

Many EtherCAT systems benefit greatly from a tight synchronization of devices running on the network. Synchronization is particularly important when drives are being controlled in cyclic synchronous position mode. In this mode the master performs all trajectory calculations and passes points to the drives in real time over the network. The drives will interpolate between the passed points and servo to the resulting position. Tight synchronization is required between the master and slave devices to maintain good servo performance.

This application note will outline some of the features of Copley EtherCAT drives which support synchronization over the EtherCAT network.

### ***Distributed clocks***

The distributed clock is the primary mechanism built into the EtherCAT network protocol to allow synchronization between the master and slaves on the network. Not every EtherCAT device supports the distributed clock protocol, but those that do can use this mechanism to share a common clock domain across the network. All Copley EtherCAT drives support the distributed clock.

When the distributed clock protocol is being used, one clock on the network is selected as the master clock, and all other devices are synchronized to it. The master controller of the network determines which clock will be used as the master clock. The master clock can either reside in the master controller itself, or in one of the slave devices on the network. In many systems the slave devices are able to capture time stamps more accurately than the master controller, so frequently the first slave device in the network chain is selected as the source of the master clock.

Every EtherCAT slave device which supports the distributed clock includes hardware which allows a very accurate local time stamp to be captured when certain registers are written over the network. These time stamps can then be used by the slave device to adjust it's local clock to remove the drift between it and the master clock on the network. The EtherCAT master uses these time stamps to calculate the network delay between devices on the network and to find an offset between each slave's local time and the system time. Once this offset has been found for each slave, the master writes the offset to a register on the slave's EtherCAT interface hardware. The result is a shared time base for every device on the network which supports the distributed clock protocol.

## ***SYNC0 pulse***

The distributed clock allows multiple devices on the network to share a common time reference, but doesn't itself provide any real functional synchronization. Additional hardware is provided on each DC enabled slave device which allows a pulse to be generated on the slave at a fixed period. This pulse, known as the SYNC0 pulse, can be used by the slave device to synchronize it's internal functions to the network.

The master is responsible for configuring the SYNC0 pulse on each slave. Typically, the master finds a SYNC period which is compatible with all slave devices, and configures the SYNC0 signal on all devices to occur simultaneously.

The acceptable SYNC periods for each slave device can be found in the documentation provided by the device manufacturer. Copley drives run an internal position loop with an update rate of 4kHz (250 microseconds). For best performance, it's recommended that the SYNC0 period used with Copley drives be an integer multiple of the 250 microsecond position loop update rate. Any integer multiple of the position loop period can be used up to a maximum period of 10ms.

Once the SYNC0 signal is configured by the master to a multiple of the drive's servo period, the drive will adjust it's internal loop to align the start of a servo period with the SYNC0 signal. Since the master typically configures the SYNC0 signals of multiple drives on the network to occur simultaneously, the result is simultaneous servo updates on multiple drives.

If for some reason it is not possible to use a SYNC0 period that is a multiple of 250 microseconds (for example, due to limitations of other devices on the network), Copley drives can also synchronize using a SYNC0 period which is a multiple of the drive's 62.5 microsecond current loop update rate. This is less desirable then using a period which is a multiple of 250 microseconds because it only allows the current loops on the drives to be synchronized. If a SYNC0 period which is a multiple of 62.5 microseconds, but not a multiple of 250 microseconds is used on a Copley drive, then the drive's current loop will be synchronized to the SYNC0 signal, but the position loop will not be.

## ***Debugging synchronization***

The distributed clock and SYNC0 signals are all generated internal to the slave devices on the network. This can make it difficult to debug and verify the correct operation of the system synchronization mechanisms. Copley drives provide some useful diagnostic capabilities that can aid the system developer in this area.

One extremely useful tool for debugging synchronization issues is to program a general purpose drive output pin to generate a rising or falling edge when the SYNC0 signal occurs on the drive. Using an oscilloscope, the SYNC0 signals of multiple drives can thereby be viewed directly. In a correctly configured system the SYNC0 signals of all drives should occur simultaneously with no drift between them.

The ability to configure an output pin to pulse on a SYNC0 signal is a feature available in all 2<sup>nd</sup> generation Copley EtherCAT drives starting with version 1.24 firmware. As of this writing the CME

user interface does not directly support this output pin configuration setting, however it is still possible to enable this feature either through the ASCII command line interface in CME, or by directly writing to the output pin configuration object over EtherCAT.

Each general purpose output pin on a Copley drive has a configuration register associated with it. When accessing these configuration registers over the drive's serial interface (including the ASCII command line in CME2), the configuration register numbers are 0x70 for output 1, 0x71 for output 2, 0x72 for output 3, etc. To configure an output pin to reflect the SYNC0 signal, the output configuration register should be set to a value of 11. For example, to configure output pin 2 (register 0x71) to output a SYNC0 pulse, the following ASCII command would be used:

```
s r0x71 11
```

That command will set the configuration in ram which causes the configuration to be immediately active, but lost on reset. To set the configuration of output 1 (register 0x70) in flash, the following command line could be used:

```
s f0x70 11
```

That configuration will become active after any drive power cycle or reset. Please refer to the ASCII command line documentation for more complete details on setting output pin configurations.

The output pin can also be configured over the EtherCAT interface. Object 0x2193 is an array type object with one sub-index for each general purpose output pin. For example, object 0x2193 sub-index 3 is used to configure output pin 3. The EtherCAT master can use a CoE mailbox access to set these configuration registers to a value of 11 to enable SYNC0 output mode. When configuring outputs through the EtherCAT interface in this way, the output is always configured in RAM and the new configuration will be immediately enabled and lost on reset.

In addition to configuring an output pin to track the SYNC0 signal from the EtherCAT core, it is also possible to configure an output pin to generate a pulse at the start of the drive's 250 microsecond servo period. To enable this configuration, the output pin configuration register should be set to the value 512 (0x200). For example, to configure output 3 in this mode the following ASCII command could be used:

```
s r0x72 0x200
```

If the drive's servo period is correctly synchronized to the SYNC0 signal, then the active edges of both output should occur simultaneously.

One other useful feature for monitoring the network synchronization is the drive's network status register. This register is available at serial port parameter number 0x102 and CoE object 0x21B4. The value stored in this read-only register is bit-mapped as follows:

| <b>Bit</b> | <b>Description</b>  |
|------------|---|
| 0          | Set if the EtherCAT master has configured SYNC0 to a legal period   |
| 1          | Set if the drive's servo period is synchronized to the SYNC0 signal |
| 2          | Set if the SYNC0 period is an integer multiple of 250 microseconds. |

- 3 Set if the master incorrectly set the SYNC0 start time less than the system time.
- 4-15 reserved.

## ***Sync manger configuration***

Copley drives support two different synchronization modes for their process data sync managers. These modes are; free running, and SYNC0 synchronization.

Object 0x1C32 sub-index 1 is used to configure the synchronization mode for the process data outputs from the drive back to the master. When this object is set to 0 (default) the output sync manager will be configured for free running mode. In this mode, the drive will update the output process data available for the master to read on every servo update. When the master sends an EtherCAT packet to read the process data from the drive, the data it will receive will be from the most recent servo cycle.

When 0x1C32.1 is set to a value of 2, the output sync manager will be placed in SYNC0 synchronization mode. In this mode the drive will sample the output data at the time of the SYNC0 pulse, and update the buffers used to transmit this data to the master shortly after that time. The master can then read this data back and know what the state of the drive was at the SYNC0 time.

Object 0x1C33.1 is used to configure the input sync manager mode. It also supports 0 (free running, default), and 2 (SYNC0 synchronization).

When 0x1C33.1 is set to 0, the drive will immediately use any input data transmitted by the master as soon as it is received. In this mode, the SYNC0 signal is not used to latch the incoming data.

When 0x1C33.1 is set to 2, the drive will only start using incoming data when a SYNC0 signal is received. This allows the master to transmit the data intended to be used at SYNC0 time early, and the drive will not use the data until the SYNC0 signal occurs.

In a typical EtherCAT system the master will periodically send process data to all devices on the network. Ideally, this process data will be received by the slave devices with a fixed delay relative to the SYNC0 signal. For example, the master may configure the SYNC0 period on all slaves to 1 millisecond, and time it's communications so that the slaves receive updated process data every millisecond, exactly 50 microseconds before the SYNC0 signal occurs.

It's very common in an EtherCAT system for the master to run on a complex PC operating system, and therefore not have the high degree of real time performance that the slaves possess. In such cases there can be a significant amount of timing jitter on the process data messages that the master sends. For example, if the master has +/- 100 microseconds of jitter on it's message transmission timing, then the slave may receive the process data update anywhere from 150 microseconds before SYNC0 to 50 microseconds after SYNC0. This can cause system level problems such as incorrect trajectory interpolation in cyclic synchronous position mode.

Configuring the process data sync managers to use SYNC0 synchronization mode can resolve the problems caused by timing jitter in the master. In this mode the master can compensate for it's worst case timing jitter by transmitting the process data to the slaves sufficiently early to ensure that the data will be received before the SYNC0 signal. The slaves will not use the process data received until the

SYNC0 time, so system can remain well synchronized even with a significant amount of timing jitter in the master.

For example, in a system with a cycle time of 1ms and +/-100 microseconds of timing jitter on the master, the master could be configured to transmit its process data with a 500 microsecond offset (half the cycle time) from the SYNC0 time on the slaves. This would ensure that the slave devices received the process data well clear of the SYNC0 update. Since the slaves are configured in SYNC0 synchronization mode, they will not use the updated process data until the SYNC0 signal occurs.

### ***Running without synchronization***

Most commercially available EtherCAT masters have the distributed clock feature built in which makes it very easy to configure a system with tight synchronization. If however a custom EtherCAT stack is being designed for a particular application, then the development of the distributed clock in the master stack can be a significant effort. In this case it may be possible to run without the use of the distributed clock although certain modes of operation should be avoided.

The most common form of motion control used on an EtherCAT network is known as Cyclic Synchronous Position (or CSP) mode. In this mode the master calculates the trajectory points and sends them down to the drives over the network with no buffering on the drives. For high quality servo performance, the drive should receive the trajectory points from the master with a fixed delay from the start of the drive's servo period. Without the use of the distributed clock there is really no way to ensure this fixed delay, resulting in reduced quality of motion. CSP mode should be avoided on systems which can not use the distributed clock.

Profile position mode is an alternative mode of operation which is much less timing sensitive. In this mode the master sends the trajectory limits (max velocity, etc) and the end point of the move. The master then writes to a control word on the drive to start the move, and the drive performs all trajectory calculations internally. In this mode a lack of tight synchronization between the master and slave devices has very little consequence. This is an ideal mode of operation for simple systems which need to run on a non-realtime operating system, or which do not have access to the distributed clock.

One more mode of operation which is useful on non-synchronized systems is interpolated position mode. This mode is similar to CSP mode in that the master performs the trajectory calculations, but in this mode the drive buffers the trajectory segments internally and pulls points from the buffer. The drive will interpolate between points using either linear or cubic polynomial interpolation. The use of the buffer allows for much looser synchronization between the master and drive without reduced quality of motion.

## Revision History

| Date       | Version | Revision        |
|------------|---------|-----------------|
| 10/03/2011 | 1.0     | Initial release |
| 3/06/2012  | 1.1     | Changed logo    |