Known Errors in the IRL1206FI microcontroller

This document provides a description of all errors identified in the integrated circuits as of the creation of the current document revision.

Document status

This document is NON-CONFIDENTIAL.

Website

http://www.irlabs.in

Product feedback

If you have any comments or suggestions regarding this product, please, contact your supplier and we kindly ask you to provide the following information:

– Product name

– Comments or a brief description of your suggestions

– Preferred method of contact and your contact details (organization, email, phone number).

Document feedback

If you have any comments or suggestions regarding this document, please, email them to techsupport@irlabs.in, specifying:

– Document title

– Document number and/or date

– Page

– Comments or a brief description of your suggestions

– Preferred method of contact and your contact details (organization, email, phone number).

Table of Contest

Overview

This document provides a description of product errors, including their severity categories. Each description includes:

– A unique identifier

– The current status of the error

– The deviation from the specifications and the conditions under which it occurs

– Impact of the error in typical applications

– Limitations, recommendations, and workarounds where applicable.

Error categories

Errors are classified into three severity categories:

Category 1.

Erroneous behaviour with no workaround. These errors severely restrict the use of the product in all or most applications, making the device unusable.

Category 2.

Erroneous behaviour that deviates from the expected operation. These errors may limit or severely impair the intended functionality but do not render the product unusable in all or most applications.

Category 3.

Erroneous behaviour that was not originally specified, but does not impact application performance when the guidelines are followed.

Error summary table

The table shows which chip versions are affected by each error. An error is indicated by the symbol "X".
Chip versions are identified by the manufacturing date stamped on the chip body, in the YYNN format, where YY is the year and NN is the week of manufacture.

ID	Description	Chips manufactured from the date	
		IRL1206FI	
		2345 (rev. 1)	2441 (rev. 2)
Category 1			
Category 2			
Category 3			
0001	SAR ADC channel skipped during sequential conversion of multiple channels after Analog-to-Digital Converter (ADC) shutdown	X	X
0002	Auxiliary LDO in the battery domain fails to start	X	X
0003	Interrupt requests are not processed in debug mode during single-step execution if a vector interrupt is active	X	
0004	dret instruction is not recognized as illegal when executed outside of Debug mode	X	X
0005	Hardware breakpoint malfunction when set on code in Debug ROM.	X	X
0006	Error in the tinfo register of the debug module	X	X
0007	c.lwsp instruction (when rd == x0) is not decoded as illegal	X	X
0008	Reading CSR registers using the csrrc, csrrsi, or csrrci instructions triggers recording	X	X
0009	Incorrect reset value of the sbaccess field in the sbcs register	X	X

ID	Description	Chips manufactured from the date	
		IRL1206FI	
		2345 (rev. 1)	2441 (rev. 2)
0010	Core halts during handling of level-triggered, non-vectored interrupts if the level goes inactive before the interrupt is fully taken	X	X
0011	IWDG stops counting when PCLK frequency is absent during timer reload or prescaler update	X	X
0012	Undefined state of JTAG TDO (PA7) pin after power up	X	X
0013	SEC_CLK pulse is skipped when RTC_CLK calibration is enabled	X	X
0014	HSE_RDY flag gets “stuck” after frequency loss on OCS_IN and a non-power reset	X	X
0015	I2C interface controller does not track clock stretching during START condition generation and when DIV=1-2	X	X
0016	DMA channel stall upon a repeated ADC request during current transaction processing	X	X
0017	Data presence in the Information Flash Memory (BOOT region)	X	X
0018	Zero FxPHASE values at low PER_LENGTH and low sampling rates	X	X
0019	Debug JTAG shift register is not cleared upon exiting the Capture-DR state when the BYPASS register is selected	X	X

Category 3 errors

0001 SAR ADC channel skipped during sequential conversion of multiple channels after ADC shutdown

Status
Under investigation.

Description
In sequential conversion mode with multiple channels, if the ADC (Gfg_REG_ADON bit) is disabled and then re-enabled, the channel where the conversion was stopped at the moment the ADC shutdown is skipped once. Conversion resumes from the next channel involved in the sequential conversion.

Conditions
Disabling and then re-enabling the ADC in the sequential conversion mode with multiple channels.

Impact
Skipping the conversion of the ADC channel that was in progress at the time of shutdown.

Recommendations and workarounds
After disabling the ADC when using sequential conversion of multiple channels:

1 Disable channel switching (Cfg_REG_CHCH bit).
2 Enable channel switching only for the channel that was being converted at shutdown and any higher-numbered channels involved in the conversion (ADC1_CHSEL register).

After re-enabling the ADC:

3 Re-enable switching for all required channels.

0002 Auxiliary LDO in the battery domain fails to start

Status
Under investigation.

Description
The auxiliary LDO of the battery domain may fail to start if the V_{CCV} power ramp does not meet specification requirements.

Conditions
V_{CCV} power ramp time is more than 1 ms or shorter than 10 µs.

Impact
The battery domain will operate using the main power supply and the main LDO. However, if the main power (V_{CCA} and V_{CC}) is disconnected, the data in the battery domain may be lost, even if there is battery power.

Recommendations and workarounds
After applying power to the VCCB pin, set the BLDO_BOOST bit of the BKP_LDO register to “1” once, then set it back to “0” after a delay of at least 20 µs.

0003 Interrupt requests are not processed in debug mode during single-step execution if a vector interrupt is active

Status
Fixed in Revision 2.

Description
In debug mode, when executing step by step (stepi command), the presence of an active vectored interrupt causes incorrect changes to the control and status registers (CSR) (xstatus, xepc, xcause, xtval, xintstatus). An incorrect change to xstatus prevents the core from processing any interrupt requests during subsequent program execution.

Conditions
An active vectored interrupt is present at the time of step execution (stepi command) in debug mode.

Impact
Incorrect updates of control and status registers (CSR) (xstatus, xepc, xcause, xtval, xintstatus) prevent accepting interrupts after issuing the continue command (a debug mode command).

Recommendations and workarounds
In the debug version of the program, implement one of the following workarounds.

Method 1. Set all used interrupts to operate in non-vectored mode: set the shv bit to “0” in the clicintattr[i] register.

Method 2. Configure all vectored interrupts except for DMA and EXT_INTx for level-triggered interrupt capture: set the trig[0] bit to “0” in the clicintattr[i] register. Configure DMA and EXT_INTx interrupts (if used) for non-vectored mode: set the shv bit to “0” in the clicintattr[i] register. Enable interrupt masking in OpenOCD during step execution using the set_maskisr command. This can be done in one of the following ways:

- Add to the OpenOCD configuration file: `riscv set_maskisr steponly`;
- Execute in GDB: `monitor riscv set_maskisr steponly`.

In this case, OpenOCD will save the current value of mstatus before executing the stepi command, then clear the global interrupt enable bits xIE in mstatus. After stepi completes, OpenOCD will restore the previously saved mstatus value. This workaround is effective if the stepi command is used on code that does not modify mstatus directly or indirectly.
An OpenOCD version with set_maskisr support is available on the technical support website of PKK Milandr IC Design Centre: https://irlabs.in/our-products/microcontrollers-and-microprocessors/ir51206fi

0004 dret instruction is not recognized as illegal when executed outside of Debug mode

Status
Under investigation.

Description
Incorrect decoding of the dret instruction (command) outside of Debug mode.

Conditions
The dret assembly instruction (command) outside of Debug mode.

Impact
The instruction is not decoded as illegal.

Recommendations and workarounds
Do not use dret in user program code.

0005 Hardware breakpoint malfunction when set on code in Debug ROM

Status
Under investigation.

Description
Breakpoints set on code in the Debug ROM do not function.

Conditions
A breakpoint is set on code in Debug ROM.

Impact
An error occurs.

Recommendations and workarounds
Avoid debugging in the Debug ROM.

0006 Error in the tinfo register of debug module

Status
Under investigation.

Description
An error in the tinfo register causes issues with the Discovery mechanism.

Conditions
Use of OpenOCD with the Discovery mechanism enabled.

Impact
An error occurs.

Recommendations and workarounds
I. Use the mainline OpenOCD from https://github.com/openocd-org/openocd, version 0.12.0 or earlier.
II. 1. Use OpenOCD for RISCV https://github.com/riscv/riscv-openocd, version 2018.12.0 or earlier;
2. Use patch* for OpenOCD for RISCV version 0.12.0+ dev-03629-g87331a82a. This patch replaces the value read from the tinfo register with the correct one.

0007 c.lwsp instruction (when rd == x0) is not decoded as illegal

Status
Under investigation.

Description
The c.lwsp command (instruction) must be decoded as an illegal instruction.

Conditions
Execution of c.lwsp command (instruction) (if rd == x0).

Impact
The instruction is not decoded as illegal.

Recommendations and workarounds
The C compiler does not generate these commands, so do not use inline assemblies in C code that include such commands.

0008 Reading CSR registers using the csrrc, csrrsi, or csrrci instructions triggers recording

Status
Under investigation.

Description
The commands (instructions) csrrs, csrrc, csrrsi, and csrrci are intended to allow reading of CSR registers without performing a write operation when rs1 = x0 (csrrs, csrrc) or uimm = 0 (csrrsi, csrrci):

- `csrrs rd, csr, x0;`
- `csrrc rd, csr, x0;`
- `csrrsi rd, csr, 0;`
- `csrrci rd, csr, 0.`

However, the csrrc, csrrsi, and csrrci commands, when used to perform a read of CSR registers, also incorrectly perform a write operation, which causes write side effects (for example, with the xNXTI register) or triggers an illegal instruction exception when accessing a read-only CSR register.

Conditions
Execution of a CSR register read using the csrrc, csrrsi, and csrrci commands when rs1 = x0 (csrrc) and uimm = 0 (csrrsi, csrrci).

Impact
In addition to the read operation, a write operation is also performed, which does not change the contents of the register (since the set/reset bit mask is 0), but causes write side effects (for example, for the xNXTI register) or triggers an illegal instruction exception when accessing a read-only CSR register.

Recommendations and workarounds
To perform a read operation on CSR registers without executing a write operation, use the csrrs command with rs1 = x0, or the csrr pseudo-command (encoded as csrrs with rs1 = x0):

- `csrrs rd, csr, x0;`
- `csrr rd, csr.`

0009 Incorrect reset value of the sbaccess field in the sbcs register

Status
Under investigation.

Description
According to the RISC-V External Debug Support Version 0.13 specification, the sbaccess field, which defines the data width for memory access via System Bus Access, should be set to a reset value of 2, which corresponds to a 32-bit transaction size. In practice, the reset value of the sbaccess field is 0, which corresponds to a 8-bit transaction size.

Conditions
Always.

Impact
Memory access in debug mode using System Bus Access will be performed with a data width different from what is expected.

Recommendations and workarounds
Set the sbaccess field to the required value before performing any operations using System Bus Access.

0010 Core halts during handling of level-triggered, non-vectored interrupts if the level goes inactive before the interrupt is fully taken

Status
Under investigation.

Description
A processor core hang (the PC counter stops, no valid exchange occurs on the AHB bus) may occur during handling of non-vectored, level-triggered interrupts. This condition may arise of the interrupt level changes to inactive after the processing started and before the interrupt has been fully taken.

Conditions
1 The interrupt is configured to be level-triggered.
2 The interrupt is non-vectored.
3 The interrupt request level changes to inactive before the interrupt is fully taken.

Impact
The microchip core stops.

Recommendations and workarounds
In the CLIC controller, configure the used interrupts in one of the following modes:
1 Vectored mode with either level-triggered or edge-triggered activation;
2 Non-vectored with edge-triggered activation.

0011 IWDG stops counting when PCLK frequency is absent during timer reload or prescaler update

Status
Under investigation.

Description
During IWDG operation, if the PCLK frequency of the IWDG block stops after a timer reload request is issued via the IWDG_KEY register (by writing the value 0xAAAA) and before the timer value is actually updated, or after a prescaler update request is issued (by writing to the IWDG_PR register) and before the prescaler value is actually updated, the timer update request signal becomes stuck in the active state. As a result, until the PCLK frequency is restored or any reset occurs, the IWDG watchdog timer stops counting and does not generate a reset. Because in case of a reset request, the IWDG is continuously reset with the reload value, and in case of a prescaler update request, it continuously updates the prescaler. Once the PCLK frequency reappears and the corresponding values are successfully updated, or after any reset, the IWDG resumes counting.

Conditions
PCLK frequency stop in the IWDG block:
1 After issuing a timer reload request via the IWDG_KEY register (by writing the value 0xAAAA) and before the timer value is actually updated during IWDG operation;
2 After issuing a timer prescaler update request via the IWDG_PR register and before the prescaler value is actually updated during IWDG operation.

Impact
The IWDG timer stops counting and does not generate a reset until the PCLK frequency is restored or any reset occurs.

Recommendations and workarounds
Implement one or more of the proposed methods in the system under development:
1 Before issuing a timer reload request and during the reset of the RVU flag, as well as before issuing a prescaler update request and during the reset of the PVU flag, switch the PCLK clock (which corresponds to the HCLK frequency) to a frequency that is guaranteed to be present in the system under development, for example, LSI (since the presence of the LSI frequency is necessary for IWDG operation; if it is absent, the IWDG will not reset the system under any circumstances);
2 Use both the IWDG and WWDG watchdog timers together, ensuring that the time between their reloads is at least one period of the LSI frequency, and start the WWDG before configuring the IWDG;
3 Use an external watchdog timer.

0012 Undefined state of JTAG TDO (PA7) pin after power up

Status
Under investigation.

Description
The JTAG interface debug TAP controller is reset by the POR (Power-On-Reset) signal during power-up and enters the Test-Logic-Reset state. If the JTAG debug interface is enabled, the PA[9:6] pins are taken under the control of the debug TAP controller.
According to the IEEE 1149.1 JTAG standard, the TAP controller in the Test-Logic-Reset state must place the TDO pin in a high-impedance state. However, due to an error, the state of the TDO pin (PA7) after a power-on reset is undefined — the pin can be in one of the following states: high impedance, logic high, logic low. A guaranteed transition of the TDO pin (PA7) to the high-impedance state after a power-on reset occurs on the falling edge of the signal at the TCK input (PA6).

Conditions
JTAG interface debug TAP controller after power-on reset, JTAG debug interface is enabled.

Impact
The state of the TDO pin (PA7) is undefined: high impedance, logic high, logic low.

Recommendations and workarounds
Take this behaviour into account during development.
To control the TDO pin (PA7) through the I/O port controller, the JTAG interface must be disabled using one of the following methods:

- Clearing the JTAG_ON bit in the BKP_LDO register;
- Enabling Flash memory address space protection;
- Disabling the JTAG interface in the CHIP_ID_CTRL register by programming the OTP memory.

0013 SEC_CLK pulse is skipped when RTC_CLK calibration is enabled

Status
Under investigation.

Description
In the RTC block, the SEC_CLK frequency is generated from the RTC_CLK frequency using a divider based on the BKP_RTC_PREDIV_S counter with the BKP_RTC_PRL counting base. For RTC_CLK calibration (slowing down), the BKP_RTC_PREDIV_A counter is used, which halts the BKP_RTC_PREDIV_S counting while BKP_RTC_PREDIV_A < RTC_CAL.
During RTC operation with RTC_CLK frequency calibration (field RTC_CAL[7:0] != 0 in the BKP_RTC_CR register), certain values of RTC_CAL and BKP_RTC_PRL may periodically lead to the simultaneous occurrence of the events BKP_RTC_PREDIV_S == BKP_RTC_PRL and BKP_RTC_PREDIV_A == 0, which results in an erroneous reset of the BKP_RTC_PREDIV_S counter and a missed SEC_CLK pulse.
When a SEC_CLK pulse is missed, the counter/calendar (registers BKP_RTC_TR/DR) and the watchdog timer BKP_RTC_WUT (when clocked from SEC_CLK: bit WUCK_SEL[2] == 1 in the BKP_RTC_CR register) do not update and begin to lag by 1 second. The corresponding flags in the BKP_RTC_CS register are also not set when a SEC_CLK pulse is missed: SECF, ALRAF, ALRBF, WUTF, and OWF.

Conditions
RTC_CLK frequency calibration is set (field RTC_CAL[7:0] != 0) and specific values of RTC_CAL and BKP_RTC_PRL are set, which periodically result in the simultaneous occurrence of the events BKP_RTC_PREDIV_S == BKP_RTC_PRL and BKP_RTC_PREDIV_A == 0.

Impact
The BKP_RTC_PREDIV_S counter is incorrectly reset, preventing the generation of the SEC_CLK pulse. When a SEC_CLK pulse is missed, the counter/calendar and the BKP_RTC_WUT watchdog timer (when clocked from SEC_CLK) do not update, and the corresponding flags are not set: SECF, ALRAF, ALRBF, WUTF, and OWF.

Recommendations and workarounds
1 Temperature compensation of the SEC_CLK frequency is used.
Do not use RTC_CLK frequency calibration (field RTC_CAL[7:0] == 0); adjust the RTC timing using the CONSTx and BKP_RTC_PRL, taking into account the following features (values are based on a quartz crystal resonator with a nominal frequency of 32,768 Hz):
- RTC_CAL slows down RTC with a step of ~0.95 ppm
- CONSTx speeds up RTC with a step of ~7.63 ppm;
- BKP_RTC_PRL performs both slowdown/speedup of the RTC with a step of ~30.52 ppm.

When adjusting the RTC timing, complete slowdown/speedup using BKP_RTC_PRL (coarse tuning), then apply speedup using CONSTx (fine tuning), increasing all CONSTx values by N, where N is a value from 0 to 3 that provides the optimal result.

Algorithm for calculating BKP_RTC_PRL and CONSTx to adjust the RTC timing by C ppm using a quartz crystal resonator with a nominal frequency of 32,768 Hz:

- Setting BKP_RTC_PRL:
  - K = C ppm / 30.52 ppm;
  - Round K down to the nearest integer;
  - BKP_RTC_PRL = BKP_RTC_PRL(nominal) – K;
- Setting CONSTx:
  - N = (C – K · 30.52) ppm / 7.63 ppm;
  - Rounding N to the nearest integer;
  - CONSTx = CONSTx(nominal) + N;
- Adjustment error in ppm:
  - E = K · 30.52 ppm + N · 7.63 ppm – C ppm.

Example of RTC slowdown (nominal frequency: 32,768 Hz):

- Initial data:
  - Initial RTC_CLK frequency: 32,773.3 Hz, i.e. ~ 32,768 Hz + 161.74 ppm;
  - Required RTC timing adjustment: –161.74 ppm (slowdown);
- Setting BKP_RTC_PRL:
  - K = –161.74 / 30.52 ≈ –5.3 → –6;
  - BKP_RTC_PRL = 32,767 – (–6) = 32773;
- Setting CONSTx:
  - N = (–161.74 – (–6) · 30.52) / 7.63 = 21.38 / 7.63 ≈ 2.8 → 3;
  - CONSTx = CONSTx(nominal) + 3;
- Adjustment error:
  - E = –6 · 30.52 + 3 · 7.63 – (–161.74) = 1.51 ppm.

2 Temperature compensation of the SEC_CLK frequency is not used.

To calibrate the RTC_CLK frequency, use only a limited set of RTC_CAL values obtained generated by the script* for the given BKP_RTC_PRL value. Before configuration, the RTC block must be disabled and reset; during configuration, the BKP_RTC_PREDIV_S register must not be written with any value other than 0. After configuration and enabling of the RTC, the values of RTC_CAL[7:0], BKP_RTC_PREDIV_S and BKP_RTC_PRL should not be modified. In this case, the error will not occur.

Example of RTC initialization:

- Enable the RTC clock source and select it using the RTC_SEL[1:0] field in the BKP_RTC_CR register;
- Wait until a valid RTC_CLK frequency is available, as specified;
- Disable the RTC by clearing the RTC_EN bit in the BKP_RTC_CR register;
- Reset the RTC by setting and then clearing the RTC_RESET bit in the BKP_RTC_CR register;
- Write the required value to the BKP_RTC_PRL register;
- Write a valid value into the RTC_CAL[7:0] field in the BKP_RTC_CR register;
- Write 0 to the BKP_RTC_PREDIV_S register;
- Wait for the write operation to complete using the WEC bit in the BKP_RTC_CS register;
- Perform any other RTC configuration;
- Enable the RTC by setting the RTC_EN bit in the BKP_RTC_CR register.

0014 HSE_RDY flag gets "stuck" after frequency loss on OCS_IN and a non-power reset

Status
Under investigation.

Description
After the HSE is enabled and reaches its operation mode (HSE_RDY = 1), if the HSE frequency on OSC_IN is lost and followed by a non-power reset (such as a reset from IWDG or nRESET), the HSE_RDY flag is set to "1" immediately after the HSE is enabled (HSE_ON = 1), even though no frequency signal is present on the OSC_IN pin.

Conditions
Loss of HSE frequency signal followed by a non-power reset.

Impact
The HSE_RDY flag becomes stuck at 1 and cannot be used to determine HSE readiness.

Recommendations and workarounds
Use the following algorithm:
1. Enable HSE as usual and wait until it is ready (HSE_RDY = 1).
2. Configure the PLL to use HSI as the clock source (CPU_C1 = HSI, ensuring HSI is active and ready), enable the PLL and wait until it is ready (PLL_RDY = 1), then disable the PLL.
3. Configure the PLL to use HSE as the clock source, enable the PLL and wait until it is ready (PLL_RDY = 1).
   - If PLL_RDY = 0, the OSC_IN pin has no frequency signal and the HSE cannot be used.
   - If PLL_RDY = 1, a frequency signal is present at the OSC_IN pin, and the HSE is running and ready for use. If the PLL operation is not required, it may be disabled.

0015 I2C interface controller does not track clock stretching during START condition generation and when DIV=1-2

Status
Under investigation.

Description
The I2C interface controller does not monitor the SCL clock stretching by the slave device during the generation of a START signal when the DIV frequency prescaler is set to 1 or 2. As a result, a START signal is initiated (by setting the START and RD/WR bits in the CMD register) while the slave device is holding the SCL line low, without waiting for SCL to return to a high level, which leads to incorrect generation of a the START signal on the I2C bus (this situation may occur during the transmission of a repeated START signal).

Conditions
The internal frequency divider DIV is set to 1 or 2, the slave device is holding the SCL line low, the I2C interface controller issues a START signal (by setting the START and RD/WR bits in the CMD register).

Impact
The START signal is sent without waiting for the SCL line to return to a high level, which results in incorrect formation of the START signal on the I2C bus.

Recommendations and workarounds
If the slave device uses SCL clock stretching, introduce a delay before issuing the START signal allowing enough time for the slave device to release the SCL line, or use a frequency prescaler DIV value of 3 or higher.

0016 DMA channel stall upon a repeated ADC request during current transaction processing

Status

Under investigation.

Description

In the DMA controller, for all channels processing requests from peripheral blocks, the corresponding `dma_waitonreq` signal is set to 1 after the data transfer is completed, the DMA controller waits for the request from the peripheral block to be de-asserted.

The ADC controller generates a DMA request based on the `Flg_REG_EOCIF` flag of the `ADC1_STATUS` register. The flag is set upon completion of a conversion (the new result is written to the `ADC1_RESULT` register) and is cleared when the `ADC1_RESULT` register is read.

During sequential ADC conversions, the following sequence of events is possible:

1. Upon completion of conversion N, a DMA request is generated.
2. The DMA controller starts processing the transaction (`dma_active = 1`) and reads the `ADC1_RESULT` register; the DMA request is then de-asserted.
3. Before the transaction processing is completed (`dma_active = 1`), conversion N+1 completes, and the DMA request is asserted again.
4. The DMA controller perceives the re-asserted request as a continuation of the previous one – a new transfer is not initiated, and the controller waits for the request to be de-asserted.

Conditions

The ADC controller performs sequential conversions, and the DMA channel reads the conversion results from the `ADC1_RESULT` register upon a request from the ADC. In this case, a new conversion completes after the DMA controller reads the `ADC1_RESULT` register, but before the current transaction processing is finished. For example, this situation can occur:

− when multiple DMA channels operate simultaneously and the channel processing requests from the ADC has a low priority;

− during a lengthy re-initialization of the DMA channel control data structure.

Consequences

The re-asserted request from the ADC is perceived by the DMA controller as a continuation of the previous one – no new data transfer is performed, and the channel enters a state of waiting for the request to be de-asserted. Since the `ADC1_RESULT` register is not read, the request remains asserted, and the DMA controller stops transferring data.

Recommendations and Workarounds

The DMA transfer must be executed with minimal delay relative to the moment the ADC request is generated. To achieve this, it is recommended to:

− when multiple DMA channels operate simultaneously:

- set a high priority level for the channel processing ADC requests in the `CHNL_PRIORITY_SET` register;
- reduce the arbitration period for the channels in use via the `R_power` field of the `channel_cfg` channel settings;
- minimize the re-initialization time of the DMA channel control data structure; additionally, if the re-initialization is performed within the DMA interrupt handler, assign a high priority to this interrupt.

0017 Data presence in the Information Flash Memory (BOOT region)

Status
Fixed for ICs with date code 2624 and later.

Description
When ICs are supplied, data may be present on pages 1 and 2 of the Information Flash Memory (BOOT region) within the address range from `0x0002_1FE0` to `0x0002_27E3` inclusive.
The presence of this data does not affect the operation of the ICs, and it can be erased without any consequences.

Conditions
For ICs marked with a date code prior to 2624.

Consequences
Data cannot be written to the specified addresses without first erasing pages 1 and 2 of the Information Flash Memory region prior to the initial use.

Recommendations and Workarounds
If it is necessary to work with the specified addresses, pages 1 and 2 of the Information Flash Memory region must be erased once before the initial use.

0018 Zero FxPHASE values at low PER_LENGTH and low sampling rates

Status
Under investigation.

Description
At sampling rates of 4 kHz and 8 kHz (specific `CLC_ADC_CFG` and `OSR_CONF` settings specified below) and certain `PER_LENGTH` values (0 or 1), the `FxPHASE` registers always show 0, regardless of the actual phase difference.

Conditions
Table 1 lists the modes in which the interphase phase is always 0, regardless of the actual phase difference.
Table 1 – Configurations causing the specific operation behavior

No.	CLC_ADC_CFG	OSR_CONF	Fs, kHz	PER_LENGTH
1	0	0	4	0
2	0	0	4	1
3	0	1	8	0
4	1	1	4	0

Consequences
It is impossible to measure the phase difference under the specified settings.

Recommendations and Workarounds
Use other `PER_LENGTH` values (e.g., 2 or higher) when operating at 4 and 8 kHz, or switch to a 16kHz sampling rate.

0019 Debug JTAG shift register is not cleared upon exiting the Capture-DR state when the BYPASS register is selected

Status

Under investigation.

Description

According to IEEE 1149.1, Section 10.1.1, if the BYPASS register is selected, the shift register must be cleared to 0 in the Capture-DR state on the rising edge of TCK. Therefore, the first bit shifted out in the Shift-DR state must be 0.

In the TAP controller of the debug JTAG interface, when the BYPASS register is selected, clearing of the shift register in the Capture-DR state is not implemented. As a result, the shift register retains the state of the last loaded bit from the previous DR-Scan sequence.

Conditions

The BYPASS register is selected for the TAP controller of the debug JTAG interface, and the exit from the Capture-DR state is executed on the rising edge of TCK.

Consequences

The shift register is not cleared to 0, but instead retains the state of the last loaded bit from the previous DR-Scan sequence. Consequently, the first bit shifted out in the Shift-DR state can be 1.

Recommendations and Workarounds

If the lack of the shift register clearing when the BYPASS register is selected can affect subsequent devices in the JTAG chain, it is recommended to place the MCU at the end of the JTAG chain, or to force-load a last bit equal to 0 into the MCU TAP controller shift register before the required operations.