EtherCAT : Errata for Industrial SDK 2.1.0.1

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

- **Issue/ Failure Description or state**
  - SDOCM00092510 : 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: 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 be not be fixed

LRD access on unused registers

- **Issue/ Failure Description or state**
  - SDOCM00098950: 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: 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

Issues fixed in 2.1.0.1

NOTE: 2.1.0.1 EtherCAT firmware for AM335x (firmware/v1.0 sub folder) is compatible with Industrial SDK 1.1.0.8

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

- **Issue/ Failure Description or state**
  - SDOCM00108541: 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 write to 0x900 register and two ports of EtherCAT slave is active or randomly for first write 0x900 after a link up event
- **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: 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 the 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: 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 : 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 : 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 update 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

Issues fixed in 1.1.0.8

SYNC0 jitter increases with distance of slave from reference

- **Issue/ Failure Description or state**
  - SDOCM00097012 : SYNC0 jitter increases with distance of slave from reference
- **Conditions in which failures occur**
  - TI ESC in a large network of slaves and away from reference slave shows variable jitter which increases with distance of the slave
- **Root cause**
  - Drift compensation algorithm in firmware requires optimization

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

- **Issue/ Failure Description or state**
  - SDOCM00115000: 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: 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 : 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: 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: 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

Issues Fixed in 1.1.0.6

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

- **Issue/ Failure Description or state**
  - SDOCM00113900: 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) when 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: 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