copley controls

AN102
v1.1

I/O extension features in Copley modules

Copley's newest line of modular drive products support a general purpose serial interface which can be used to interface to external electronics. This feature is supported on all modular drives in the *plus* product family. The following drives support this feature starting with firmware version 1.54:

AEM, APM, SEM, SPM, AE2, AP2, SE2, SP2.

Overview

Modular drives are designed to be mounted on a customer's PC board. The PC board will generally contain other electronics specific to the customer's system requirements. The I/O extension feature allows the CANopen or EtherCAT master to interface to this I/O indirectly through the Copley module. The network master sends data to the module which in turn acts as a master on the SPI bus to control the external hardware.

The SPI bus interface is implemented in the drive using several of it's general purpose input and output lines. Specifically, three outputs are used (clock, data and chip select) and one input is used (data). The following I/O pins are used on each module supporting the feature:

Drive	Data out	Clock	Chip Select	Data in
AEM, APM, SEM, SPM	OUT3	OUT4	OUT5	IN10
AE2, AP2, SE2, SP2	OUT4	OUT5	OUT6	IN18

The I/O extension feature can be configured in one of three general ways.

- Disabled. In this case the drive I/O are available for normal functions such as limit switches, home switches, etc.
- I/O extension. Defined below, the I/O act as an SPI master interfacing to external hardware.
- SLI mode. In this mode the I/O interface to simple shift registers to drive CAN/EtherCAT status LEDs and address switches. This is the mode used on Copley development kits.

The I/O extension feature is configured using several drive parameters:

Serial port parameter	**CANopen / Ecat Object**	**Description**
0x15A	0x2198	I/O options register
0x18C	0x21A1	I/O extension configuration register.
0x18D	0x21A2	Transmit data. Writing to this register will initiate a new transfer.
0x18E	0x21A3	Receive data. After a transfer, the data received over the SPI bus can be read from this location.

The I/O options register is used to configure various drive options related to the general purpose I/O of the drive. For the module products, the following two bits are related to the I/O extension feature:

Bits	**Description**
0	Set to enable the I/O extension function. Clear to use the I/O as normal general purpose I/O
1	When using the I/O in SLI mode, this bit determines whether the CANopen/EtherCAT status LED is treated as a single bi-color status LED, or whether it's two different LEDs (one red error LED, one green state LED).

This parameter is typically configured by CME during initial drive setup.

The I/O extension configuration register is mapped as follows. Undocumented bits should be written as zero, and ignored on reads.

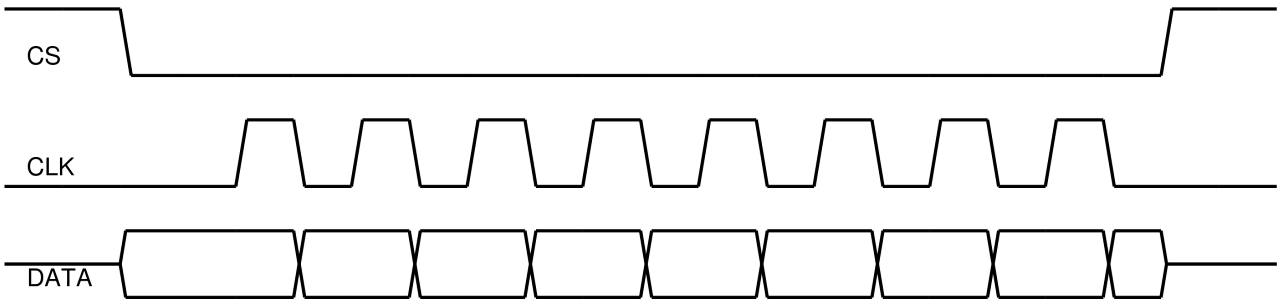
Bits	Description
0-7	Number of bits to transfer. The value written here should be 1 less then the number of bits to transfer in a single cycle. For example, write 31 to transfer 32 bits.
9	If set, the drive will automatically restart the transfer every loop cycle. If clear, a new transfer is only started when the transmit data register is written.
10	If set, the chip select line will be left low after the transfer. This is useful if more then 32 bits must be transferred without raising the CS line.
11	Status bit which is set at the end of a transfer when the receive data register is updated. This bit is automatically cleared when the received data register is read.
12	Clock polarity. 0 = clock is low between transfers. 1 = clock is high between transfers.
13	Data phase. 0 = sample on the leading clock edge. 1 = sample on the trailing clock edge.
16-23	Clock period in 100ns units. The SPI clock period will be (N*100+20) ns if this value is set to *N*.
28	Enable SPI mode. Must be set to 1 for SPI mode. If clear, SLI mode will be used.

The transmit and receive data parameters (0x18D, 0x18E) are of variable length depending on how many bits of data the drive is configured to transfer. When transmit data is written to parameter 0x18D, up to 16 words (256 bits) of data can be written. When receive data is read back from parameter 0x18E, the number of words returned will depending on the number of bits of data that the module is configured to transfer. For example, if the module is configured to transfer between 17 and 32 bits of data, two words will be returned on a read.

When accessing the transmit/receive data buffers using the CANopen / EtherCAT objects, the number of bytes of data read/written will be dependent on the configuration. This is the same as writing to the drive parameters over the serial port, the only difference is that the CANopen objects are written/read in units of bytes rather then 16-bit words. For example, if the module is configured to transfer 24 bits of data, then 3 bytes would be read from object 0x21A3.

Clock polarity & data phase

There is some variation in the handling of the SPI bus by slave devices. Some devices require the clock to start a transfer high, some require it to start low. Some sample their data on the rising edge of the clock, some on the falling edge. The clock polarity and data phase bits in the configuration word allow the drive to interface to any of these devices.



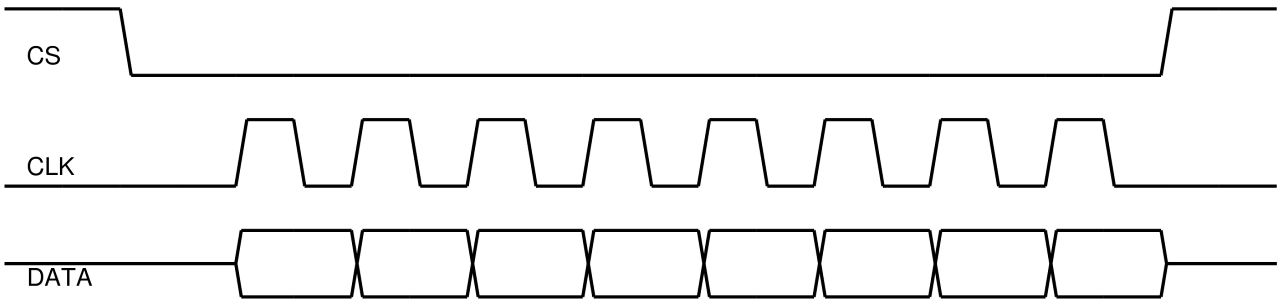
Clock polarity = 0, Data phase = 0

The waveform above gives an example of an SPI transfer where both clock polarity and data phase are set to zero.

Before the transfer starts, the chip select line is high, the clock line is low, and the data lines don't hold any useful data. Typically SPI peripheral chips will tristate the data line when the chip select is high (as shown above). The Copley drive can not tristate it's general purpose output pin, so it will be held at an undefined value.

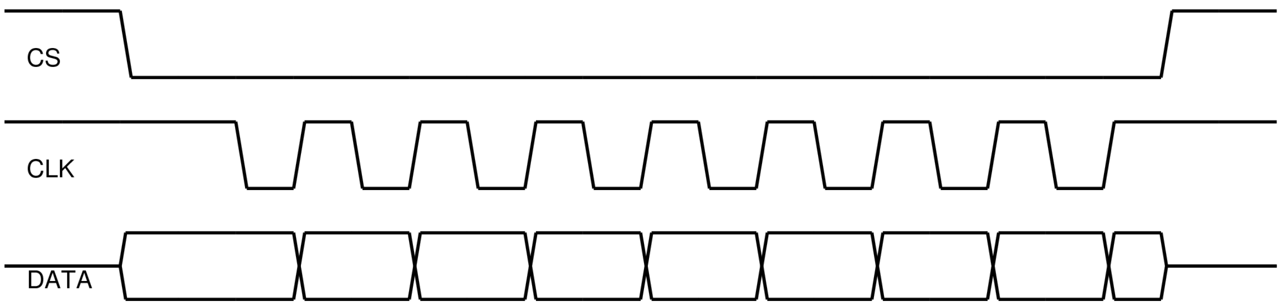
When the transfer starts, the chip select line falls. At this time the Copley module and remote device should both drive the first bit of data onto their respective data pins.

On each rising clock edge, both devices sample their data input pins. On the falling clock edges both devices update the value of their data outputs for the next bit of the transfer.



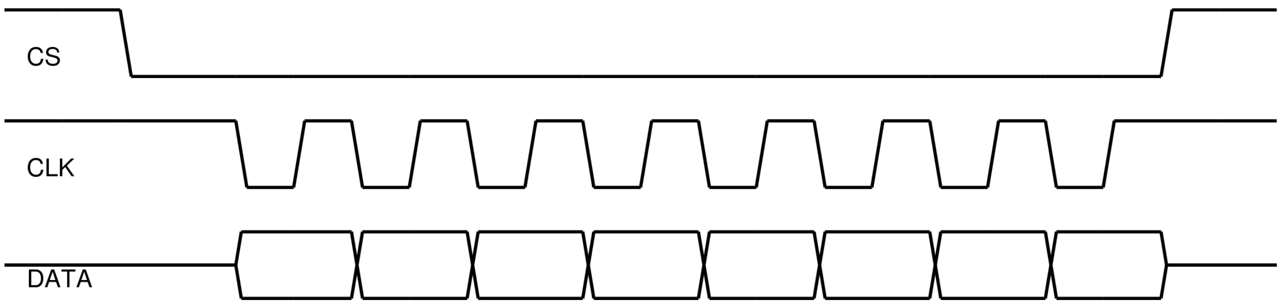
Clock polarity = 0, Data phase = 1

The image above gives an example transfer in which the data phase is set to 1. In this case, valid data is output on the first edge of each bit (the rising clock edge here, since clock polarity is 0). Data is sampled by each device on the falling clock edge.



Clock polarity = 1, Data phase = 0

When clock polarity is set to 1, the clock is normally high between transfers. In the above image the data phase is set to zero, so the data is immediately driven at the start of transfer and sampled on the first clock edge (a falling edge here). The next bit is output on the rising edge of the clock.



Clock polarity = 1, Data phase = 1

When clock polarity is 1 and data phase is 1, the data is updated on the falling edge of the clock and sampled on the rising edge of the clock.

Initiating data transfer

To initiate a data transfer over the SPI port, the configuration register should first be initialized to the correct settings based on the remote hardware requirements. Once the port has been configured, a new transfer is started by writing to the transmit register.

Every servo period (typically 250 microseconds) the drive firmware checks to see if new data has been written to the transmit register. If so, it will initiate a new SPI bus transfer. The data written to the transmit register will be clocked out and the response from the remote hardware will be clocked in. Once all bits have been transferred, the returned data will be stored in the receive register.

Bit 11 of the configuration register is used as a status bit to indicate that the most recent transfer has finished and new data is available in the receive register. This status bit is automatically cleared when the receive data register is read. This status bit can optionally be used to determine when the SPI bus transfer has completed and new receive data is available.

Automatic transfer mode

In some cases it's convenient to constantly transmit the same value to the remote hardware. In this case bit 9 of the configuration register can be used. When this bit is set, the drive will constantly resend the value most recently written to the transmit register every servo cycle. After completing each transfer, the receive register value will be updated with the value read from the remote hardware.

Revision History

Date	Version	Revision
8/6/2012	1.0	Initial release
1/31/2014	1.1	Fixed an error in the description of the SPI clock period setting.