Anybus CompactCom 40 EtherCAT

Application note EtherCAT sync error counters

Version history

Version	History	Author	Date
1.00	Initial version	JoG	2018-03-26
1.01	Updated chapter 1.3. Added chapter 1.4 and 1.5.	MaE	2020-01-13
1.02	Minor changes after review	JoG	2020-05-29

About this document

This document describes how to implement the synchronization error counters used for DC synchronization diagnostics on EtherCAT.

The Anybus CompactCom 40 EtherCAT is referred to as “module” throughout this document.

Trademarks

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.

Referenced documents

Short	Title	Number	Author	Ver
[ABCC40SDG]	Anybus CompactCom 40 Software Design Guide	HMSI-216-125	HMS Industrial Networks AB	4.0
[ETG1020]	EtherCAT® Protocol Enhancements	ETG.1020 S (R) V1.2.0	EtherCAT® Technology Group	V1.2.0
[ABCC40NWG]	Anybus CompactCom 40 EtherCAT Network Guide	SCM-1202-034	HMS Industrial Networks AB	2.3

Table of Contents

1 EtherCAT synchronization error counters

1.1 Background

To make it easier to diagnose issues with DC synchronization on EtherCAT there are some synchronization error counters and an error flag available to be read by the master.

These are located on the sync manager parameter objects (0x1C32 and 0x1C33).

Version 2.1.25 and later of the EtherCAT Conformance Test Tool will check that these synchronization error counters are present, and if they are not the conformance test will fail.

For more information about SYNC, see [**ABCC40SDG**].

1.2 Sync manager parameter objects

The sync manager parameter objects contain configuration, information and diagnostics for synchronization behavior on EtherCAT. They are specified in [**ETG1020**] and are used by the Anybus CompactCom 40 EtherCAT according to Table 1.

The entries marked orange in Table 1 above are mandatory to support when DC synchronization is used. They are however not (except for sub-index 12, Cycle Time Too Small) implemented by the module. This is because synchronization monitoring is a highly application specific task. The module cannot know the specific requirements of the host application and is therefore unable to implement these counters.

Instead the error counters and sync error flag must be implemented by the host application.

Object index	Sub-index	Name	Data type	Access	Description
0x1C32 Output SyncManager Parameter	0	Highest sub-index supported	UNSIGNED8	R	
	1	Synchronization Type	UNSIGNED16	R or RW	Currently active operation mode. The module will indicate which operation mode the master has enabled. Free run (0x00) or DC Sync0 (0x02).
	2	Cycle Time	UNSIGED32	R or RW	The cycle time configured by the EtherCAT master. This entry is not used by the module in Free run mode and shows the Sync0 Cycle Time (Register 0x9A3-0x9A0) in ns when using DC Sync0.
	3	Shift Time	UNSIGNED32	R or RW	The value written here by the master will be forwarded to the “Output valid” attribute on the Sync host object and is used by the host application to shift the output valid point relative to the sync event.
	4	Synchronization Types supported	UNSIGNED16	R	The module will set bit 0 (Free Run supported) if attribute “Supported sync modes” on the Sync host object isn’t implemented. Otherwise this entry will set bit 0 if bit 0 is set in “Supported sync modes” and set bit 2 if bit 1 is set in “Supported sync modes”.
	5	Minimum Cycle Time	UNSIGNED32	R	Shows the minimum cycle time supported by the device. Reflects attribute “Min cycle time” on the Sync host object.

Object index	Sub-index	Name	Data type	Access	Description
	6	Calc and Copy Time	UNSIGNED32	R	Time needed by the application controller to copy the process data from the Sync Manager to the local memory and perform calculations if necessary before the data is sent to the process. Reflects attribute “Output processing” on the Sync host object.
	7	Minimum Delay Time	UNSIGNED32	R	Not supported by the module.
	8	Get Cycle Time	UNSIGNED16	RW	Not supported by the module.
	9	Delay Time	UNSIGNED32	R	Always set to 0 in the module.
	10	Sync0 Cycle Time	UNSIGNED32	RW	Not supported by the module.
	11	SM-Event Missed	UNSIGNED16	R	Not supported by the module but must be supported when DC synchronization is used.
	12	Cycle Time Too Small	UNSIGNED16	R	Always set to 0 in the module.
	13	Shift Time Too Short	UNSIGNED16	R	Not supported by the module but must be supported when DC synchronization is used.
	14	RxPDO Toggle Failed	UNSIGNED16	R	Not supported by the module.
	15	Minimum Cycle Distance	UNSIGNED32	R	Not supported by the module.
	16	Maximum Cycle Distance	UNSIGNED32	R	Not supported by the module.
	17	Minimum SM SYNC Distance	UNSIGNED32	R	Not supported by the module.
	18	Maximum SM SYNC Distance	UNSIGNED32	R	Not supported by the module.
	19-31	-	-	-	Reserved for future use
	32	Sync Error	BOOL	R	Not supported by the module but must be supported when DC synchronization is used.
0x1C33 Input SyncManager Parameter	0	Highest sub-index supported	UNSIGNED8	R	
	1	Synchronization Type	UNSIGNED16	R or RW	Same value as 0x1C32:01.
	2	Cycle Time	UNSIGED32	R or RW	Same value as 0x1C32:02.
	3	Shift Time	UNSIGNED32	R or RW	The value written here by the master will be forwarded to the “Input capture” attribute on the Sync host object and is used by the host application to shift the input capture point relative to the sync event.
	4	Synchronization Types supported	UNSIGNED16	R	Same value as 0x1C32:04.
	5	Minimum Cycle Time	UNSIGNED32	R	Same value as 0x1C32:05.
	6	Calc and Copy Time	UNSIGNED32	R	Time in ns needed by the application controller to perform calculations on the input values if necessary and to copy the process data from the local memory to the Sync Manager before the data is available for EtherCAT. Reflects attribute “Input processing” on the Sync host object.

Object index	Sub-index	Name	Data type	Access	Description
	7	Minimum Delay Time	UNSIGNED32	R	Not supported by the module.
	8	Get Cycle Time	UNSIGNED16	RW	Not supported by the module.
	9	Delay Time	UNSIGNED32	R	Always set to 0 in the module.
	10	Sync0 Cycle Time	UNSIGNED32	RW	Not supported by the module.
	11	SM-Event Missed	UNSIGNED16	R	Not supported by the module.
	12	Cycle Time Too Small	UNSIGNED16	R	Always set to 0 in the module.
	13	Shift Time Too Short	UNSIGNED16	R	Not supported by the module.
	14	RxPDO Toggle Failed	UNSIGNED16	R	Not supported by the module.
	15	Minimum Cycle Distance	UNSIGNED32	R	Not supported by the module.
	16	Maximum Cycle Distance	UNSIGNED32	R	Not supported by the module.
	17	Minimum SM SYNC Distance	UNSIGNED32	R	Not supported by the module.
	18	Maximum SM SYNC Distance	UNSIGNED32	R	Not supported by the module.
	19-31	-	-	-	Reserved for future use
	32	Sync Error	BOOL	R	Not supported by the module.

Table 1 Sync manager parameter objects

1.3 Implementation in CompactCom Host Application Example Code

The SYNC event to the application can be trigged by the SYNC pin on the application interface (MI0/SYNC) or in the ABCC interrupt register (SYNCI). This implementation uses SYNC event through ABCC interrupt register.

Before creating this example make sure that correct operating mode against ABCC module is configured. Read more about operating modes in abcc_cfg.h and how to configure in abcc_drv_cfg.h.

1.3.1 Creating the error counters

The CompactCom Host Application Example Code has been prepared with functionality for implementing the necessary EtherCAT sync error counters.

Enabling this is done in following steps:

1. Enable **ECT_OBJ_ENABLE** in abcc_obj_cfg.h. This will enable the EtherCAT host object.

   ```
   #define ECT_OBJ_ENABLE TRUE
   ```

2. Enable **SYNC_OBJ_ENABLE** in abcc_obj_cfg.h. This will be done by setting **ABCC_CFG_SYNC_ENABLE** to TRUE in abcc_drv_cfg.h.

   When **ABCC_CFG_SYNC_ENABLE** is set to TRUE, **ABCC_CFG_USE_ABCC_SYNC_SIGNAL** in abcc_drv_cfg.h will also be TRUE. **ABCC_CFG_USE_ABCC_SYNC_SIGNAL** must be set to FALSE in this example. In this example the ABCC interrupt will be used for indicating SYNC event.

   ```
   #define SYNC_OBJ_ENABLE ABCC_CFG_SYNC_ENABLE
   #define ABCC_CFG_SYNC_ENABLE TRUE
   #define ABCC_CFG_USE_ABCC_SYNC_SIGNAL FALSE
   ```

3. Enable **ABCC_CFG_INT_ENABLED** in abcc_drv_cfg.h to enable the ABCC interrupt.

   ```
   #define ABCC_CFG_INT_ENABLED TRUE
   ```

4. Set **APPL_ACTIVE_ADI_SETUP** to APPL_ADI_SETUP_SYNC in appl_adi_config.h.
   Read more about the different ADI_SETUP in appl_adi_config.h.

   ```
   #define APPL_ACTIVE_ADI_SETUP APPL_ADI_SETUP_SYNC
   ```

5. Enable **ECT_SUPPORT_SYNC_ERR_COUNTERS** in abcc_obj_cfg.h

   ```
   #define ECT_SUPPORT_SYNC_ERR_COUNTERS TRUE
   ```

   This will automatically enable the “Object sub-index translation” attribute in the EtherCAT host object and define the necessary information needed to implement the mandatory object entries. **ECT_IA_OBJ_SUB_TRANS_ENABLE** will be set to TRUE when **ECT_SUPPORT_SYNC_ERR_COUNTERS** is set to TRUE.

   Following sub-indices, 11-13 and 32 from object 0x1C32 and 12 from object 0x1C33 are added to **APPL_asAdiEntryList** in appl_adimap_sync.c when **ECT_SUPPORT_SYNC_ERR_COUNTERS** is set to TRUE.

   ```
   ECT_SUB_TRANS_1C32_11, "SM-Event Missed"
   ECT_SUB_TRANS_1C32_12, "Cycle Time Too Small"
   ECT_SUB_TRANS_1C32_13, "Shift Time Too Short"
   ECT_SUB_TRANS_1C32_32, "Sync Error"
   ECT_SUB_TRANS_1C33_12, "Cycle Time Too Small"
   ```

   This is necessary in the application for support the “Object sub-index translation” attribute in the EtherCAT host object.

6. **ABCC_CFG_STRUCT_DATA_TYPE** and **ABCC_CFG_ADI_GET_SET_CALLBACK** must be set to FALSE in this implementation. This is done in abcc_drv_cfg.h.

```
#define ABCC_CFG_STRUCT_DATA_TYPE FALSE
#define ABCC_CFG_ADI_GET_SET_CALLBACK FALSE
```

7. In this application EoE is turned off. Disable EoE in EtherCAT host object. Set **ECT_IA_ENABLE_EOE_ENABLE** to TRUE and **ECT_IA_ENABLE_EOE_VALUE** to FALSE in abcc_obj_cfg.h.

```
#define ECT_IA_ENABLE_EOE_ENABLE TRUE
#define ECT_IA_ENABLE_EOE_VALUE FALSE
```

   For more information about EoE see [**ABCC40NWG**].

The Host Application Sample Code can now be built, and when the tested together with an Anybus CompactCom 40 EtherCAT module the mandatory synchronization error counters will be accessible.

So far, the object entries only exist with static values. The sync error counter behavior should also be implemented.

1.3.2 Implementing error counter behavior

1.3.2.1 0x1C32:11 SM-Event Missed error counter

This error counter is defined the following way in [**ETG1020**]:

This error counter is incremented when the application expects a SM event but does not receive it in time and as consequence the data cannot be copied any more.
used in DC Mode

Basically, what this means is that the SM-Event Missed error counter should be incremented by 1 whenever there are two sync events without an update of Rx (Read) process data in between.

Implementing this in the CompactCom Host Application Example Code could be done by incrementing the ADI corresponding to the SM-Event Missed error counter whenever **`APPL_SyncIsr()`** is called twice without **`ABCC_CbfNewReadPd()`** being called in between.

The corresponding ADI could be set to 0 on every successful transition to the PROCESS_ACTIVE state.

1.3.2.2 0x1C32:12 Cycle Time Too Small

This error counter is defined the following way in [**ETG1020**]:

This error counter is incremented when the cycle time is too small so that the local cycle cannot be completed and input data cannot be provided before the next SM event.
used in Synchronous or DC Mode

This means that the Cycle Time Too Small error counter should be incremented whenever RxPDO (Read process) data is delivered too often, so the application doesn't have time to perform the necessary copying and calculations of the process data.

Implementing this in the CompactCom Host Application Example Code could be done by incrementing the ADI corresponding to the Cycle Time Too Small error counter whenever **`ABCC_CbfNewReadPd()`** is called too often so the host application doesn't have time to process the delivered data.

The corresponding ADI could be set to 0 on every successful transition to the PROCESS_ACTIVE state.

Important note: In Anybus CompactCom EtherCAT versions 2.09 and lower, the sub-indices on 0x1C32 implemented by the module has higher priority than the ones implemented by the host application. This means that on these versions it is not possible for the host application to affect the value on 0x1C32:12, it will always be 0.

1.3.2.3 0x1C32:13 Shift Time Too Short

This error counter is defined the following way in [**ETG1020**]:

This error counter is incremented when the time distance between the trigger (Sync0) and the Outputs Valid is too short because of a too short Shift Time or Sync1 Cycle Time.
used in DC Mode

This means that the Shift Time Too Short error counter should be incremented each time the sync event occurs too close to the "Outputs Valid" point (the point when outputs should be actualized). If the shift time isn't used (meaning Sync Event = Outputs Valid) then this error counter is kept at 0 at all times.

Implementing this in the CompactCom Host Application Example Code could be done by incrementing the ADI corresponding to the Shift Time Too Short error counter whenever a sync event occurs and the value of the "Output Valid" attribute on the Sync host object is lower than the lowest supported value.

The corresponding ADI could be set to 0 on every successful transition to the PROCESS_ACTIVE state.

1.3.2.4 0x1C32:32 Sync Error

This entry is defined the following way in [**ETG1020**]:

Shall be supported if SM-Event Missed or Shift Time Too Short Counter is supported
Mappable in TxPDO
0: no Synchronization Error or Sync Error not supported 1: Synchronization Error

This flag should be set to TRUE whenever the ADI corresponding to either SM-Event Missed or Shift Time Too Short error counters are larger than 0. If both error counters are 0 the flag should be cleared.

1.3.2.5 0x1C33:12 Cycle Time Too Small

This error counter is defined the following way in [**ETG1020**]:

Same value as 0x1C32:0C
used in SM Synchronous or DC Mode

This means that the ADI corresponding to this Cycle Time Too Small error counter could reference the same variable as the ADI corresponding to 0x1C32:12. See section 1.3.2.2.

Important note: In Anybus CompactCom 40 EtherCAT versions 2.09 and lower, the sub-indices on 0x1C33 implemented by the module has higher priority than the ones implemented by the host application. This means that on these versions it is not possible for the host application to affect the value on 0x1C33:12, it will always be 0.

1.4 Object 0x101F: Error Settings

This object is used for setting the error reaction behavior of the EtherCAT slave. If the slave supports several error reactions this object shall be supported, otherwise this object is optional. See [**ETG1020**] for object description.

1.4.1 Implementation of object

This object is not implemented in ABCC 40 EtherCAT module, so if it should be supported, it needs to be implemented in the host application. Implementation could be done by using ADI translation attribute in the EtherCAT host object. See [**ABCC40NWG**] for information about ADI translation attribute in EtherCAT host object.

1. Enable `ECT_IA_ADI_TRANS_ENABLE` by setting it to TRUE in abcc_obj_cfg.h.

   ```
   #define ECT_IA_ADI_TRANS_ENABLE TRUE
   ```

2. Add object index to `ECT_IA_ADI_TRANS_VALUE`, in this case 0x10F1. In this case we use 0xF0F1 as ADI instance number.

   ```
   #define ECT_IA_ADI_TRANS_VALUE { { 0xF0F1, 0x10F1 } }
   ```

3. Set `ECT_IA_ADI_TRANS_SIZE` to number of added values in `ECT_IA_ADI_TRANS_VALUE`. In this case 1, since only 1 ADI was added.

   ```
   #define ECT_IA_ADI_TRANS_SIZE 1
   ```

4. Object code for 0x10F1 is RECORD, an array that contain different datatypes. To be able to add this object to APPL_asAdiEntryList in appl_adimap_sync.c `ABCC_CFG_STRUCT_DATA_TYPE` must be set to TRUE in abcc_drv_cfg.h.

   ```
   #define ABCC_CFG_STRUCT_DATA_TYPE TRUE
   ```

5. Create a struct for object 0x10F1 containing sub index for object 0x10F1. See abcc_ad_if.h for how to create a struct for an object when `ABCC_CFG_STRUCT_DATA_TYPE` is set to TRUE.

6. Add an ADI for object 0x10F1 to `APPL_asAdiEntryList[]` in appl_adimap_sync.h. See abcc_ad_if.h how to add an ADI to asADIEntryList[] when the ADI is a struct.

7. If this object should be added to the project, the project needs to be modified to build correctly once `ABCC_CFG_STRUCT_DATA_TYPE` is set to TRUE.

1.5 Host Application Status Register

The host application status is primarily used in SYNC applications. It is used in applications that has the ability to indicate process data errors to the master. If the application sets an error code to the application status register the EtherCAT state and Anybus state will be affected. The value that is written to the register will be translated to an ALStatusCode and sent on the EtherCAT network.

The AppStatusType is defined in abp.h

See Appendix B.3 Application Status Register in [**ABCC40NWG**] for more details how the EtherCAT and Anybus state is affected and how the register value is translated to ALStatusCode.