Errata

IWRL6432W Device Silicon Errata Silicon Revision 2.1

ABSTRACT

This document describes the known exceptions to the functional specifications (advisories). This document may also contain usage notes. Usage notes describe situations where the device's behavior may not match the presumed or documented behavior. This may include behaviors that affect device performance or functional correctness.

Table of Contents

1 Introduction

This document describes the known exceptions to the functional and performance specifications to TI CMOS Radar Devices (IWRL6432W)

2 Device Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of Radar / mmWave sensor devices. Each of the Radar devices has one of the two prefixes: X2Ix or IWRLx (for example: **X2I6432** QGYFF). These prefixes represent evolutionary stages of product development from engineering prototypes (X2I) through fully qualified production devices (IWRL).

Device development evolutionary flow:

X2I —	Experimental device that is not necessarily representative of the final device's electrical specifications and may not use production assembly flow.
IWRL —	Production version of the silicon die that is fully qualified.

X2I devices are shipped with the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

Texas Instruments recommends that these devices not to be used in any production system as their expected end –use failure rate is still undefined.

3 Device Markings

Figure 3-1 shows an example of the IWRL6432W Radar Device's package symbolization.

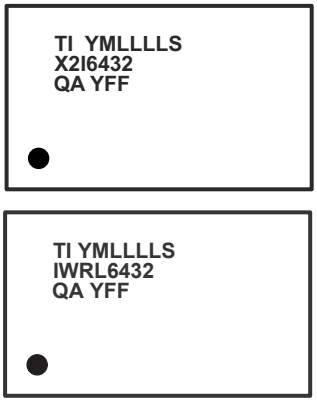

Figure 3-1. Example of Device Part Markings

This identifying number contains the following information:

- **Line 1:** TI Letters
- **Line 1:** Lot Trace Code
  - YM = Year/Month Code
  - LLL = Assembly Lot
  - S = Primary Site Code
- **Line 2:** Device Number
- **Line 3:** Safety Level and Security Grade
  - Q = Non-Functional Safety
  - G = General
  - A = Authenticated boot
- **Line 3:**
  - YFF = Package Identifier

4 Advisory to Silicon Variant / Revision Map

Table 4-1. Advisory to Silicon Variant / Revision Map

Advisory Number	Advisory Title	IWRL6432W
		ES2.1
Analog / Millimeter Wave		
ANA #51	Continuous Wave Streaming CZ mode: Sudden jump in RX output codes every 20.97152 msec	x
ANA#57	SNR degradation at 60 GHz in the presence of strong near range reflector	x
Digital Subsystem		
DIG #1	ePWM: Glitch during Chopper mode of operation	x
DIG #3	Limited UART baud rates	x
DIG #4	RS232 AutoBaud Rate feature doesn't support trimmed ROCSC variation.	x
DIG #5	Internal Bus access to SPI for data transfer not supported when SPI smart-idle mode is enabled.	x
DIG #6	CRC: CRC 8-bit data width and CRC8-SAE-J1850 and CRC8-H2F possible use in CAN module is not supported	x
DIG #8	Shared RAM clock gating default values	x
DIG #9	TOP_IO_MUX register space not accessible from RS232 for debug purposes.	x
DIG #10	Incorrect behavior of frame stop API	x
DIG #14	Corrupted Data Store for Partial Write in Shared Memory	x
DIG #15	Boot failure, if metaimage is multiple of 2K	x
DIG #16	Boot failure for images less than size 8k over SPI	x

5 Known Design Exceptions to Functional Specifications

ANA #51	***Continuous Wave Streaming CZ mode: Sudden jump in RX output codes every 20.97152 msec***
Revision(s) Affected	IWRL6432W ES2.1
Details	On Continuous Wave Streaming CZ mode, the Rx data shows a sudden jump in output codes every 20.97152 milliseconds. This is not an issue in the Radar Functional mode when chirps are used. However, this issue will be seen when testing Rx chain in lab using continuous stream mode.
Workaround	In order to use Continuous stream (CW) mode for testing, it is recommended to start data capturing from the first sample itself to make sure the glitch occurs at deterministic samples. Please follow the below sequence to achieve this: • Configure the RDIF (Radar Data Interface) • Arm the DCA1000 (Data capture card) • Enable the continuous stream mode. The glitch will not be seen with this sequence. For example, if the user analyzes first 20ms of data or between 21 and 41ms.

ANA #57	***SNR degradation at 60 GHz in the presence of strong near range reflector***
Revision(s) Affected	IWRL6432W ES2.1
Details	There is a non-linearity of the synthesizer when crossing 60-GHz which causes increased noise floor at RX output in the presence of a strong near range reflector.
Workaround	Chirps with large RF bandwidth (> 1.5GHz) will have negligible noise floor impact. For lower bandwidth chirps, avoid 60GHz.

DIG #1 ***ePWM: Glitch during Chopper mode of operation***

Revision(s) Affected

IWRL6432W ES2.1

Details

During chopper mode operation, a glitch may be observed on the ePWMA and ePWMB output signals from the ePWM module.

Workaround

If the use case is impacted by a glitch, it is recommended to disable the PWM chopper control function by setting the LPRADAR:APP_PWM:PCCTL:CHPEN register bit to 0.

The below table shows the Register Address for above workaround.

Bits	Name	Address
0	LPRADAR:APP_PWM:PCCTL:CHPEN	0X57F7 FC3C

DIG #3 *UART: Limited UART baud rates*

Revision(s) Affected

IWRL6432W ES2.1

Details

Due to a design limitation (related to the clocking scheme), UART doesn't support standard baud rates above 115200 bits per second. Higher baud rates up to 1.25Mbps can be supported but they are non-standard.

Applications requiring UART cannot use standard baud rates above 115200 bits per second

Standard Baud Rates supported :

XTAL (MHz)	40	
Ideal Baud rate (bps)	Actual Baud	Error %
115200	113636.36	1.36
76800	75757.58	1.36

Non- Standard baud rates supported:

XTAL (MHz)	**40**
Maximum baud (bps)	1250k
	833.33k
	625k
	500k
	416.66k
	357.14k
	312.5k

Workaround

It is recommended to use the following workarounds based on application needs:

- Use of non-standard baud rates can provide up to 1.25Mbps throughput, if external MCU can support the same non-standard baud rates.
- Use SPI instead, if use-case needs higher throughput.

DIG #4	***RS232: Auto Baud Rate feature doesn't support trimmed RCOSC variation***
Revision(s) Affected	IWRL6432W ES2.1
Details	Once RCOSC is trimmed, the expected clock frequency and the variation observed in frequency (tolerance on RC clock) do not support the required Auto Baud rate setting for RS232. Currently Auto Baud is disabled by default for 64xx ES2.1

DIG #5 ***Internal Bus access to SPI for data transfer not supported when SPI smart-idle mode is enabled.***

Revision(s) Affected

IWRL6432W ES2.1

Details

Smart-idle mode needs to be disabled for SPI before the first trigger for data transfer access. If the SPI smart-idle mode is required to be enabled, it has to be enabled again once the access is complete.

Workaround

It is recommended to follow the below sequence:

Auto Wake-up = 1 & Controller mode

1. Configure McSPI as required
2. Enable SmartIdle (by setting LPRADAR:APP_CTRL:SPI1_SMART_IDLE_ENABLE for SPI1 and LPRADAR:APP_CTRL:SPI2_SMART_IDLE_ENABLE for SPI 2)after ensuring that there is **no** pending transaction from/to SPI or any more access to be done to McSPI by CPU or DMA
3. If any register or memory access to McSPI has to be done, disable SmartIDLE mode (by setting LPRADAR:APP_CTRL:SPI1_SMART_IDLE_ENABLE=0 for SPI 1 and LPRADAR:APP_CTRL:SPI2_SMART_IDLE_ENABLE =0 for SPI 2)
4. In Controller mode, the external host is not going to toggle the SPI_CS, hence there will not be any wakeup => there is no difference between (LPRADAR:APP_CTRL:SPI1_SMART_IDLE_AUTO_EN is 1 or 0 for SPI 1 and LPRADAR:APP_CTRL:SPI2_SMART_IDLE_AUTO_EN is 1 or 0)

Auto Wake-up = 1 & Peripheral mode

1. Configure McSPI as required
2. Enable SmartIdle (by setting LPRADAR:APP_CTRL:SPI1_SMART_IDLE_ENABLE for SPI1 and LPRADAR:APP_CTRL:SPI2_SMART_IDLE_ENABLE for SPI 2) after ensuring that there is **no** pending transaction from/to SPI or any more access to be done to McSPI by CPU or DMA
3. If any register or memory access to McSPI has to be done by any master (DMA / CPU), disable SmartIDLE mode (by setting LPRADAR:APP_CTRL:SPI1_SMART_IDLE_ENABLE=0 for SPI 1 and LPRADAR:APP_CTRL:SPI2_SMART_IDLE_ENABLE =0 for SPI 2)
4. If there is wakeup from McSPI (because of some SPI_CS toggle), then the clock is automatically enabled.
5. Disable SmartIdle configuration (by setting LPRADAR:APP_CTRL:SPI1_SMART_IDLE_ENABLE=0 for SPI 1 and LPRADAR:APP_CTRL:SPI2_SMART_IDLE_ENABLE =0 for SPI 2) to do the register access.

The below table shows the Register Addresses for above workaround.

Bits	Name	Address
0	LPRADAR:APP_CTRL:SPI1_SMART_IDLE_ENABLE	0x560603A8
2	LPRADAR:APP_CTRL:SPI1_SMART_IDLE_AUTO_EN	0x560603A8
0	LPRADAR:APP_CTRL:SPI2_SMART_IDLE_ENABLE	0x560603AC

DIG #5 (continued) ***Internal Bus access to SPI for data transfer not supported when SPI smart-idle mode is enabled.***

2	LPRADAR:APP_CTRL:SPI2_SMART_IDLE_AUTO_EN	0x560603AC

DIG #6	***CRC: CRC 8-bit data width and CRC8-SAE-J1850 and CRC8-H2F possible use in CAN module is not supported***
Revision(s) Affected	IWRL6432W ES2.1
Details	1. 8-bit data width is not supported. Minimum data width supported is 16-bit. 2. CRC types CRC8-SAE-J1850 and CRC8-H2F are not supported.
Workaround	1. 16/32/64-bit data widths are supported. 2. It is recommended to not use the above mentioned unsupported polynomials.

DIG #8 *Shared RAM clock gating default values*

Revision(s) Affected IWRL6432W ES2.1

Details

Possibility of Shared RAM data corruption while exiting from deep sleep mode when clock gating registers are not reprogrammed.

The reset value for Front End Controller Sub System (FECSS), Application Sub System (APPSS) and Hardware Accelerator Sub System (HWASS) shared memory clock gate control is 1 . The clock ICG controls are coming from the following registers.

Bits	Name	Address
0	LPRADAR:FEC_CTRL:FECSS_SHARED_MEM_CLK_GATE : FECSS_SHARED_MEM_CLK_GATE_HWA_ENABLE	0x5200002C
0	LPRADAR:APP_CTRL:APPSS_SHARED_MEM_CLK_GATE:APPSS_SHARED_MEM_CLK_GATE_MEM0_HWA_ENABLE	0x56060398
2	LPRADAR:APP_CTRL:APPSS_SHARED_MEM_CLK_GATE:APPSS_SHARED_MEM_CLK_GATE_MEM1_HWA_ENABLE	0x56060398

When APPSS tries to access shared memory bank 0 via VBUSM SCR while FECSS is accessing shared memory via AHB, wrong read values of zero from the shared RAM on the APPSS is observed .

If only one of the clock gates (either HWA or FEC/APP) is enabled based on the allocation, the data is read correctly. Since the clock gating controls are coming from control registers space, these values get reset again and hence needs to be re-programmed after every deep sleep exit.

Workaround

Program ICG controls of clock reaching to shared memory based on different shared memory configuration. The ICG control needs to be re-programmed after every deep sleep exit too.

Configuration	Software care-about
Memory is shared with M3	Disable the following ICG control :- LPRADAR:FEC_CTRL:FECSS_SHARED_MEM_CLK_GATE : FECSS_SHARED_MEM_CLK_GATE_HWA_ENABLE
First 128kb is shared with M4	Disable the following ICG control :- LPRADAR:APP_CTRL:APPSS_SHARED_MEM_CLK_GATE:APPSS_SHARED_MEM_CLK_GATE_MEM0_HWA_ENABLE

DIG #8 (continued) ***Shared RAM clock gating default values***

256kb is shared with M4	Disable the following ICG controls :- • LPRADAR:APP_CTRL:APPSS_SHARED_MEM_CLK_GATE:APPSS_SHARED_MEM_CLK_GATE_MEM0_HWA_ENABLE • LPRADAR:APP_CTRL:APPSS_SHARED_MEM_CLK_GATE:APPSS_SHARED_MEM_CLK_GATE_MEM1_HWA_ENABLE

DIG #9 ***TOP_IO_MUX register space not accessible from RS232 for debug purposes***

Revision(s) Affected IWRL6432W ES2.1

Details

RS232 is not able to write TOP_IO_MUX registers unless the space is programmed for user mode access.

Workaround

It is recommended to use the following sequence:

1. From Processor or DAP : Unlock TOP_IO_MUX registers (by programming LPRADAR:TOP_IO_MUX:IOCFGKICK0 = 83E7 0B13h and LPRADAR:TOP_IO_MUX:IOCFGKICK1 = 95A4 F1E0h)
2. From Processor or DAP : Write to TOP_IO_MUX registers, LPRADAR:TOP_IO_MUX:USERMODEEN should be set to 0xADADADAD
3. Now TOP_IO_MUX registers can be accessed from RS232.

The below table shows the Register Addresses for above workaround.

Bits	Name	Address
0:31	LPRADAR:TOP_IO_MUX:IOCFGKICK0	0x5A000068
0:31	LPRADAR:TOP_IO_MUX:IOCFGKICK	0x5A00006C
0:31	LPRADAR:TOP_IO_MUX:USERMODEEN	0x5A000060

DIG #10 ***Incorrect behavior of frame stop API***

Revision(s) Affected IWRL6432W ES2.1

Details

The Frame Timer latches Frame Stop command in hardware registers which takes affect at the end of current frame. Frame Stop API issued when Frame Timer has already stopped will result in un-intended stop in the next frame trigger because of the latched stop bit.

Workaround

1. Unnecessary Sensor Stop API should be avoided.
2. The application may have to wait for one complete frame period before getting frame stop.
3. Application should wait for FECSS to complete Burst End and Frame End activities after receiving the Frame Stop conformation.

DIG #14 *Corrupted Data Store for Partial Write in Shared Memory*

Revision(s) Affected IWRL6432W ES2.1

Details

Internal shared memory has ODD and EVEN banking structure. For a particular address range, partial write (less than 32 bit) to EVEN bank corrupts same address of ODD bank with next data on the bus. When shared memory is allocated to M4/M3, back to back full word write access to location A followed by sub-word write access to location B corrupts data in location A.

When memory is shared with M4/M3, issue is seen in the following address range:

Memory	Address Range
APP_CPU_SHARED_RAM	0x0048 0000 - 0x004B FFFC
FEC_CPU_SHARED_RAM	0x2120 8000 - 0x2121 FFFC

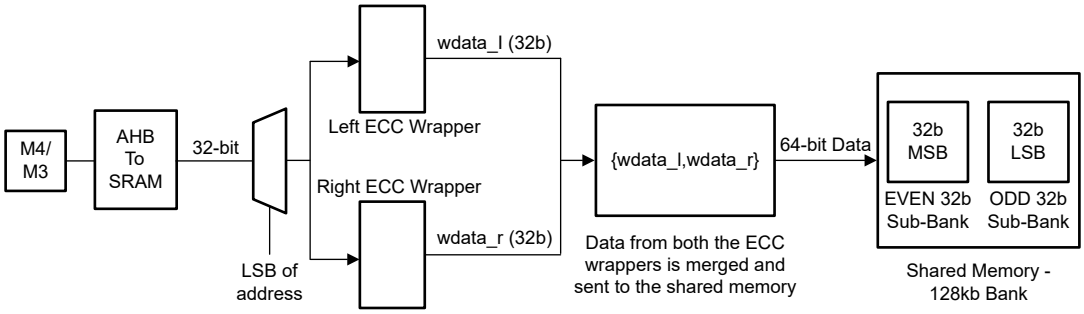

Figure 5-1. Shared Memory Logic Diagram When Shared with M4/M3

When shared with M3/M4, the incoming data bit width is 32 bit as shown in the diagram. So, depending on LSB of address, signals are sent to either left or right ECC wrapper.

Workaround

1. Use shared memories as code memory when shared with processor.
2. Disable ECC for non functional safety devices – ECC is disabled for shared memories in RBL for non functional safety devices.

DIG #15	***Boot failure, if metaimage is multiple of 2K***
Revision(s) Affected	IWRL6432W ES2.1
Details	Metaimages that are a multiple of 2048 bytes will fail to boot.
Workaround	1. Add a constant non-volatile variable to increase the metaimage size, so that it is not aligned to 2048 bytes. 2. Update MMWAVE-L-SDK to version 5.4 or above; mmWave LSDK 5.4 and above includes changes to the metaimage generator tool to add a minimal config file (~64 bytes) in case the image is a multiple of 2048 bytes.

DIG #16 *Boot failure for images less than size 8k over SPI*

Revision(s) Affected

IWRL6432W ES2.1

Details

The EDMA address linking is not done in few cases (during SPI continuous download), due to which boot will fail over SPI continuous download image for the particular metaimage size ranges mentioned in the below table:

Image Size (Bytes)	Issue Present
<2048	No
>2048 & <4096	No
>=4096 & <6144	Yes
>=6144 & <8192	Yes
>=8192	No

Workaround

Use image >8KB for boot over SPI. In case of lower image size, constant data will be appended during compile time to create an image >8 KB.

6 Trademarks

All trademarks are the property of their respective owners.

Revision History

DATE	REVISION	NOTES
December 2024	*	Initial Release

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATA SHEETS), 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, regulatory 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 or other applicable terms available either on 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.

TI objects to and rejects any additional or different terms you may have proposed.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265