EtherCAT Slave Errata for PRU-ICSS-EtherCAT Firmware Version 5.4.242

Single datagram accessing multiple FMMU mapped areas using LRD/LWR commands

- **Issue/ Failure Description or state**
  - SDOCM00092510/PINDSW-47 : Single datagram accessing multiple FMMU mapped areas in a single slave will only update the data corresponding to first FMMU in datagram
- **Conditions in which failures occur**
  - Single datagram accessing multiple FMMU mapped areas in single slave.

    FMMU0(0x1000:0x1007)->SM2 (Write SM)

    FMMU1(0x1000:0x1007)->SM3 (Read SM)

    FMMU2(0x1008:0x100F)->SM4 (Write SM)

    FMMU3(0x1008:0x100F)->SM5 (Read SM)

    - Single LRD to access (0x1000:100F) will only access SM3
    - Single LWR to access (0x1000:100F) will only access SM2
    - 2 LRD/LWRs are required in this case to access both pair of SMs - LRD1/LWR1(0x1000:0x1007) and LRD2/LWR2(0x1008:0x100F)
- **Root cause**
  - Increased code memory requirements in firmware to implement this support
- **Work around**
  - For above example instead of one LRD datagram to logical address 0x1000 and length 16, master needs to send two LRD datagrams, one to logical address 0x1000 and length 8 second to logical address 0x1008 and length 8 or use LRW in place of LRD/LWR which is more efficient for most of the use cases

NOTE: This issue will not be fixed

PD/PDI watchdog counter issue

- **Issue/ Failure Description or state**
  - SDOCM00098105/PINDSW-72: PDI/PD watchdog counter incremented by 1 whenever PDI/PD watchdog is disabled using EtherCAT master
- **Conditions in which failures occur**
  - Whenever EtherCAT master disables WD by writing zero to respective Watchdog Time registers (0x410:0x411 or 0x420:0x421)
- **Root cause**
  - This is PRU-ICSS h/w behavior

NOTE: This issue will not be fixed

LRD access on unused registers

- **Issue/ Failure Description or state**
  - SDOCM00098950/PINDSW-74: LRD access on unused registers results in WKC increment
- **Conditions in which failures occur**
  - LRD access on unused registers result in WKC increment
- **Root cause**
  - Firmware does not support register protection in LRD mode at this moment, it requires more firmware footprint to support, this minor spec compliance does not justify the footprint increase and no Write Only registers in ESC

NOTE: This issue will not be fixed

LRW access to non-interleaved input and output process data of multiple slaves does not work

- **Issue/ Failure Description or state**
  - SDOCM00105048/PINDSW-141: LRW access to non-interleaved input and output process data of multiple slaves does not work. SOEM accesses slaves in LRW mode this way
- **Conditions in which failures occur**
  - Single LRW datagram accessing FMMU mapped areas in multiple slaves and PD out is mapped

    FMMU0(0x1000:0x1007)->SM2 #1(Write SM) FMMU1(0x1008:0x100F)->SM2#2 (Write SM)

    FMMU2(0x1010:0x1017)->SM3 #1 (Read SM) FMMU3(0x1018:0x101F)->SM3#2(Read SM)
    - Single LRW access from (0x1000:101F)
- **Root cause**
  - Increased code memory requirements in firmware to implement this support as well as non-interleaved access I/O data is not a very optimal use of EtherCAT – it increases the cycle time overhead/datagram size and not effective use of LRW datagram which can perform read and write in the same cycle.
- **Work around**
  - Use LRD/LWR datagram to access process data
  - Use LRW datagram to access process data
    - Input and output overlaid on the same logical address range (TwinCAT usage)
    - Input and output of a given slave back to back in logical address space
      - FMMU0(0x1000:0x1007)->SM2 #1(Write SM) FMMU1(0x1008:0x100F)->SM3#1 (Read SM)
      - FMMU2(0x1010:0x1017)->SM2 #2 (Write SM) FMMU3(0x1018:0x101F)->SM3#2 (Read SM)

NOTE: This issue will not be fixed

Dynamic register access permission change is not supported on AM335x

- **Issue/ Failure Description or state**
  - PINDSW-1005: Permissions of below mentioned registers are not updated when SYNC out unit control is switched from ECAT to PDI controlled, leading to mandatory “read-only” registers being actually “writable”
    - 0x910-0x913, 0x920-0x923, 0x930, 0x931, 0x934, 0x935, 0x981, 0x990-0x993, 0x9A0-0x9A9
- **Conditions in which failures occur**
  - When SYNC out unit control is switched from ECAT to PDI controlled.
- **Root cause**
  - The register access permissions are updated only at the start of the application and are not dynamically changed. The given issue arises when the SYNC out unit control is switched from ECAT to PDI controlled. During this change the permission of the above mentioned registers should be updated. Due to firmware memory restrictions on AM335x, the dynamic change of permissions CANNOT be implemented.

NOTE: This issue will not be fixed

Error frames with "SOF but no SFD" are not counted if arrived on reverse path port

- **Issue/ Failure Description or state**
  - PINDSW-2204 : If a frame with SOF but no SFD arrives on the reverse path, EtherCAT Slave cannot detect such frame. Therefore, it also doesn't count this frame in the error counters. According to EtherCAT specs it should be counted. These error frames are detected and counted on the forward path. The issue is only for the reverse path.
- **Conditions in which failures occur**
  - With at least two devices connected in the chain, on the device with both ports connected, if a frame without SFD arrives on the OUT port (reverse path port), then that frame is not counted in any of the error counters of the ESC
  - When the hardware forwards these frames (without SFD) it adds a SFD and a specific pattern pattern "0x5a 0x5a 0x5a 0x5a" at the end of the frame along with an odd nibble
- **Root cause**
  - Because of the way firmware handles the Ethernet frames on reverse path, there is no hardware/software provision for the firmware to even detect frames without SFD. Thus, it cannot increment the error counters accordingly.
- **Work around**
  - When such frame (without SFD, very rare phenomenon) is received by a slave, even though the receiving slave cannot detect the frame, it forwards the frame with added odd nibble. So all the following slaves will detect that frame as forwarded error.
  - Also when the slave forwards the frame, it adds a specific pattern "0x5a 0x5a 0x5a 0x5a" at the end of the frame, which can be identified using wireshark logs.

NOTE: This issue will not be fixed

System time of next Sync0 pulse register (0x990:0x993) is not updated instantaneously

- **Issue/ Failure Description or state**
  - PINDSW-2360: System time of next Sync0 pulse register (0x990:0x993) is not updated instantaneously, resulting in read of incorrect value if read immediately after sync pulse.
- **Conditions in which failures occur**
  - When Sync0 pulse register is read immediately after a sync pulse occurs.
- **Root cause**
  - The Sync0 pulse register is updated in the firmware with a typical delay of ~1.7us and worst case delay of ~5.6us. If the Sync0 pulse register is read after the sync pulse but before the register is updated (before the above limit expires) it can result into read value returning previous pulse time which is incorrect.

NOTE: This issue will not be fixed

Lost Link Counter register (0x310/0x311) increments with "2" on every link down event instead of "1"

- **Issue/ Failure Description or state**
  - PINDSW-3120 : The Lost Link Counter register (0x310 and 0x311) increments with "2" on every link down event instead of "1"
- **Conditions in which failures occur**
  - On Port0/1 Link disconnect. The Lost Link Counter register counts the link down events on the Port.
- **Root cause**
  - The reason for this issue is the MDIO link interrupt is mapped to PRU as PULSE interrupt. The Pulse nature of MDIO Link interrupt lead to PRU interrupt being set twice. Therefore, the firmware increments the counter twice for every link down event.
  - This change to Pulse mode was done to fix link stability issue in a certain rare condition after multiple link disconnect-connect cycles. The issue led to the Port always remaining closed until we have link disconnect/connect action on the other port.
- **Work around**
  - Revert back the MDIO Link interrupt to Edge in tiesc_pruss_intc_mapping.h file. Customers need to make sure that above mentioned link stability issue is not seen in their setup before making this change.

NOTE: This issue will not be fixed

Issues fixed till PRU-ICSS-EtherCAT Firmware Version 5.4.242

Circulating Bit ignored if Port 0 is set to Auto/Auto Close with no Link connected

- **Issue/ Failure Description or state**
  - PINDSW-4679 : If the Port 0 is set as Auto or Auto Close and no link is connected, then the circulating bit in the frames received on ESC is getting ignored.
- **Conditions in which failures occur**
  - Set Port 0 to Auto or Auto Close and disconnect the link
    - If a frame with circulating bit 0 is sent then the bit is not updated to 1
    - If a frame with circulating bit 1 is sent then the frame is not dropped
- **Root cause**
  - In the specific condition mentioned above the firmware was not checking the circulating bit and therefore ignoring it

Setting loop control for Port1 to Open with no link leads to stall in transmit logic in ICSSG

- **Issue/ Failure Description or state**
  - PINDSW-4678 : In SoCs with ICSSG setting loop control for Port1 to Open with no link leads to stall in transmit logic and therefore leads to ESC not responding to master
- **Conditions in which failures occur**
  - Set Loop control for Port1 to Open with no link connected.
    - Send packet to Port0. ESC tries to transmit the received packet on Port1 and leads to the issue
- **Root cause**
  - When the firmware tries to send packet on Port1 with no link connected, the ICSSG Tx FIFO goes into overflow state because the overflow detection logic in the firmware was broken

EtherCAT Slave in DC slave mode takes longer to recover from reference clock shifts

- **Issue/ Failure Description or state**
  - PINDSW-4310 : EtherCAT Slave in DC slave mode takes around 60 seconds to recover from reference clock shifts compared to ET1100 which takes 5-10 seconds
- **Conditions in which failures occur**
  - Connect two or more Ethercat slaves and configure the network in DC slave mode (Master is the DC clock master)
  - Now in this network if a sudden constant shift is added to the master clock, the slaves takes longer to settle down (around a minute) if TI slave is the first slave compared to ET1100 (few seconds)
- **Workaround**
  - Update the filter depth (0x935) for the first slave to 0xC (which is the default value for rest of the slaves). This makes the first slave more stable and resilient to jitter in the network.

2.5x more DC sync jitter with TI devices compared to ET1100 in DC slave mode

- **Issue/ Failure Description or state**
  - PINDSW-3898 : We see around 2.5x more Jitter with TwinCAT2 in DC slave mode, when we use 3 AM335x devices in chain, compared to EL9800 Board as first device and AM335x as last two devices in chain.
- **Conditions in which failures occur**
  - Connect 3 TI ESC devices in chain with TwinCAT as the master in DC slave mode and measure DC sync jitter
- **Root cause**
  - The TI ESC firmware has a fast compensation logic to improve the jitter but the same logic fails at very high jitter (typically seen in DC slave mode). The fix is to limit the use of fast compensation to the range it works best in.

Datagram access starting in middle of FMMU are incorrectly handled on AM335x

- **Issue/ Failure Description or state**
  - PINDSW-2217/2565 : A datagram which accesses data starting in middle of a FMMU, is handled incorrectly leading to invalid data write/read and incorrect WKC count with LRW frames.
- **Conditions in which failures occur**
  - Case with Large TxPDO (2Kbytes) and RxPDO (5 bytes)
    - Datagram will be divided into two frames. First frame with LRW datagram of 1473 bytes and second frame with LRW datagram of 581 bytes.
    - For first frame data is written correctly and WKC is also incremented correctly
    - For the second frame, when the data write should occur in the middle of FMMU, it happens instead to the start of FMMU. Also the read is done at start of FMMU leading to WKC increment for read as well. Therefore both data write and WKC increment is incorrect in such scenarios
- **Root cause**
  - When the Logical address is converted to physical address in firmware, the offset from the start is not taken into consideration in the conversion and the result is always the starting address of the FMMU

ESC does not insert odd nibble under certain error conditions

- **Issue/ Failure Description or state**
  - PINDSW-887/PINDSW-1552 :ESC does not insert odd nibble under certain error conditions E.g. Broken LRW frame due to > 135m cable length. This leads to subsequent slaves counting them as original error and not forwarded error

- **Conditions in which failures occur**
  - This issue is only seen when LRW frame is broken in the middle (this does not occur under normal circumstances, typically due to extremely poor line condition).
  - Issue only happens with given slave which processes corresponding logical address in FMMU where frame break occurred. Subsequent slaves insert odd nibble correctly in this case… Also slave which removes odd nibble also will count this error as forwarded error, subsequent slave (first one) will count this as original error as error counters are based on Receive and not Transmit.

- **Root cause**
  - Firmware rarely does not insert odd nibble when LRW frame is broken due to bad line condition like longer than expected cable length for given PHY

Multiple switches from INIT to OP state gives "Invalid DC timings!" warning in TwinCAT

- **Issue/ Failure Description or state**
  - PINDSW-918 : Mutiple switches from INIT to OP state on a network of 2 or more boards with Distributed clocks enabled, occasionally gives "Invalid DC timings!" warning in TwinCAT
- **Conditions in which failures occur**
  - Connect 2 or more boards in a chain to TwinCAT master with Distributed clocks enabled, and keep switching continuously from INIT to OP and back to INIT. In this setup, occasionally when the setup goes from INIT to OP it gives warning "Invalid DC timings!"
  - The issue was observed on AM3 and AM4 boards
- **Root cause**
  - There was a corner case scenario in the reverse path firmware it can occasionally get stuck in the parsing logic and not parse any of the incoming frames till it gets out of the loop causing it to not parse DC sync related frames

ESC has discrepancy in error counters on OUT port

- **Issue/ Failure Description or state**
  - PINDSW-1710: When error frames of some specific types are received on OUT port, incorrect error counters are incremented. These error are counted on IN port error counters instead of OUT port error counters
- **Conditions in which failures occur**
  - Below is the table describing error scenarios when this issue is seen

No	Error Scenario	Input frame	Output frame(ICE)	Error counters incremented (ICE)
1	Zero length frame	55 55 55 55 55 55 55 5d	55 55 55 55 55 55 55 5d 5a 5a 5a 5a 0	0x300 (bug - shall be 0x302), 0x30c
2	2 bytes length frame	55 55 55 55 55 55 55 5d 00 00	55 55 55 55 55 55 55 5d 00 00 5a 5a 0	0x300 (bug - shall be 0x302), 0x30c
3	Short preamble without SFD	55 55 55 55 55 55	55 55 55 55 55 55 55 55 55 5d 5a 5a 5a 5a 0	0x300 (bug - shall be 0x302), 0x30c
4	Long preamble without SFD	55 55 55 55 55 55 00 00 00 00 00 00 00 00 00 00 00 00 88 a4 1c 10 01 00 ff ff 00 03 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 cf 28 (68 bytes)	55 55 55 55 55 55 00 00 00 00 00 00 00 00 00 00 00 88 a4 1c 10 01 00 ff ff 00 03 10 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 cf 28 55 55 55 d5 5a 5a 5a 5a 0 (76 bytes+odd nibble)	0x300 (bug - shall be 0x302), 0x30c
5	Long preamble with SFD and odd nibble	55 55 55 55 55 55 55 55 55 5d 5a 5a 5a 5a 0	55 55 55 55 55 55 55 55 55 5d 5a 5a 5a 5a 0	0x308 (bug - shall be 0x309), 0x30c
6	Long preamble with Invalid FCS	55 55 55 55 55 55 55 55 55 5d 5a 5a 5a 5a (14 bytes)	55 55 55 55 55 55 55 55 55 5d 5a 5a 5a 5a 0 (14 bytes + odd nibble)	0x300 (bug - shall be 0x302), 0x30c

- **Root cause**
  - In the above mentioned error cases, the firmware was not mapping the ports correctly to forward/reverse path, leading to incorrect counters being incremented.

Datagram access starting in middle of FMMU are incorrectly handled

- **Issue/ Failure Description or state**
  - PINDSW-1531: A datagram which accesses data starting in middle of a FMMU, is handled incorrectly leading to invalid data write/read and incorrect WKC count with LRW frames.
- **Conditions in which failures occur**
  - Case with Large TxPDO (2Kbytes) and RxPDO (5 bytes)
    - Datagram will be divided into two frames. First frame with LRW datagram of 1473 bytes and second frame with LRW datagram of 581 bytes.
    - For first frame data is written correctly and WKC is also incremented correctly
    - For the second frame, when the data write should occur in the middle of FMMU, it happens instead to the start of FMMU. Also the read is done at start of FMMU leading to WKC increment for read as well. Therefore both data write and WKC increment is incorrect in such scenarios
- **Root cause**
  - When the Logical address is converted to physical address in firmware, the offset from the start is not taken into consideration in the conversion and the result is always the starting address of the FMMU

Writing to the edge of the Process Data RAM overwrites registers at the start of the ESC address space

- **Issue/ Failure Description or state**
  - PINDSW-929: Writing to the edge of the Process Data RAM overwrites registers at the start of the ESC address space on AM437x/AM57x devices supporting EtherCAT
- **Conditions in which failures occur**
  - Writing multiple bytes to Process Data RAM while part of the access is within the valid memory range (<= 0x7FFF) but remaining data is outside the available range. This results in overwriting of the registers near start of ESC address space (offset 0x0) due to wrap around of overflowed address.
- **Root cause**
  - The address range checks only validates the start address for write and not for the total length of write.

SYNC0 lost (0x2c) error seen occasionally when Device is powered on and off (or INIT to OP transitions)

- **Issue/ Failure Description or state**
  - PINDSW-874 / PINDSW-925 / PINDSW-926 : SYNC0 lost (0x2c) error seen occasionally when Device is powered on and off (or INIT to OP transitions) during regression tests

- **Conditions in which failures occur**
  - Issue reported here : https://e2e.ti.com/support/arm/sitara_arm/f/791/p/529633/1935811
  - In order to reproduce the issue locally, we connected 2 slaves in TwinCAT DC mode. Used configuration: TwinCAT -> AM335x ICE -> AM437x IDK
  - Then disconnected/connected AM437x power cable, after 20 tries or so following error is seen at TwinCAT log window

```
Error  'Box 2 (TIESC-002)' (1002): 'SAFEOP to OP' failed! Error: 'check device state for OP'. AL Status '0x0014' read and '0x0008' expected. AL Status Code '0x002d - no SYNC0 or SYNC1 received‘
Error 'Box 2 (TIESC-002)' (1002): state change aborted (requested 'OP', back to 'SAFEOP')
```

- **Root cause**
  - Issue root caused to corruption of PRU firmware variable tracking SYNC pulse width while activating SYNC

The register 0x0980 (mandatory writable register) is read-only

- **Issue/ Failure Description or state**
  - SDOCM00121527/SDOCM00121996/PINDSW-512 - The register 0x0980 (mandatory writable register) is read-only
- **Conditions in which failures occur**
  - EtherCAT Master performs a write (APWR, FPWR, BWR) to ESC offset 0x980
- **Root cause**
  - Register permission was set read only for 0x980 from driver side
- **Workaround**
  - In bsp_esc_reg_perm_init, add following line if not already
    - ***pRegPerm->reg_properties[0x980] = TIESC_PERM_RW;***

Executing CTT on OUT port has malformed ECAT packets

- **Issue/ Failure Description or state**
  - SDOCM00121537/PINDSW-514: Executing CTT on OUT port has malformed ECAT packets
- **Conditions in which failures occur**
  - Conformance Tests executed from OUT port (Port1) of ESC and observe the traffic using a packet sniffer
- **Root cause**
  - This unusual configuration exposed cycle budget issues in the firmware causing TX to underrun, additional cycle budget required when doing process data read using APRD command
- **Workaround**
  - NA

Circulating frame handling immediately post power reset

- **Issue/ Failure Description or state**
  - SDOCM00121961/PINDSW-556 - Circulating frame handling immediately post power reset for AM437x and AM57x IDK
- **Conditions in which failures occur**
  - Reboot the slave and send frames with circulating bit set.
- **Root cause**
  - Fixed init value of DL status Port1 to OPEN (01) than default CLOSE (00), in order to respond to the packets from Host PC towards Port1

Packet loss during state change from INIT to PREOP in redundancy

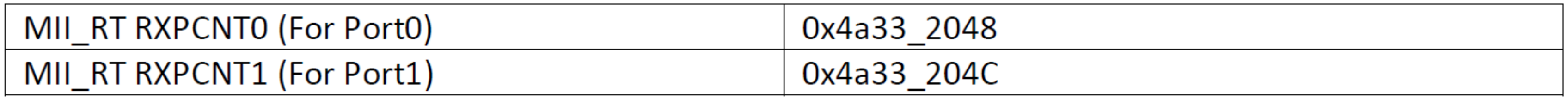
- **Issue/ Failure Description or state**
  - SDOCM00122000/PINDSW-560 - Packet loss during state change from INIT to PREOP in redundancy mode
- **Conditions in which failures occur**
  - Four slaves connected forming a redundant network in DC mode and TwinCAT3 master is used
  - A state change from INIT to PREOP has a lost frame.
- **Root cause**
  - DL control register write from master causing change in ESC Port configuration. Since DL Loop Control state m/c did not wait for ongoing frame activity @ reverse path to complete during port closure, resulting in a broken frame [depending on number of slaves and frame size]

PDO corruption issue post SDO/CoE access

- **Issue/ Failure Description or state**
  - PINDSW-681 : PDO corruption issue post SDO/CoE access
- **Conditions in which failures occur**
  - Master sends PDO and SDO one after the other. If first byte (is a status and not changed in normal circumstances) of PDO is changed at SDO in the vicinity, it is judged as error
- **Root cause**
  - This is a side effect of DC bug fix added in firmware build 0x3c0. This may cause register corruption under very rare circumstances. Depending on the value of firmware state variables above code can corrupt pointers used by PDO path. This DC bug fix was corrected in firmware build 0x3d3 onwards
- **Workaround**
  - Recommend customers using Industrial SDK 1.1.0.10/2.1.1.2 to upgrade to firmware build 0x3d7 [Industrial SDK 1.1.1.1]or 0x3e3 [PRU-ICSS-EtherCAT 01.00.00]

Frame with long preamble and SFD but valid FCS is dropped by ESC

- **Issue/ Failure Description or state**
  - SDOCM00120927/PINDSW-452: Frame with long preamble and SFD but valid FCS is dropped by ESC
- **Conditions in which failures occur**
  - EtherCAT frame with long preamble (> 7 bytes or 14 nibbles) is send to TI ESC
- **Root cause**
  - PRU-ICSS MII h/w default settings, treats long preamble as error and generates 5a 5a 5a 5a odd nibble response even when EtherCAT frame is valid with correct FCS
- **Workaround**
  - Disable long preamble error detection in PRU-ICSS MII h/w by setting 1 to below registers instead of 0xE1 (HW reset value)



MII_RT RXPCNT0 (For Port0)	0x4a33_2048
MII_RT RXPCNT1 (For Port1)	0x4a33_204C

Slaves do not go to OP state in DC mode when TI ESC is reference slave in DC Slave Mode

- **Issue/ Failure Description or state**
  - SDOCM00121174/PINDSW-477: Slaves do not go to OP state in DC mode when TI ESC is reference slave in DC slave mode (IgH or Acontis EtherCAT master)
- **Conditions in which failures occur**
  - DC Slave mode is configured at master (IgH or Acontis EtherCAT master) and TI ESC is reference slave
- **Root cause**
  - Firmware used System Time Delay (non-zero) to enable drift compensation loop at slave
- **Workaround**
  - Set non-zero System Time Delay(0x928:0x92b) value for reference slave – say 1ns

Error counter discrepancy with ET1100

- **Issue/ Failure Description or state**
  - SDOCM00121177/PINDSW-479: Error counter discrepancy with ET1100
- **Conditions in which failures occur**
  - Rx Invalid Frame Counter Port0/1 (0x300/0x302) counts EtherCAT parse errors and short frames with valid FCS
  - ECAT processing counter (0x30c) does not count following
    - Physical Layer Errors
    - Non-EtherCAT frames if 0x100.Bit0 == 1
    - Circulating bit=1 and Port0 is Automatically Closed
  - Detection of forwarded error was not possible when SFD was missing in frame due to h/w truncating frame beyond max preamble count marker (16 nibbles including SFD - SDOCM00120927)
- **Root cause**
  - Spec interpretation issue due to ambiguity in relevant sections
- **Workaround**
  - None

DC sync is not working correctly when System Time PDI controlled Time synchronization is enabled

- **Issue/ Failure Description or state**
  - SDOCM00119732/PINDSW-348: DC sync is not working correctly when System Time PDI controlled Time synchronization is enabled
- **Conditions in which failures occur**
  - When System Time PDI control is enabled (SYSTEM_TIME_PDI_CONTROLLED is set to 1 in tiescbsp.h) and Host CPU is initiating DC drift compensation control loop
- **Root cause**
  - Firmware was updating DC shadow registers when System Time PDI control is enabled, this caused inconsistent behavior when System Time PDI controlled mode was active
- **Workaround**
  - None

LATCH0/1 misses to latch positive edge sometimes in continuous mode

- **Issue/ Failure Description or state**
  - SDOCM00119422/PINDSW-332:   LATCH0/1 misses to latch positive edge sometimes in continuous mode
- **Conditions in which failures occur**
  - Latch0 is continuous mode
- **Root cause**
  - Firmware used Latch0 pin raw status to allow saving H/W latch value to register. Depending on the latch input pulse duty cycle probability of positive edge latching reduced significantly
- **Workaround**
  - Potential workaround is to increase input pulse width (duty cycle 50-50)

Inconsistency of 64-bit DC System Time

- **Issue/ Failure Description or state**
  - SDOCM00119421/PINDSW-331: Inconsistency of 64-bit DC System Time
- **Conditions in which failures occur**
  - When EtherCAT master reads DC system Time (0x910:0x917), back to back inconsistency seen w.r.t to MSW (System Time High)
- **Root cause**
  - PRU-ICSS IEP counter is 32-bit HW counter, System Time High is managed by firmware and there were race conditions in firmware around 32-bit time overflow (~ every 4.2s) handling
- **Workaround**
  - Use 32-bit DC System Time (0x910:0x913)

Port0 in Auto Close mode does not always open Port on a valid frame arrival post link up event

- **Issue/ Failure Description or state**
  - SDOCM00118530/PINDSW-299: Port0 in Auto Close mode does not always open Port on a valid frame arrival post link up event
- **Conditions in which failures occur**
  - Modify the contents of register DL Control(0x0100) from F401 to F501. Disconnect and reconnect cable (multiple times) from master to slave quickly. The slaves do not come to OP state.
- **Root cause**
  - Fixed a bug with DL link loop control state m/c, firmware opened the port on valid frame in AutoClose mode only when link change event occurred along with this
- **Workaround**
  - Configure Port0 in Auto Mode always

WKC update issues with unsupported port error counter access

- **Issue/ Failure Description or state**
  - SDOCM00118196/PINDSW-286: WKC update issues with unsupported port error counter access
- **Conditions in which failures occur**
  - EtherCAT master accessing port error counters for non-existent port incrementing WKC
- **Root cause**
  - Due to a misconfiguration of register configuration from driver to firmware, those registers were treated read only by firmware
- **Workaround**
  - Apply patch mentioned in https://cqweb.ext.ti.com/cqweb/main?command=GenerateMainFrame&service=CQ&schema=SDO-Web&contextid=SDOWP&entityID=SDOCM00118196&entityDefName=IncidentReport&username=readonly&password=readonly

Write to MI PDI Access State register (0x517) clears ESC Type (0x0000) register

- **Issue/ Failure Description or state**
  - SDOCM00118195/PINDSW-285 : Write (0, 0xFE, 0xFF) to MII Management PDI Access State register (0x517) clears ESC Type (0x0000) register
- **Conditions in which failures occur**
  - Write (0, 0xFE, 0xFF) to MII Management PDI Access State register (0x517) clears ESC Type (0x0000) register
- **Root cause**
  - TI ESC does not support switching PDI and ECAT access via ECAT interface, so this access shall be forbidden. There was a bug in the firmware which caused null pointer write during such access

Dummy traffic on link loss in a passive network when link partner is non-TLK PHY

- **Issue/ Failure Description or state**
  - SDOCM00118194/PINDSW-284: Dummy traffic on link loss in a passive network when Link partner is non-TLK PHY
- **Conditions in which failures occur**
  - In a network of slaves (even with no master connected) when Port0 link is broken, randomly activity LED of Port1 blinks. This happens when TI ESC connected to non-TLK link partner
- **Root cause**
  - Issue is a side effect of enabling Fast RXDV detection in TLK, this causes a burst of RX_ERRs to be generated during link loss as PHY detects a false RX_DV on seeing J symbol – and leads to link activity on cable break (this is normal and allowed though). This also causes periodic false RX_DV depending on link partner in a network not connected to any EtherCAT master
- **Workaround**
  - Disable Fast RX_DV detection by clearing Bit1 of SWSCR1 (0x9)

Clear UDP checksum on processing of UDP encapsulated EtherCAT packets

- **Issue/ Failure Description or state**
  - SDOCM00118024/PINDSW-275: Clear UDP checksum on processing of UDP encapsulated EtherCAT packets
- **Conditions in which failures occur**
  - When UDP encapsulated EtherCAT frames are send from PC tool with non-zero checksum, HLOS like windows, drops UDP packets with non-zero checksum when it validates and this causes an interop issue. Clearing checksum is a requirement of tunneling over UDP…
- **Root cause**
  - Firmware did not clear UDP checksum as this was not a mandatory requirement

Rx Time Port1 (0x904) time stamping not done correctly when write to 0x900 is done cyclically

- **Issue/ Failure Description or state**
  - SDOCM00108541/PINDSW-174: Rx Time Port1 (0x904) time stamping is missed when write (BWR) to 0x900 is done cyclically
- **Conditions in which failures occur**
  - When EtherCAT master cyclically writes to 0x900 register and two ports of EtherCAT slave are active
  - For first write to 0x900 after a link up event randomly
- **Root cause**
  - Parsing algorithm was not scheduled on time due to cycle budget crunch in firmware and this resulted in RX L2 FIFO overflow and parsing errors
  - Parsing algorithm and firmware optimizations were done to fix overflow
  - Parsing algorithm had a bug : which missed to detect write to 0x900 when datagram address (ado) word is split into two banks of L2 FIFO as separate bytes

ESC error counter clear behavior is not ET1100 compliant

- **Issue/ Failure Description or state**
  - SDOCM00116053/PINDSW-217: Writing a single byte in range 0x0300:0x030b does not clear complete range
- **Conditions in which failures occur**
  - ESC register range 0x300:0x30b is non-zero
  - Write to any register in this range
- **Root cause**
  - Firmware supports write clear per byte for error counters

PDI ISR on h/w pin is not generated for ICEv2

- **Issue/ Failure Description or state**
  - SDOCM00117382/PINDSW-250: PDI ISR on h/w pin is not generated for ICEv2 even though PDI ISR DIGIO pin selection register (0xE0A) is not set to 0xFF
- **Conditions in which failures occur**
  - If PDI ISR over DIGIO pin is enabled and pinmux for this pin is configured correctly to select pr1_edio_data_outN function
- **Root cause**
  - Due to a bug in driver, PRU-ICSS DIGIO SW dataout mode is not enabled. Register Offset PRU_IEP_DIGIO_CTRL_REG was used instead of PRU_IEP_DIGIO_EXP_REG

SM0/1 events may trigger PDI ISR even when they are masked

- **Issue/ Failure Description or state**
  - SDOCM00117503/PINDSW-253 : SM0/1 (mailbox) events may trigger PDI ISR even when they are masked
- **Conditions in which failures occur**
  - It has been reported that when TwinCAT does CoE access in OP mode, ARM sees more SM2 events than number of actual cyclic data events.
- **Root cause**
  - This is root caused to a firmware issue, in the scenarios when ARM clears PDI ISR late (E.g.:- SM2 event) and other AL events are raised (E.g.:- mailbox event) even though masked. Firmware re-triggers PDI ISR when previous SM2 event (or any enabled event at AL event mask register) is pending in AL event IRQ register and any of disabled event (at AL event mask register) occurs meanwhile Fixed firmware to use only newly generated enabled event to trigger interrupt to host
- **Work around**
  - Clearing the AL event IRQ events as early as possible

SM2 first byte data corruption is seen randomly

- **Issue/ Failure Description or state**
  - SDOCM00117692/PINDSW-259 : SM2 first byte data corruption is seen randomly. This can also lead to ESC type register corruption
- **Conditions in which failures occur**
  - When PDI side mailbox read or write occurs and ECAT is reading the same buffer or just switched to corrupted buffer
- **Root cause**
  - Firmware used register R11 as pointer to ECAT and AL event request registers. This is corrupted (but unnoticed as broadside access XIN [register file load from scratch register bank] was made when PDI side mailbox read or write occurs) and post PDI operation, firmware updates AL/ECAT event request register. XIN loads SM start address pointers to register file and this explains random corruption of first few bytes of SM2

SYNC0 jitter increases with distance of slave from reference

- **Issue/ Failure Description or state**
  - SDOCM00097012/PINDSW-62 : SYNC0 jitter increases with distance of slave from reference
- **Conditions in which failures occur**
  - TI ESC in a large network of slaves - shows variable jitter which increases with distance of the slave from reference slave
- **Root cause**
  - Drift compensation algorithm improvements are made, speed counter filter depth increased

Frame corrupted when LRW issued with logical address ending just before FMMU logical start address

- **Issue/ Failure Description or state**
  - SDOCM00115000/PINDSW-213: Frame corrupted when LRW issued with logical address ending just before FMMU logical address
- **Conditions in which failures occur**
  - FMMU start address is X
  - LRW datagram last byte accessed at X-1
- **Root cause**
  - Fixed the firmware bug with logical address boundary check

Process data corruption when PDI update is slow (slave running non-real time OS)

- **Issue/ Failure Description or state**
  - SDOCM00115132/PINDSW-214: Process data corruption when PDI update is slow (slave running non-real time OS)
- **Conditions in which failures occur**
  - Large process data (say 256 bytes) exchanged between slave and master
  - PDI process data write is delayed more than one cycle time
  - PLC sees corrupted Process Data Input
- **Root cause**
  - Fixed a bug in the firmware with triple buffer pointer handling

CTT TF1100.2.5 Mailbox Read Service Repeat 2 regression failure

- **Issue/ Failure Description or state**
  - SDOCM00115487/PINDSW-215 : CTT TF1100.2.5 Mailbox Read Service Repeat 2 regression failure
- **Conditions in which failures occur**
  - When above CTT test is repeated in regression mode (>= 10000 times) sporadic failure is seen
- **Root cause**
  - There was an offset change for t_host_interface.sm_config_ongoing variable in firmware due to merge issue which was not migrated to host interface, so stack/application missed the SM busy status
- **Work around**
  - Update t_host_interface struct in tiescbsp.h as shown below

```
typedef struct {
   Uint8 reserved1[0x90];
   Uint32 system_time_low;
   Uint32 system_time_high;
-  Uint8 sm_config_ongoing;
-  Uint8 reserved2[7];
+  Uint16 reserved;
+  Uint8 sm_config_ongoing;
+  Uint8 reserved2[5];
```

NOTE: Work around is not needed >= 1.1.0.8

Power on Reset DC filter setting (System Time Filter depth of 4) causes PRU firmware lockup when DC is enabled

- **Issue/ Failure Description or state**
  - SDOCM00116155/PINDSW-218: Power on Reset DC filter setting (System Time Filter depth of 4) causes PRU firmware lockup when DC is enabled
- **Conditions in which failures occur**
  - Reported by IgH master which did not configure System Time Filter depth and enabled DC mode of operation
  - Results in random PRU lock up making device unresponsive to state change and error reset requests from master
- **Root cause**
  - Fixed the firmware to avoid this lock up scenario for all supported filter depth
  - Recommended filter depths are Speed Counter Filter Depth (12) and System Time Filter Depth (0)

PortX in Auto-close mode opens port incorrectly on Link up event

- **Issue/ Failure Description or state**
  - SDOCM00116714/PINDSW-222: PortX in Auto-close mode opens port incorrectly on Link up event
- **Conditions in which failures occur**
  - PortX in AutoClose mode, Port can be opened only if a valid frame received over PortX or Write to DL Control (0x100) with “01” via Other port and not on Link Up event
  - PortX in AutoClose mode opens on Link Up event
- **Root cause**
  - Firmware state m/c bug is fixed

In low cycle times with high-jitter masters, ECAT API bsp_get_process_data_address fails sporadically

- **Issue/ Failure Description or state**
  - SDOCM00113900/PINDSW-200: In low cycle times with high-jitter masters, ECAT API bsp_get_process_data_address in file tiescbsp.c fails sporadically
- **Conditions in which failures occur**
  - When running TwinCAT master on laptop or PC with onboard NIC, this is observed that bsp_get_process_data_address returns 0
- **Root cause**
  - Fixed a bug with updating lock_state (incorrectly) - only address meant to be updated by firmware

When PDI-side register permissions are enabled, Link/Activity LED indications are reversed

- **Issue/ Failure Description or state**
  - SDOCM00114419/PINDSW-209: When PDI-side register permissions are enabled, Link/Activity LED indications are reversed
- **Conditions in which failures occur**
  - ENABLE_PDI_REG_PERMISSIONS is enabled in tiescbsp.h while building the ethercat application and observe Link and activity LED behavior are reversed while running the app in ICE EVM
- **Root cause**
  - The address of PRU MII RX LINK polarity was changed to 0x0E0C in ISDK 1.1.0.5. PDI register permission array did not reflect this, this causes LED task controlling the activity to behave incorrectly