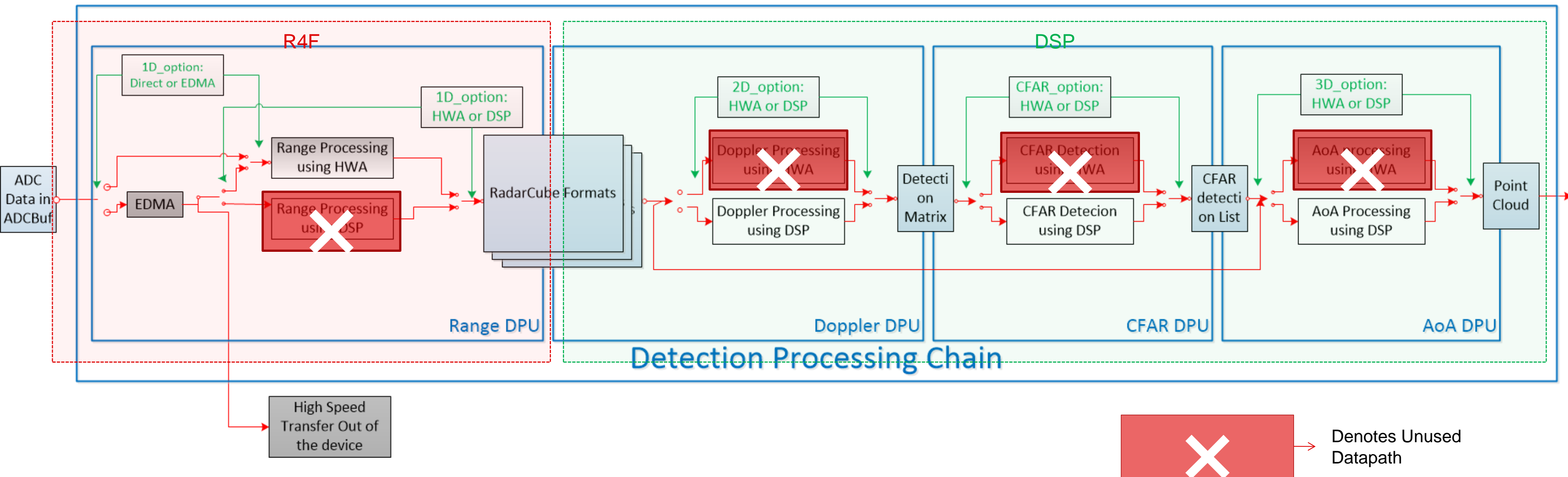
# xWR64xx OOB Demo Chain (DPC)

- Detection Processing Chain (DPC) for **xWR64xx** Out of Box demo.
- Demonstrates Radar Signal Processing using Cortex R4F + HWA
- Also runs on **xWR68xx** (C674x DSP not used)

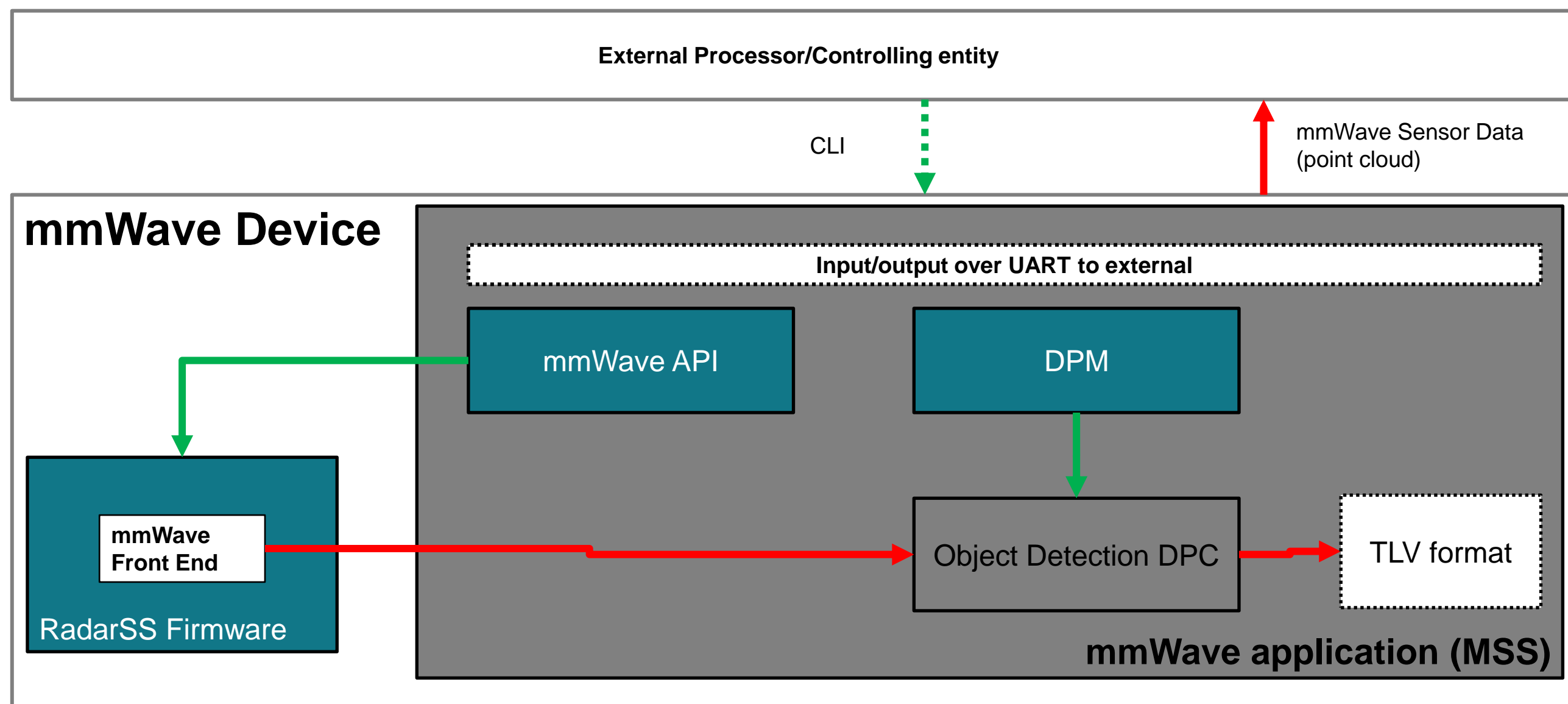


# xWR68xx OOB Demo Chain (DPC)

- Detection Processing Chain (DPC) for xWR68xx Out of Box demo.
- Demonstrates Radar Signal Processing using Cortex R4F + HWA + C674x DSP
- DSP provides higher performance and frees up R4F for other processing (e.g. Object Tracking)



# 68xx OOB Demo - Application View



# Software Development and Debugging



# Software Development Resources

- **Software Development Resources:** The following resources are key to learning about software development for TI processors.
  - **BIOS users guide:** Installed as part of [MMWAVE-SDK](#), it is available in the TI install directory e.g. C:\ti\bios\_6\_73\_01\_01\docs\Bios\_User\_Guide.pdf.
    - This is the one of the best resources for learning about SYSBIOS and software development for TI processors.
  - **Linker command files:** Understanding these is fundamental to developing applications for TI RTOS ([SYSBIOS](#)). Refer to the following resources:
    - [TI Linker Command File Primer](#)
    - [Advanced Linker Techniques for Convenient and Efficient Memory Usage](#)
- **Other helpful resources**
  - [Getting Started with Code Composer Studio v7](#)
  - [Debugging Common Application Issues with TI-RTOS](#)
  - [TI-RTOS Workshop](#)
  - [Introduction to C6000 Architecture](#)
  - [C6000 Cache - Overview \(7 of 15\)](#)
  - [Using C6000 EDMA3 - Part 1 \(13 of 15\)](#)

# MMWAVE-SDK Debugging

- **Pre-requisites:** The user should be familiar with general CCS debugging techniques e.g. running code in CCS, putting breakpoints, observing the values of variables and memories etc. Following resources can be used to ramp-up on these topics if needed.
  - [Getting Started with Code Composer Studio v7](#)
  - [Debugging Common Application Issues with TI-RTOS](#)
- In addition to the above, the user must be familiar with basic procedures for running an MMWAVE application in CCS debug mode. This is explained in the [mmWave Industrial Toolbox](#) at the following page:
  - [Using CCS Debug for Development](#)

Using CCS Debug for Development

## Using CCS Debug for Development

This mode should be used when debugging with CCS is involved and/or developing an mmWave application where the .bin files keep changing constantly different image to the device's RAM on every boot.

### 0. Requirements

- PC with:
  - Recommended OS: Windows 7 or 10
  - Code Composer Studio (version as specified in demo User's Guide)
  - mmWave SDK installed (version as specified in demo User's Guide)
- The EVM should already be setup for **Flashing Mode** according to the appropriate hardware setup guides for your EVM:

EVM	Guide
IWR6843ISK, IWR6843ODS, or IWR6843AOPEVM + MMWAVEICBOOST Carrier Board	<a href="#">Hardware Setup for Flashing in MMWAVEICBOOST Mode</a>
IWR1443BOOST, IWR1642BOOST, or IWR1843BOOST	<a href="#">Hardware Setup of IWRXXXXBOOST for Flashing Mode</a>

Note: The IWR6843AOPEVM cannot be used by itself for CCS debug mode. It must be used as an antenna module in conjunction with the MMWAVE

### 1. Flash the CSS debug firmware using Uniflash

The debug binary is provided in the mmWave SDK. Flash the appropriate binary according to the instructions for [using UniFlash](#)

# MMWAVE-SDK Debugging

- **Rebuilding SDK code for debug:**
  - All MMWAVE-SDK code (drivers, OOB demo application and DPU libs) is built with `-O3` optimization, which does not allow single step debugging in the corresponding components.
  - In order to enable step debugging in CCS, the `-O3` optimization options needs to be removed (or changed to a lower level)
    - Remove the `-O3` flag in the SDK common makefile from `R4F_CFLAGS` and/or `C674_CFLAGS` directives to change build options for ARM and/or DSP code.
    - Re-build the demo and the desired pre-compiled libs (e.g. drivers, DPUs) separately with the non-optimized configuration to enable debugging.
- **A note about DPUs:** All DPUs are linked with the application as pre-compiled libs, so just re-building the application/demo code does not re-build the DPU libs automatically. Any DPUs that need to be debugged, should be re-built first with the new build options and then the applications code re-compiled to link with the new DPU lib(s). This applies to the SDK drivers as well.
- Removing optimization may increase the memory requirements and/or break real-time behavior.

## SDK Compile flags for Cortex R4F core

```
mmwave_sdk.mak (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\common) - GVIM
File Edit Tools Syntax Buffers Window Help
111 # Compiler flags used for the R4 Builds:
112 R4F_CFLAGS = -mv7R4 --code_state=16 --float_support=VFPv3D16 --abi=eabi -me
113             --define=SUBSYS_MSS --define=$(PLATFORM_DEFINE)
114             --define=_LITTLE_ENDIAN --define=DebugP_ASSERT_ENABLED $(R4F_INCLUDE)
115             -g -O3 -display_error_number --diag_warning=225 --diag_wrap=off
116             --little_endian --preproc_with_compile --gen_func_subsections
117             --emit_warnings_as_errors $(R4F_CFLAGS_ENUM_TYPE)
118
```

## SDK Compile flags for C674x DSP core

```
mmwave_sdk.mak (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\common) - GVIM
File Edit Tools Syntax Buffers Window Help
207 C674_INCLUDE = -i$(MMWAVE_SDK_INSTALL_PATH) -i$(C674_CODEGEN_INSTALL_PATH)/include $(
208
209 # Compiler Flags for C674 Builds:
210 C674_CFLAGS = -mv6740 --abi=eabi --gcc -g -O3 -mf3 -mo --define=SUBSYS_DSS
211             --define=$(PLATFORM_DEFINE) --define=_LITTLE_ENDIAN --display_error_nu
212             --define=DebugP_ASSERT_ENABLED --diag_warning=225 --diag_wrap=off
213             --preproc_with_compile $(C674_INCLUDE) --emit_warnings_as_errors
214
```

# MMWAVE-SDK Debugging

- **Disabling Real time asserts:**
  - If the DPC cannot complete the frame (or sub-frame) processing within the real-time deadline, it raises as assert causing the application to exit
  - When step debugging a distributed DPC which runs on both R4F and DSP cores (e.g. in the 68xx point cloud detection chain), this assert should be disabled on both cores in DPC code as shown otherwise application would crash as shown in this CCS console log.
  - NOTE: Make sure to re-enable the frame timing assert when running the DPC in non debug mode.

```
objectdetection.c (C:\ti\mmwave_sdk_03_04_00_03\packages\ti\datapath\dpc\objectdetection\objdethwa\src) - GVIM
File Edit Tools Syntax Buffers Window Help
418 static void DPC_ObjectDetection_frameStart (DPM_DPCHandle handle)
419 {
420     ObjDetObj      *objDetObj = (ObjDetObj *) handle;
421
422     objDetObj->stats.frameStartTimeStamp = Cycleprofiler_getTimeStamp();
423
424     DebugP_log2("ObjDet DPC: Frame Start, frameIndx = %d, subFrameIndx = %d\n",
425               objDetObj->stats.frameStartIntCounter, objDetObj->subFrameIndx);
426
427     /* Check if previous frame (sub-frame) processing has completed */
428     DPC_Objdet_Assert(objDetObj->dpmHandle, (objDetObj->interSubFrameProcToken == 0)
429     objDetObj->interSubFrameProcToken++;
430
```

```
Console
iwr6843.ccxml:CIO
[Cortex_R4_0] *****
Debug: Launching the MMW Demo on MSS
*****
Debug: Launched the Initialization Task
Debug: mmWave Control Initialization was successful
Debug: mmWave Control Synchronization was successful
[C674X_0] Debug: DPM Module Sync is done
[Cortex_R4_0] Debug: CLI is operational
Debug: Sending r1RfSetLdoBypassConfig with 0 0 0
===== Heap Memory Stats =====
              Size      Used      Free      DPCUsed
System Heap(TCMB)  32768    25840    6928      2048
              L3      786432    131072    655360
localRam(TCMB)    4096      512      3584
===== Heap Memory Stats =====
              Size      Used      Free      DPCUsed
System Heap(L2)   32768    16104    16664      0
              L3      786432    8192    778240
localRam(L2)     50176    15016    35160
localRam(L1)     16384     5632    10752
Starting Sensor (issuing MMWave_start)
{module#9}: "../objdetrangehwa.c", line 640: error {id:0x10000, args:[0x1a19c, 0x1a19c]}
xdc.runtime.Error.raise: terminating execution
```



# Understanding Error Codes: Datapath Errors

- When demo runs into error conditions, an error code is generated and printed out on CCS console
- The error code is a negative integer e.g. shown in the picture.
- Can be from various sources such as Drivers, Control modules, DPC, DPU or demo (i.e. application)
- Error code defined as: (**Module error code base** minus **Module specific error code**)
  - The module error code base values are defined in **packages\ti\common\mmwave\_error.h**.
  - The base error codes for DPC and DPU are define in **packages\ti\datapath\dpif\dp\_error.h**
  - Individual DPU specific error codes defined in the DPU header files
- Example:** Parsing the error **-30430** shown here
  - The error code is from module with error base "-30000", which indicates it is DPU error
  - Referring to **dp\_error.h**, base "-30400" is from AOA Proc.
  - Then find the error code in **aoaprocdsp.h** for error(-30) which is DPU\_AOAPROCDSP\_ESCRATCHSIZE
- NOTE:** In SDK demos, these error codes are not sent out on UART so the demo must be run in CCS debug mode to get the error code
- Refer to SDK module documentation at the following location (in the SDK install directory) for more details:
  - [file:///C:/ti/mmwave\\_sdk\\_03\\_05\\_00\\_04/packages/ti/demo/xwr68xx/mmwave/docs/doxygen/html/index.html#mmwave\\_error](file:///C:/ti/mmwave_sdk_03_05_00_04/packages/ti/demo/xwr68xx/mmwave/docs/doxygen/html/index.html#mmwave_error)

## Datapath Errors

```
Debug: mmWave Control Initialization was successful
Debug: mmWave Control Synchronization was successful
Debug: Sending rIRfSetLdoBypassConfig with 3 1 0
Debug: Init Calibration Status = 0xffe
Azimuth Tx: 1 (MIMO:0), Elev Tx:0
Ant setting virtualAzim: 4 , virtual Elev :0
chirpThreshold 1 max = 64,
Error: DPM Report 4 received with error:-30430 arg0:0x64 arg1:0x80050e8
{module#9}: "src/mss/mss_main.c", line 1391: error {id:0x10000, args:[0
xdc.runtime.Error.raise: terminating execution
```

**-30430 = -30000 -400 -30**

```
mmwave_error.h (C:\ti\mmwave_sdk_03_05_00_04\packages\ti\common) - GVIM
File Edit Tools Syntax Buffers Window Help
76 * Base Error Code for the mmWave data path (-30000 - -59999)
77 *****
78 #define MMWAVE_ERRNO_DPU_BASE (-30000)
79 #define MMWAVE_ERRNO_DPC_BASE (-40000)
80 #define MMWAVE_ERRNO_DEMO_BASE (-50000)
81
82 #ifdef __cplusplus
83 }
84 #endif
```

**-30000 = DPU Error base**

```
dp_error.h (C:\ti\mmwave_sdk_03_05_00_04\packages\ti\datapath\dpif) - GVIM1
File Edit Tools Syntax Buffers Window Help
51 #define DP_ERRNO_RANGE_PROC_BASE (MMWAVE_ERRNO_DPU_BASE -100)
52 #define DP_ERRNO_DOPPLER_PROC_BASE (MMWAVE_ERRNO_DPU_BASE -200)
53 #define DP_ERRNO_CFARCA_PROC_BASE (MMWAVE_ERRNO_DPU_BASE -300)
54 #define DP_ERRNO_AOA_PROC_BASE (MMWAVE_ERRNO_DPU_BASE -400)
55 #define DP_ERRNO_STATIC_CLUTTER_PROC_BASE (MMWAVE_ERRNO_DPU_BASE -500)
56 #define DP_ERRNO_DEMO_BASE (MMWAVE_ERRNO_DPU_BASE -600)
57
```

**-400 = AOA Proc Error base**

```
aoaprocdsp.h (C:\ti\mmwave_sdk_03_05_00_04\packages\ti\datapath\dpc\dp\aoaprocc) - GVIM2
File Edit Tools Syntax Buffers Window Help
126 /**
127 * @brief Error Code: One of the provided scratch buffers has insufficient size
128 */
129 #define DPU_AOAPROCDSP_ESCRATCHSIZE (DP_ERRNO_AOA_PROC_BASE-30)
130
131 /**
```

**-30 = Insufficient Memory Error**

# Understanding Error Codes: mmWave Errors

- Besides Datapath errors, there is another class of errors known as mmWave module errors.
- These represent errors returned by the RF Front-End e.g. incorrect profile configuration.
- Defined as a combination of mmWave error, Subsystem error and error level as shown.
  - mmwave errors defined in `packages\ti\control\mmwave\mmwave.h`
  - Subsystem errors defined in `packages\ti\control\mmwavelink` for `mmwavelink.h`
  - Error level represents WARNING or ERROR.
- Example:** The error shown in the log here indicates
  - The error is from module(-3100 i.e. mmwave) with error -8 (MMWAVE\_EPROFILECFG)
  - The Subsystem (mmwavelink) error is 36 which is defined as `RL_RET_CODE_PF_START_FREQ_INVALID_IN` in `mmwavelink.h`, which indicates invalid start frequency specified in ProfileCfg API.

## mmWave Link errors

```
Console
iwr6843.cxml:CIO
[Cortex_R4_0] *****
Debug: Launching the MMW Demo on MSS
*****
Debug: Launched the Initialization Task
Debug: mmWave Control Initialization was successful
Debug: mmWave Control Synchronization was successful
[C674X_0] Debug: DPM Module Sync is done
[Cortex_R4_0] Debug: CLI is operational
Debug: Sending rIRfSetLdoBypassConfig with 0 0 0
Error: mmWave Config failed [Error code: -3108 Subsystem: 36]
```

Bits(31::16)	Bits(15::2)	Bits (1::0)
mmwave error	Subsystem error	error level

```
mmwave_err.h (C:\ti\mmwave_sdk_03_05_00_04\packages\ti\common) - GVIM3
File Edit Tools Syntax Buffers Window Help
60 #define MMWAVE_ERRNO_EDMA_BASE (-3000)
61 #define MMWAVE_ERRNO_BASE (-3100)
62 #define MMWAVE_ERRNO_CSI_BASE (-3200)
```

```
mmwave.h (C:\ti\mmwave_sdk_03_05_00_04\packages\ti\control\mmwave) - GVIM3
File Edit Tools Syntax Buffers Window Help
148 */
149 #define MMWAVE_EPROFILECFG (MMWAVE_ERRNO_BASE-8)
150
151 /**
152 * @brief Error Code: mmWave link chirp configuration failed
153 */
154 #define MMWAVE_ECHIRPCFG (MMWAVE_ERRNO_BASE-9)
```

```
mmwavelink.h (C:\ti\mmwave_sdk_03_05_00_04\packages\ti\control\mmwavelink) - GVIM3
File Edit Tools Syntax Buffers Window Help
441 /*! Profile config API */
442 #define RL_RET_CODE_PF_IND_INVALID_IN (35U) /* PF indx >= 4 */
443 #define RL_RET_CODE_PF_START_FREQ_INVALID_IN (36U) /* PF freq const is not
444 with[76GHz,81GHz] in limit */
445 #define RL_RET_CODE_PF_IDLE_TIME_INVALID_IN (37U) /* PF idle time const > 5.24ms */
446 #define RL_RET_CODE_PF_IDLE_TIME_1INVALID_IN (38U) /* Maximum DFE spill time (refer
```



# Extending SDK architecture for advanced applications

# Developing a Custom Radar Processing Chain

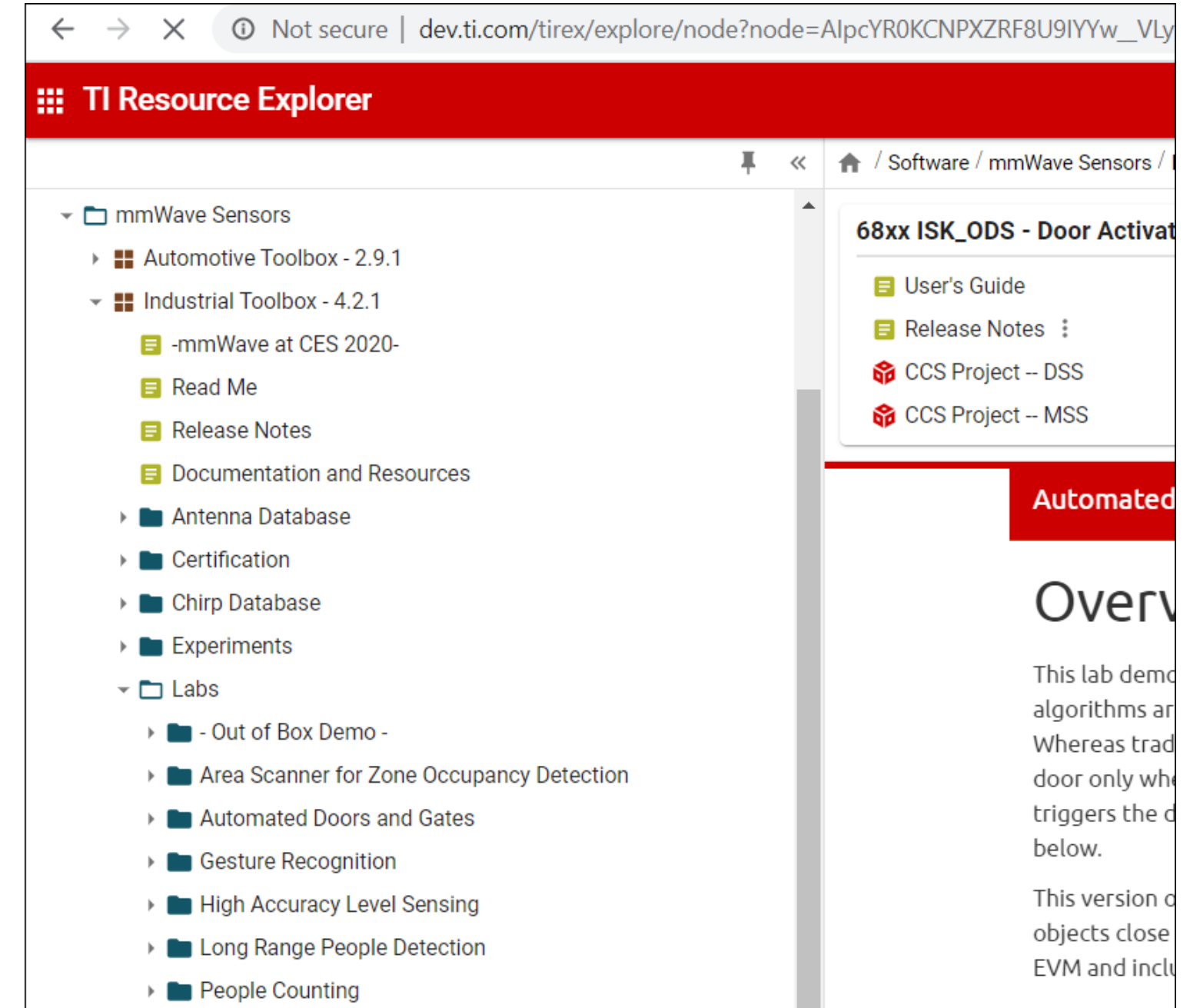
## Key considerations for developing a custom mmWave processing chain with SDK 3.x architecture

- Understand the data processing chain for the target application and model it in terms of DPC and DPUs
- Re-use the SDK Out of Box Detection Processing **OR** develop a new Detection Chain
  - SDK Out of box DPC is a Range-Doppler based processing chain.
  - Lower angular resolution as compared to Range-Azimuth (Capon Beamforming) chain
    - SDK Detection Chain re-used for Long Range People Detection and Area Scanner Demos
    - But a Higher resolution Capon Beamforming Detection Chain was developed for Indoor People Counting Demos
  - Range Processing is still re-used irrespective of the rest of the Detection Processing
- Additional processing requirements beyond the OOB point cloud detection
  - Can the additional processing be added to an existing DPU or do we need to create a new DPU
    - Object Tracking - new DPU
    - Static Object Detection - new DPU
    - 2D AoA (Angle of Arrival) using DSP – Enhancement to existing AoA DPU
- Enhancements needed at the DPC and Application level e.g.
  - Tracking
  - Beam-steering
  - Static object detection
- Memory and MIPS requirements for the additional processing

# Examples of Custom DPUs and DPCs

## New DPUs and DPCs developed in mmWave Industrial Toolbox

- TrackerProc DPU and ObjectDetectionAndTracking DPC
  - 2D/3D Object tracking using SDK 3.3 Out of Box detection chain
  - Used in the following demos
    - Long Range People detection
    - Traffic Monitoring
    - Area Scanner
    - Automated Doors and Gates
- TrackerProcCapon DPU and Capon3D DPC
  - 2D/3D Object Tracking using Capon beamforming detection chain
  - Used in the following demos
    - 3D People Counting Demo – Side Mount
    - 3D People Counting Demo – Overhead Mount
    - Sense and Direct HVAC Control Demo
- StaticDetProc DPU and StaticObjDet DPC
  - 2D/3D Object tracking using SDK 3.3 Out of Box detection chain with added Static detection capability
  - Used in the following demos
    - Area Scanner
    - Automated Doors and Gates
- Available in Industrial Toolbox 4.x download under **C:\ti\mmwave\_industrial\_toolbox\_4\_x\_x\labs\common**



# Area Scanner and Automated Doors

- Custom processing chain developed for [Area Scanner](#) and [Automated Doors](#) Demos.
- Based on xWR68xx OOB Range-Doppler Detection Chain with addition of 2D AoA using DSP
- Adds Object Tracking and Static Detection Capabilities

68xx ISK\_ODS - Area Scanner

68xx Area Scanner Lab User's Guide

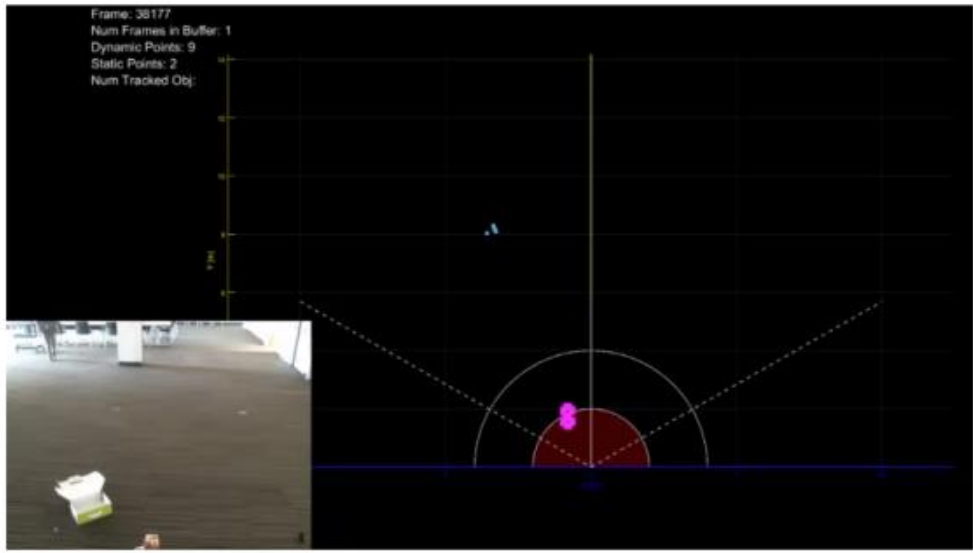
## Overview of 68xx Area Scanner

This lab demonstrates the use of TI mmWave sensors for Area Scanner applications. The range, velocity, and angle data from mmWave sensors can enable the detection of objects or people as they enter or exit zones of interest.

Key features in this lab are the static detection and group tracking algorithms:

- Typically it can be difficult to distinguish static objects of interest that have been left in an area from the static floor or fixtures that are also in the area. Using the static detection algorithm enables the detection of static objects such as boxes, carts, or other equipment that have been left behind while ignoring the permanent static objects that comprise the scene.
- With group tracking, the sensor is able to gauge a moving object's trajectory and speed, enabling the area scanner system to predict before a safety zone is breached and to dynamically adjust the system's behavior depending on the object's speed of approach.

The lab runs on the TI mmWave sensor IWR6843ISK and IWR6843ISK-ODS EVM and includes a MATLAB visualizer.



Frame: 30177  
Num Frames in Buffer: 1  
Dynamic Points: 9  
Static Points: 2  
Num Tracked Obj: 1


68xx ISK\_ODS - Door Activation Sensor

Automated Doors User's Guide

## Overview

This lab demonstrates the use of TI mmWave sensors in an automated door application. Tracking algorithms are used to sense range, speed, and direction of motion to trigger the opening of a door. Whereas traditional door sensors operate based only on proximity, this lab uses the sensor to open the door only when a person is approaching the door. Velocity information obtained by the sensor also triggers the door to open earlier if a person is moving towards the door faster as is shown in the GIF below.

This version of the lab incorporates a new static object detection algorithm to detect non-moving objects close to the sensor. The lab runs on the TI mmWave sensor IWR6843ISK and IWR6843ISK-ODS EVM and includes a MATLAB visualizer.



Overview  
QuickStart  
Developer's Guide  
Need More Help?

# Demo: Long Range Outdoor People Detection and Tracking

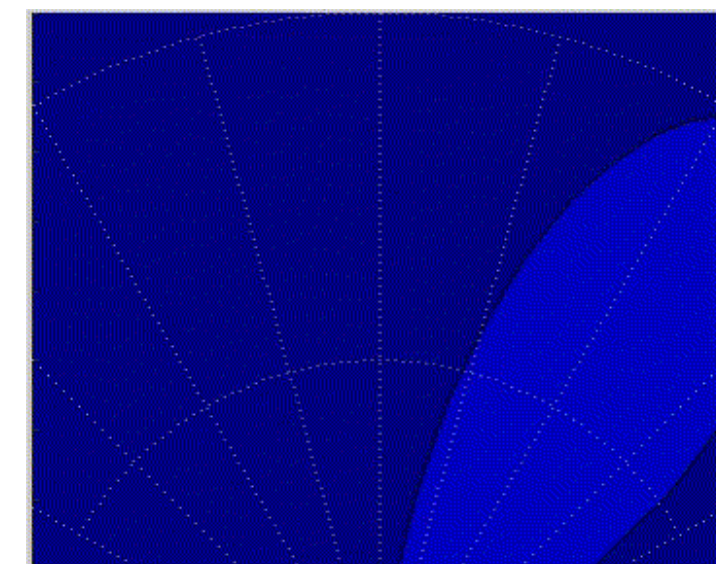
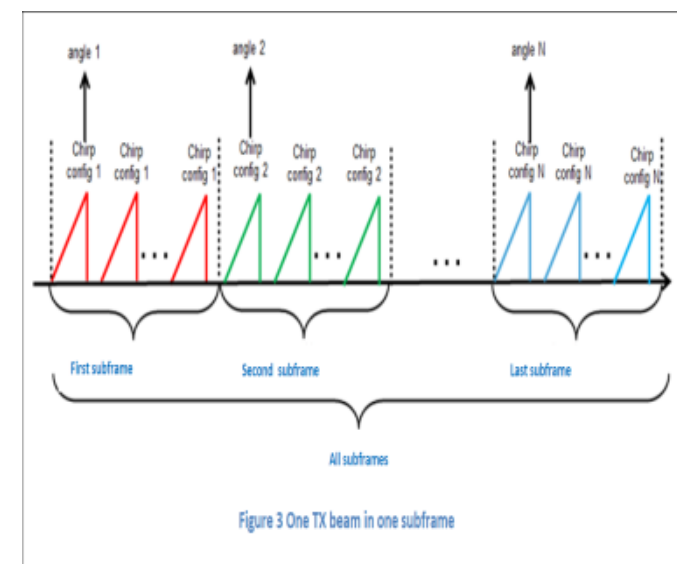
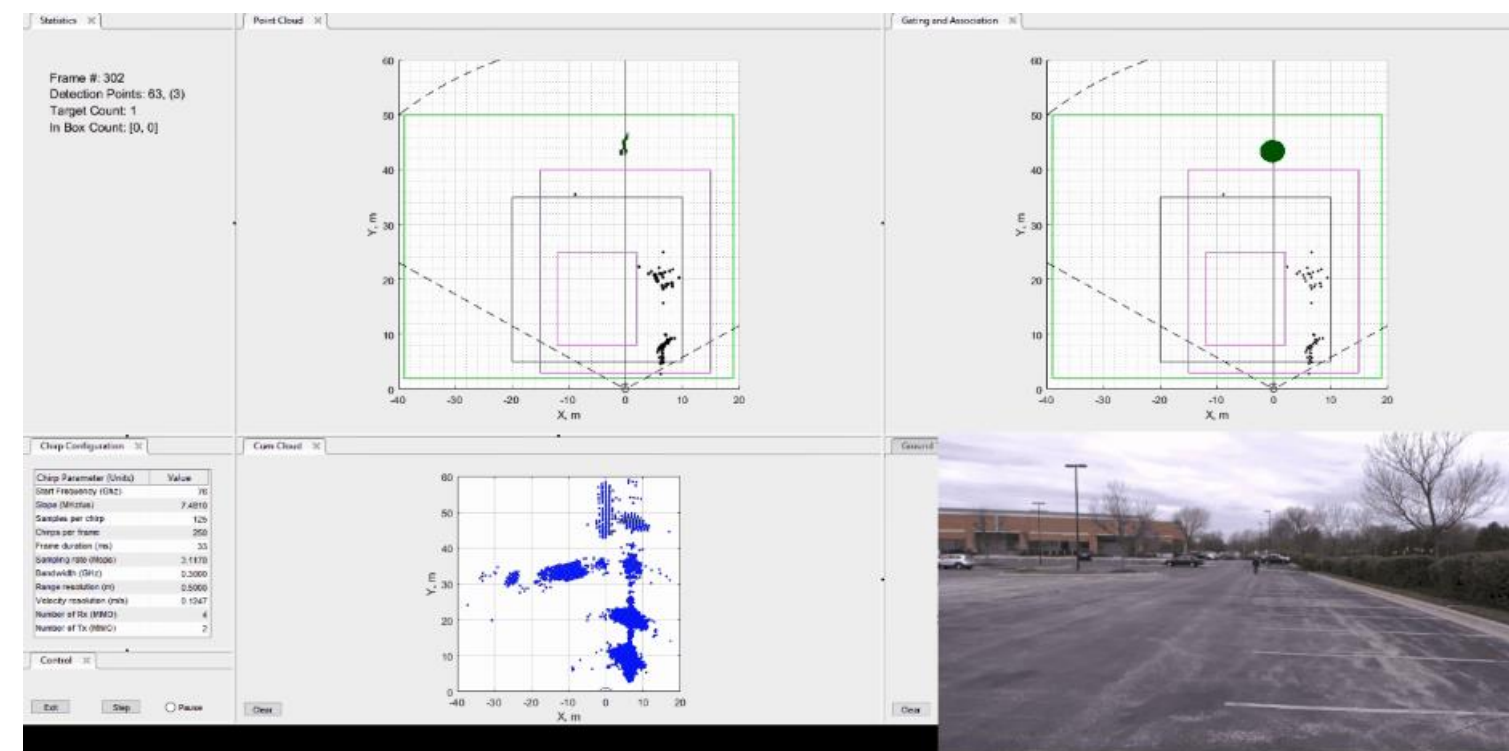


# 100m People Detection and Tracking

## • Features

- SDK 3.3 Range Doppler Detection Chain with Object tracking on R4F
- Tracker DPU developed on top of SDK 3.x
- Runs on IWR6843ISK ES2.0 and IWR1843BOOST
- Supports 2D and 3D tracking
- People tracking tested upto 100m with IWR6843ISK ES2.0 (3TX SIMO)
- Re-used for Traffic Monitoring Demo on IWR1843/6843
- Supports ISK, ODS and AOP antennas (for Indoor applications such as Area Scanner with Static Detection)
  - DSP based AoA DPU modified to add 2D DoA for ODS/AOP
- Built on SDK 3.x OOB Demo
  - Advanced features included for free e.g. Run time CFAR tuning, FoV filtering, ADC data streaming over LVDS.

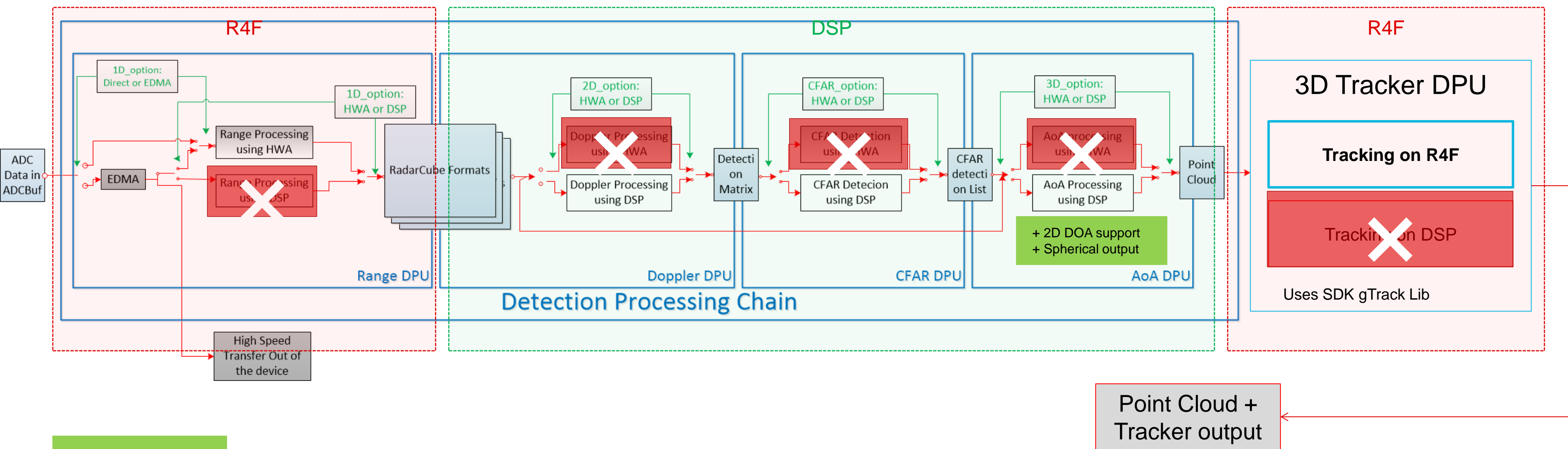
## Overview



Sub frame based TX Beam Steering



# 3D Long Range Tracker Chain



Denotes new or modified components

Denotes Unused Datapath



**©2020 Texas Instruments Incorporated. All rights reserved.**

The material is provided strictly "as-is" for informational purposes only and without any warranty.  
Use of this material is subject to TI's **Terms of Use**, viewable at [TI.com](https://www.ti.com)

## IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES "AS IS" AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI's products are provided subject to TI's Terms of Sale ([www.ti.com/legal/termsofsale.html](http://www.ti.com/legal/termsofsale.html)) or other applicable terms available either on [ti.com](http://ti.com) or provided in conjunction with such TI products. TI's provision of these resources does not expand or otherwise alter TI's applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265  
Copyright © 2020, Texas Instruments Incorporated