Hardware in loop for the data play back from the DMM interface

ABSTRACT

This application note guides how to use the DMM interface in the XWR1642 and XWR1843, XWR684 devices to playback the raw ADC data that might have been captured over LVDS interface. During playback the data from the RF front end is ignored and instead taken from the DMM interface. This data could then be sent to the DSP for further processing and object detection.

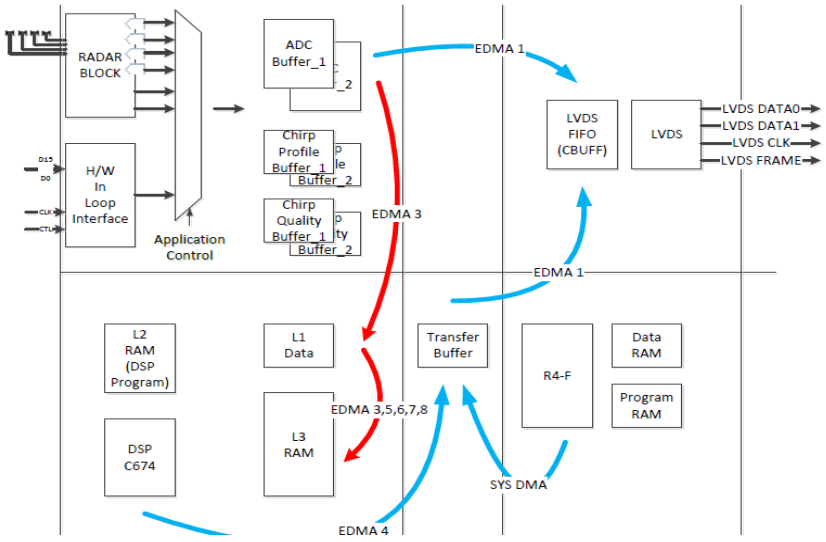
Table of Contents

1. Introduction

The raw ADC data captured over LVDS interface can be played back by using the The Data Modification Module (DMM) interface to emulate the field test scenario in the development environment. In this mode the radar front is bypassed and the ADC buffer is filled with the same data from the DMM interface instead. The Data Modification Module (DMM) receives data over a dedicated port from external source and writes the data directly to the address given in the packet and programmed in the module. This address could be internal buffers like ADC buffer, or could be registers. The timing emulated over the DMM interface needs to match the timing corresponding to the actual chirp configuration in order to emulate the actual scenario accurately. This includes the generation of chirp done interrupt at the desired intervals.

2. Data flow block diagram

In the normal operation the ADC data got from the Radar front end is transferred to the L1 and L3 memories, which is picked by the DSP for processing. Incase of the playback mode first the user application transfers ADC data over LVDS interface, which is captured for later use. This LVDS data is then transformed to DMM interface format (parallel CMOS interface) for playback. The user selects the mux control to fill the ADC buffer from the DMM interface instead of the ADC, after that the same data flow to enable DSP processing (as the original application) is followed.



3. DMM interface signals and modes

DMM uses port composed of the below signals:

Signal name	Description
DMMSYNC (or CTL)	This signal has to be provided by external hardware. It signals the start of a new packet. It has to be active (high) for one full

	DMMCLK cycle, starting with the rising edge of DMMCLK. If the DMMSYNC pulse is longer than a single DMMCLK cycle and two falling edges of DMMCLK see a high pulse on DMMSYNC, the module will treat the second DMMSYNC pulse as the start of a packet and will flag a PACKET_ERR_INT
DMMCLK	The clock is externally generated and can be suspended between two packets. For this feature, CONTCLK must be set to 0. If the clock is not stopped between two packets, CONTCLK must be set to 1. Data will be latched on the falling edge of DMMCLK signal.
DMMDATA[15:0]	These pins receive the packet information transmitted by the external hardware. Data is latched on the falling edge of DMMCLK Note: Depending upon device family number of available pins could vary
DMM_MUX_CTL (or SEL)	This pin is used to select between 2 DMMs. 0 → DMM1 ; 1 → DMM2 Same SYNC, CLK, DATA are used between 2 DMMs available.

DMM interface can be operated in two modes, Trace mode and Direct Data mode. Each of the DMM IP's can be programmed in either of the modes. The selection of the mode is done in the "GLBCTRL" register of the corresponding DMM IP:

3.1 Trace mode:

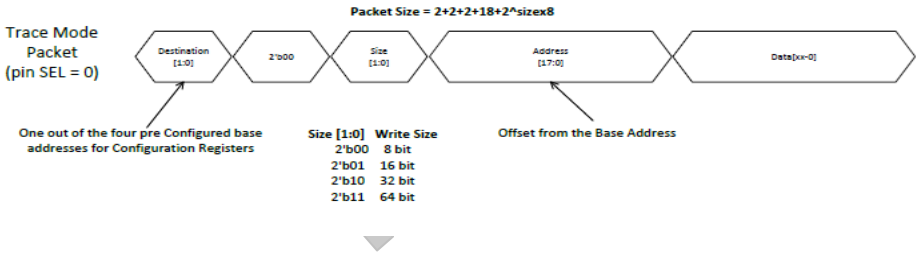


Figure 2: Trace mode data format

This mode is typically used to write to a single register address. One packet consists of 2 bits denoting the destination in which the data is stored, 2 status bits, 2 bit HSIZE, the 18 bit address and actual data bits. DEST[1:0] selects one of the 4 possible destination addresses. Each of the 4 destinations can have a different base destination address as programmed by the user in the DEST0REG1/DEST1REG1/DEST2REG1/DEST3REG1 registers. Once the base address is selected the final destination address is derived by taking the ADDR[17:0] bits as LSB bits and bits [18:31] from the destination register as MSB address bits.

DEST[1:0]	Destination
00	Dest 0
01	Dest 1
10	Dest 2
11	Dest 3

STAT[1:0] are unused.
HSIZE[1:0] selects the data size that needs to be written. Below is the mapping of the HSIZE value to the data size:

HSIZE[1:0]	Write Size
00	8 bit
01	16 bit
10	32 bit
11	64 bit

3.2 Direct Data Mode:

Figure 3: Direct Data mode format

In this mode the packet consists of data only. The bus width can be either 8/16/3 bits wide. The target address is picked from the "DDMDEST". This address can be programmed as the ADC buffer start address, for example.

4. LVDS data to DMM interface packets

Below is the example for the XWR1642 device, scheme would be similar for XWR1843 and XWR6843 family of devices number of available parallel DMM data pins could vary depending upon device family.

The user needs to take the raw LVDS data and convert it 19 bit parallel DMM interface format file. The 19 signals are D[0:15], CLK, SYNC/CTL , MUX_SEL and the DMM format file consists of the full sequence of these 19 bits to be sent to the radar device. In order to feed the DMM data to the radar sensor a programmable logic like an FPGA could be used or processor could be used.

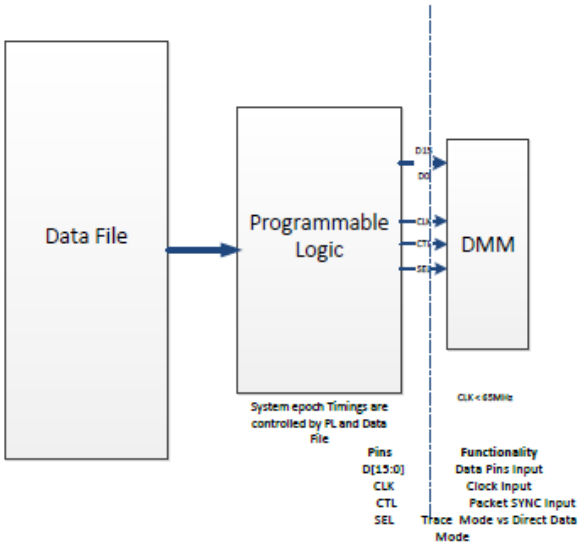


Figure 4: DMM data file data transfer

Below is an example of a Trace mode packet format that need to be provided on the DMM interface pins from the external device. In this example 16bit wide data bus is assumed. This mode is typically used to write a register value.

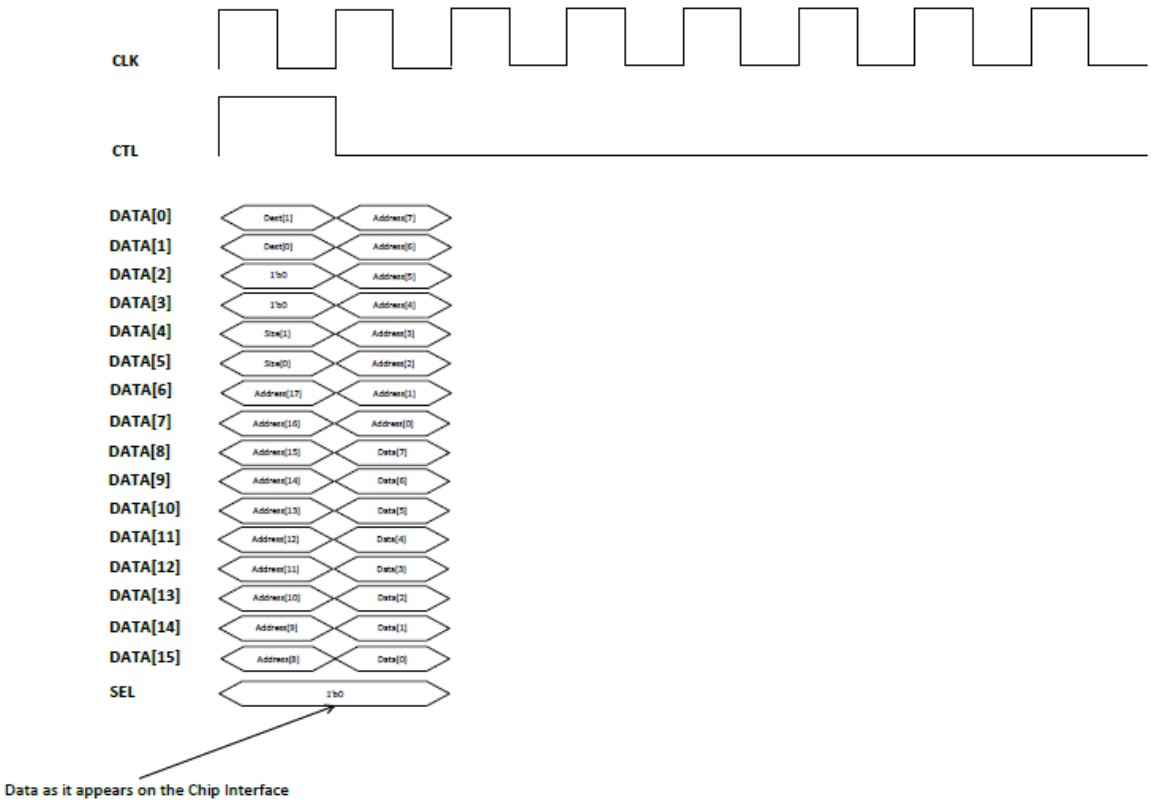


Figure 5: Trace mode packet example

In case of DDM mode the data is written to the destination address in the form of packets. Each packet is defined by a pulse on the CTL or SYNC line which is 1 DMM CLK wide. To start the next packet a second pulse needs to be provided.

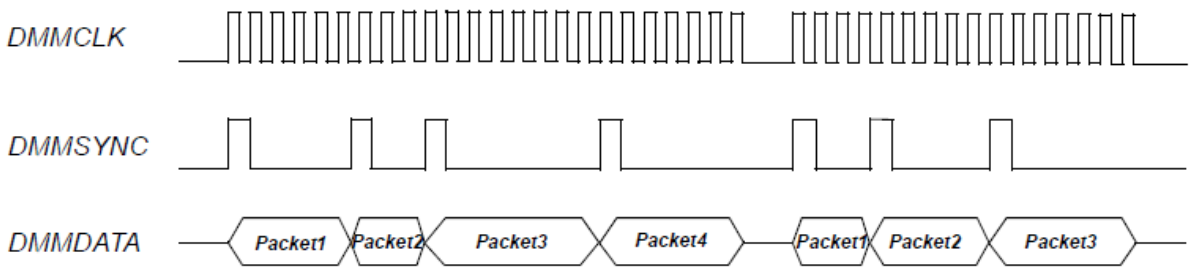
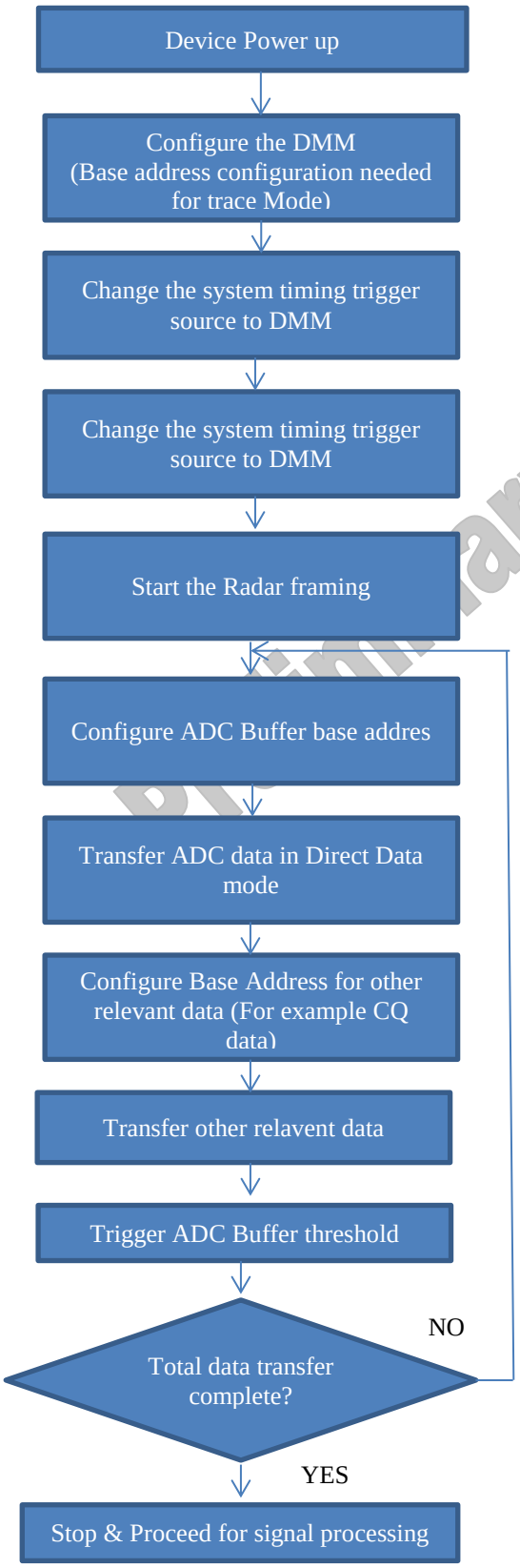


Figure 6: DDM mode packet format

5. Typical sequence for playback



Frame start, ADC buffer threshold etc would come from the DMM interface trigger instead of from the ramp timing engine

Note that software shoud not configure the Radar chirp profiles in BSS. DMM interface is used to set a flag which can be inrepreted by functional software

Configure ADC Buffer address and size in Direct Data module DMM

5.1 Initial DMM interface configuration

The user application needs to perform the following initialization of the DMM IP:

1) Set the PIN MUX to enable the 19 DMM signals on the desired XWR chip ball requirments
2) A reset pulse (0-1-0) is provided by the DMM IPs (1 and 2) using bit 16 of the “GLBCTRL” register of both DMMs.
3) DMM interface pins are configured in functional mode using the “DMMPC0” register of both DMMs.
4) The DMM1 IP is configured to Trace mode and DMM2 IP is configured to DDM mode and the bus widths are selected to 16bit. This is done in the “GLBCTRL” register.
5) The DMM1 IPs destination0 is set to the MSS register space. This is done by writing the value 0xFCFFF700 in the “DEST0REG1” register.

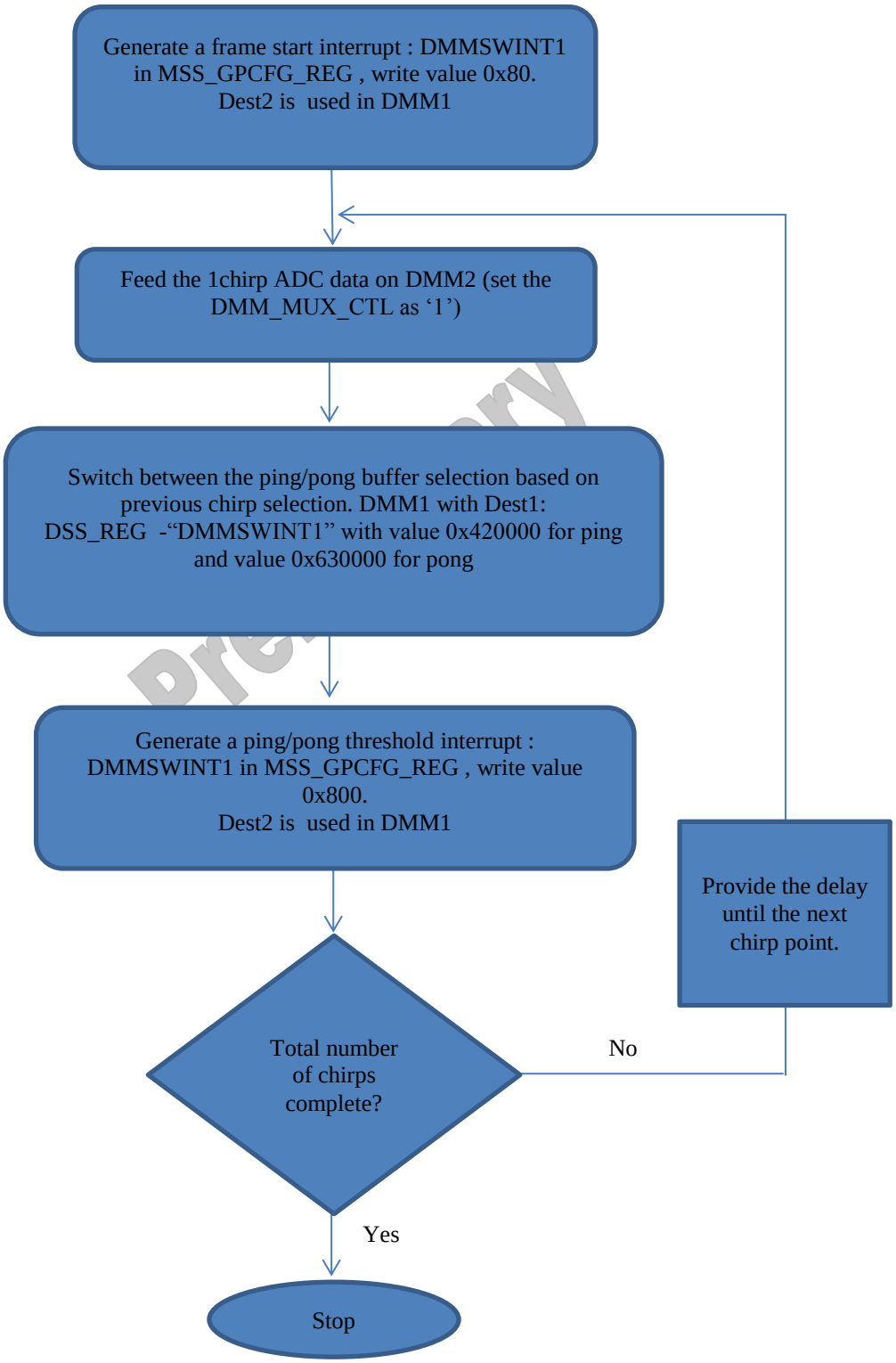
5.2 Below sequence should be followed to write into ADC_BUFFER configuration via the DMM1 interface

The next set of configurations can be done using the DMM1 interface in trace by writing into the desired address. The DEST0 is used for this.

1) The DMM2 destination address is set to be the ADC buffer base address. This is done by writing a value of 0x52000000 (this is ADC buffer base address) to DMM2_DDMDEST register.
2) The block size is set to 32K by writing value 0xB to the DMM2_BLOCKSIZE register.
3) The DMM1 destination 1 is set to DSS register space. This is done by writing the value 0x50000400 in the “DEST1REG1” register.
6) The DMM2 destination 2 is to the second MSS register space. This is done by writing the value 0xFFFFF800 in the “DEST2REG1” register.
7) Select the MUX to enable writing the ADC buffer via DMM interface instead of the front end. This is done by writing the DSS_REG -“DMMSWINT1” with value 0x420000. DMM1 interface with DEST1 is used to write this register.
8) Select the MUX for ping/pong threshold interrupt and frame start interrupt to come from DMM interface. This is done by writing the value 0x880 in “DMMSWINTSEL1” in MSS_GPCFG_REG set. DMM1 interface with DEST2 is used for this.

5.3 ADC data input via DMM2 interface

Flow chart to write the chirp data into the ADC buffer and trigger interrupts is shown below. A similar flow could be followed for the CQ and CP buffers as well.


Generate a frame start interrupt : DMMSWINT1 in MSS_GPCFG_REG , write value 0x80.
Dest2 is used in DMM1
Feed the 1chirp ADC data on DMM2 (set the DMM_MUX_CTL as '1')
Switch between the ping/pong buffer selection based on previous chirp selection. DMM1 with Dest1: DSS_REG -"DMMSWINT1" with value 0x420000 for ping and value 0x630000 for pong
Generate a ping/pong threshold interrupt : DMMSWINT1 in MSS_GPCFG_REG , write value 0x800.
Dest2 is used in DMM1
Provide the delay until the next chirp point.
Total number of chirps complete?
No
Yes
Stop

6. Psudo code for the configuration and data transfer for the HIL operation

Below example provides the sequence of operations needed to be done before initiating any transfer on DMM pins, below configurations needs to be done in mmWave sensor device. This example is provided for XWR1642 device.

// DMM IO Pin mux Configuration must be done prior to following operaiton

// DMM1 Reset
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF700, 0x00010000);
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF700, 0x00000000);

// DMM1 Configuration (16 pin pads in Functional Mode)
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF76C, 0x0007FFFF);
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF78C, 0x0007FFFF);

// DMM1 Configuration in Trace Mode
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF700, 0x0004000A);

// DMM1 Configuration (Destination Register 0 with DMM2 Base Address)
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF72C, 0xFCFFF700);
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF730, 0x00000009);

// DMM2 Reset
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF600, 0x00010000);
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF600, 0x00000000);

// DMM2 Configuration (16 pin pads in Functional Mode)
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF66C, 0x0007FFFF);
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF68C, 0x0007FFFF);

// DMM2 Configuration in DDM Mode
WRITE MEMORY ADDRESS WITH THE VALUE (0xFCFFF600, 0x0004050A);

Once the above configuration is done for XWR16 device, DMM is ready to receive data from outside.
For the XWR16 HIL usecase, below sequence should be followed to write into ADC_BUFFER.

Use **DMM1 in Trace mode** with **Dest0** to write into below registers and below mentioned values

0xFCFFF61C (DMM2_DDMDEST)	0x52090000 (ADC Buffer PING Base Address)
0xFCFFF620 (DMM2 BlockSize)	0xB (Block Size → 32K)
0xFCFFF73C (DMM1 DESTREG1)	0x50000400 (DSS REG Base Address)
0xFCFFF740 (DMM1 DEST1BL1)	0x00000009 (Block Size)

Use **DMM1 in Trace mode** with **Dest1** to write into below registers and below mentioned values

0x50000660 (DSSSWINT1)	0x00420000 (ADC Buffer Write Enable from DMM)

Once these configurations are done **DMM_MUX_CTL** should be driven to value "1" to use DMM2

Now start writing into ADC_BUF PONG through **DMM2** in **DDM Mode** (32KB)

Once PING memory is written switch back to DMM1 by making **DMM_MUX_CTL** as "0"

Use **DMM1 in Trace mode** with **Dest0** to write into below registers and below mentioned values

0xFCFFF61C (DMM2_DDMDEST)	0x52098000 (CQ Buffer Base Address)

Switch back to DMM2 by making **DMM_MUX_CTL** as "1"

Now start writing into CQ Buffer through **DMM2** in **DDM Mode** (32KB)

The above mentioned steps are repeated to fill PING buffers by programming required bits in DSSSWINT1 register in DSSREG space

7. Refernces

1. Technical Reference Manual http://www.ti.com/lit/ug/swru522d/swru522d.pdf
2. XWR1843 Datasheets
3. XWR1642 Datasheets
4. XWR6843 Datasheets