EtherCAT : Errata for Industrial SDK 1.1.0.6

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

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
- Root cause
  - Firmware monitors writes to 0x900, misses to update Port1 time stamp under increased traffic conditions

NOTE: Fix is under evaluation

For stack integrators: Known Issues/Limitations

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: This issue will be 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