EtherCAT : Errata for Industrial SDK 1.1.0.1

For application developers: Known Issues/Limitations

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

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

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

NOTE: This issue will be addressed in subsequent releases

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

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

Bytewise LRD does not work correctly for last byte in the datagram for odd byte length > 1

- Issue/ Failure Description or state
  - SDOCM00103985: Bytewise LRD does not work correctly for last byte in the datagram for odd byte length > 1
- Conditions in which failures occur
  - When LRD access of size 3,5,7,2n+1 bytes is made to slave from master, last byte is corrupted
- Root cause
  - Due to a firmware bug bitmask was incorrectly applied on transmit data from slave

NOTE: This issue is fixed in 1.1.0.3 SDK

DC SYNC start time programmed earlier than current system time causes slave to hang

- **Issue/ Failure Description or state**
  - SDOCM00103985: DC SYNC start time programmed earlier than current system time causes slave to hang
- **Conditions in which failures occur**
  - Include bsp_get_latch0_posedge_time calls in task1
  - Configure slave for DC Synchron
  - From Device menu on TwinCAT, do a Online Delete and Online Reset of slave
  - Slave temporarily hangs, and this happens repeatedly
  - Even the LINK/ACT LEDs stop blinking to indicate packets
- **Root cause**
  - Error was triggered by a master bug (DC SYNC start time programmed earlier than current system time) - but firmware was not handling the error scenario correctly. Fixed the same

NOTE: This issue is fixed in 1.1.0.3 SDK

Overnight regression on CTT #33 fails very rarely and slave rejects further state transitions

- **Issue/ Failure Description or state**
  - SDOCM00104218: Overnight regression on CTT #33 fails very rarely and slave rejects further state transitions
- **Conditions in which failures occur**
  - Run large number of iterations of CTT test #33 (SafeOp > Init ErrFlag = 0, SmSett0 or SmSett1 not match)
  - Slave hangs and subsequent CTT tests fail in the error state
- **Root cause**
  - Fixed a corner case in TI ESC HAL driver when a timeout occurs spinlock is not freed causing a firmware lockup (very rarely)

NOTE: This issue is fixed in 1.1.0.3 SDK

PDO write from master to slave has a turnaround time of 1.3 * bus cycle time

- **Issue/ Failure Description or state**
  - SDOCM00104727 : PDO write from master to slave has a turnaround time of 1.3 * bus cycle time (0.3 * bus cycle time is expected)
- **Conditions in which failures occur**
  - Send master clock time as process data output
  - Compare on slave, time difference between the master write time and time when the slave application receives it
  - Send the difference back to master as process data input
  - 0.3 times Bus cycle time (expected), 1.3 times Bus cycle time (seen)
- **Root cause**
  - Fixed a bug with buffer exchange which prevented most recent buffer received from ECAT master to be available for PDI

NOTE: This issue is fixed in 1.1.0.3 SDK

Multiple FMMU access to non-SM mapped memory in a single LRW datagram with read only FMMUs causes missed frames

- **Issue/ Failure Description or state**
  - SDOCM00105027: Multiple FMMU access to non-SM mapped memory in a single LRW datagram with read only FMMUs causes missed frames
- **Conditions in which failures occur**
  - Send an LRW in each cycle using "Append EtherCAT Cmd“ option in TwinCAT
  - Configure multiple FMMUs with read only access that will come within logical address + size of LRW
  - Activation of FMMU causes missed frames
- **Root cause**
  - Multiple FMMU access to non-SM mapped memory in a single LRW datagram with read only FMMUs is not handled in firmware and fixed the same

NOTE: This issue is fixed in 1.1.0.3 SDK

SYNC1 not generated for SYNC1 cycle not equal to SYNC0 cycle

- **Issue/ Failure Description or state**
  - SDOCM00105261: SYNC1 not generated for SYNC1 cycle not equal to SYNC0 cycle
- **Conditions in which failures occur**
  - Configure PINMUX so that pr1_edc_sync1_out is mapped expansion header and observe the signal using a scope
  - Enable SYNC1 and configure SYNC1 cycle time = N*SYNC0 cycle time (N=1,2,3,..) using EtherCAT master and
- **Root cause**
  - ICSS SYNC unit register not initialized correctly when SYNC1 cycle time is not equal to SYNC0 cycle time - resulted in SYNC1 not being triggered

NOTE: This issue is fixed in 1.1.0.3 SDK

ESC type register (0x0) does not read 0x90 (TI), read 0x02 instead

- **Issue/ Failure Description or state**
  - SDOCM00105262: ESC type register (0x0) does not read 0x90 (TI), read 0x02 instead
- **Conditions in which failures occur**
  - Read ESC Type register after packet transmission
  - This causes DC interop issue with certain masters (which uses ESC Type to configure DC mode) due to DC incorrectly configured for 32-bit mode
- **Root cause**
  - Fixed a firmware bug which resulted in ESC register @ 0x0 byte0 being overwritten during packet transmission

NOTE: This issue is fixed in 1.1.0.3 SDK

Could not read process data input of length equal to 1 from PDI side

- **Issue/ Failure Description or state**
  - SDOCM00105263: Could not read process data input of length equal to 1 from PDI side
- **Conditions in which failures occur**
  - Send master clock time as process data output
- **Root cause**
  - Fixed process data handling in TI ESC HAL layer for single byte SM as SM is disabled by firmware. Also fixed a corruption of output bytes till last byte when process data output bytes are greater than process data input bytes in LRW datagram

NOTE: This issue is fixed in 1.1.0.3 SDK

For stack integrators: Known Issues/Limitations