TechNews: EtherCAT Distributed Clocks

DC Handling mit dem Sybera EtherCAT Master

Prinzip der Distributed Clocks

Bei EtherCAT wird zur Synchronisation der Slaves auf das Prinzip der verteilten Uhren, nachfolgend Distributed Clocks (DC), gesetzt. Hierfür ist es nötig, dass zunächst jeder Slave seine eigene lokale Zeit durch einen eigenen Taktgeber hat. Der erste DC-fähige Slave im Strang wird überlicherweise als Reference Clock gesetzt. Alle anderen Slaves synchronisieren sich von nun an auf diese Reference Clock. Hierfür sendet der Master regelmäßig ein Telegramm, in das der Slave welcher als Reference Clock dient seine Uhrzeit einträgt und alle anderen Slaves diese Uhrzeit auslesen.

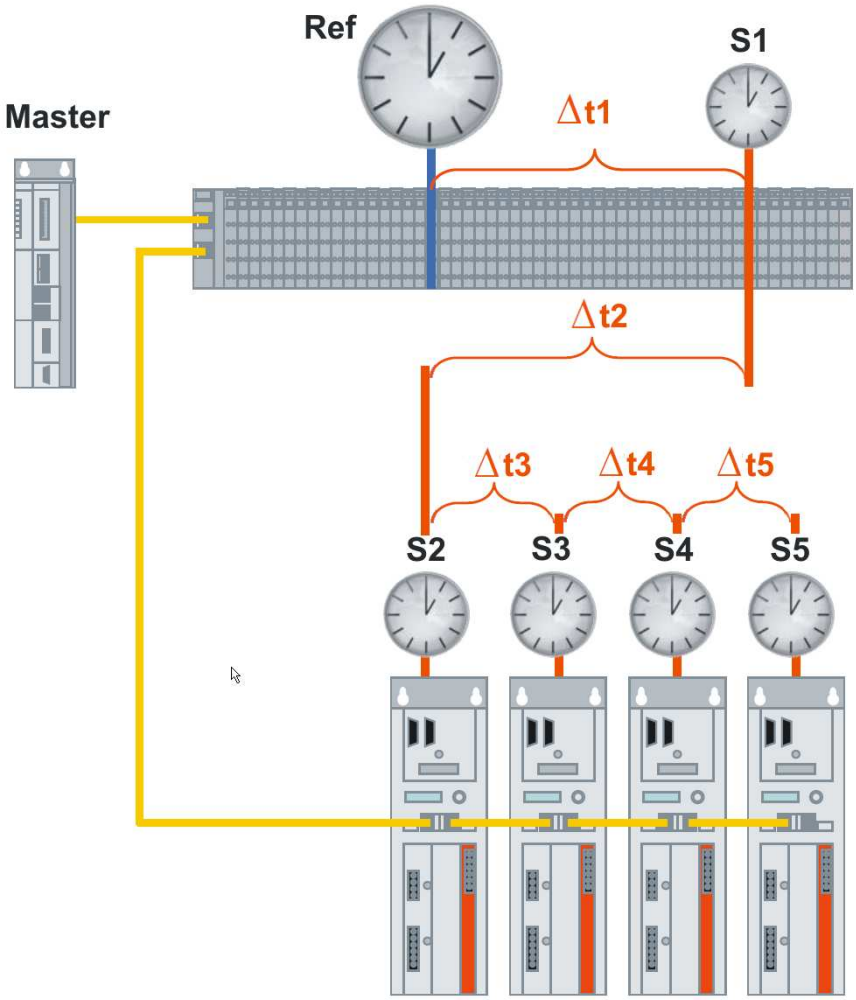

Bild: © EtherCAT Technology Group
Text: Author Martin Ritz, Embedded Communication, emtas GmbH

SYBERA GmbH, Geschäftsführer J.Rall, Hohenzollernstr.2, 71088 Holzgerlingen, Tel: 07031-744-608 (Fax: -609), Email: info@sybera.de

TechNews: EtherCAT Distributed Clocks

DC Handling mit dem Sybera EtherCAT Master

Laufzeitbestimmungen

Sowohl die Slaves, als auch die Übertragungsstrecke sorgen für eine geringe Verzögerung des Telegramms auf dem Bus. Für eine exakte Synchronisation ist es nötig, dass diese Verzögerung bestimmt und bei der Synchronisierung berücksichtigt werden muss. Zur Bestimmung dieses Offsets sendet der Master in der Startphase ein bestimmtes Telegramm in das jeder Slave auf dem Hin- und auf dem Rückweg den Empfangszeitpunkt hineinschreibt. Hieraus kann der Master die Verzögerungen zwischen den einzelnen Slaves exakt bestimmen. Neben der Reference Clock existiert im Netzwerk noch die Master Clock, welche den absoluten Bezug zur Realität darstellt. Diese kann zur Nachregelung der Reference Clock beim Systemstart, aber auch im Betrieb genutzt werden. Somit werden folgende Effekte eliminiert:

- Unterschiede zwischen den lokalen Zeiten der Slaves zur Reference Clock
- Unterschiede von der Reference Clock zur Master Clock
- Laufzeitunterschiede in Abhängigkeit von den Teilnehmer und dem Übertragungsmedium
- Driften der einzelnen Slave Clocks durch kontinuierliche Driftkorrektur

DC Intern

Die DC (Distributed Clocks) Unit ist ein Hardware-Baustein innerhalb des ESC (EtherCAT Slave Controller), der die lokalen Uhren auf eine gemeinsam geteilte Systemzeit synchronisiert. Die Synchronisation der lokalen Uhren ist die Voraussetzung für den erfolgreichen Betrieb der SYNC-Signale und / oder LATCH-Signale, und damit für einen korrekten Betrieb der Synchronisationsmodi der Slave-Applikation.

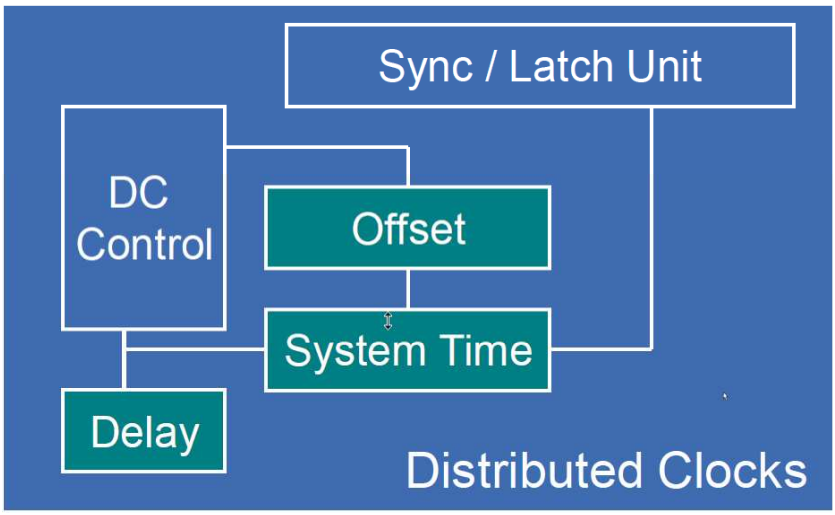

Bild: © EtherCAT Technology Group
Text: Author Martin Ritz, Embedded Communication, emtas GmbH

SYBERA GmbH, Geschäftsführer J.Rall, Hohenzollernstr.2, 71088 Holzgerlingen, Tel: 07031-744-608 (Fax: -609), Email: info@sybera.de

TechNews: EtherCAT Distributed Clocks

DC Handling mit dem Sybera EtherCAT Master

Nutzung der Distributed Clocks

In der Regel verfügt der EtherCAT Slave Controller über zwei zeitgesteuerte Interrupts, SYNC0 und SYNC1. Somit ist es möglich auf eine vom Anwender vorgegebene Zeit einmalig zu reagieren oder zyklische Aktionen auszulösen. Solche Aktionen können zum Beispiel das gezielte Schreiben der Ausgangsdaten oder das Einlesen der Eingangsdaten sein. Somit würde sichergestellt, dass die Eingangsdaten aller Slaves vom gleichen Zeitpunkt stammen oder, dass die Ausgänge aller Slaves synchron gesetzt werden. Die Distributed Clocks (DC) Unit EtherCAT Slave-Controller unterstützen die folgenden Funktionen:

- Taktsynchronisation zwischen den Slaves (und dem Master)
- Erzeugung von synchronen Ausgangssignalen (SyncSignals)
- Präzise Zeitstempelung von Eingangsereignissen (LatchSignals)
- Erzeugung von synchronen Interrupts
- Synchrone digitale Output-Updates
- Synchrone digitale Input-Abtastung

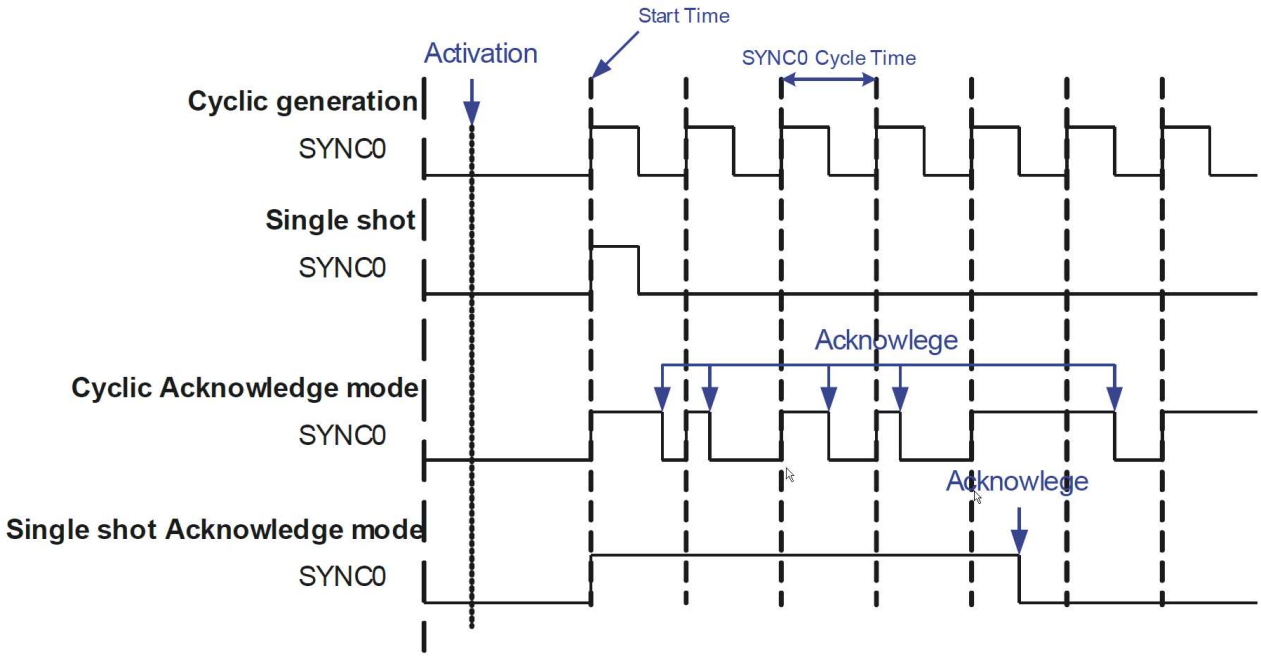

Bild: © EtherCAT Technology Group
Text: Author Martin Ritz, Embedded Communication, emtas GmbH

SYBERA GmbH, Geschäftsführer J.Rall, Hohenzollernstr.2, 71088 Holzgerlingen, Tel: 07031-744-608 (Fax: -609), Email: info@sybera.de

TechNews: EtherCAT Distributed Clocks

DC Handling mit dem Sybera EtherCAT Master

Code Beispiel:

Folgendes Code-Beispiel zeigt die Distributed Clock Synchronisierung und Aktivierung mit dem Sybera EtherCAT Master. Zu beachten ist, dass die Synchronisierung direkt nach dem State-Wechsel zu PRE-OPERATIONAL erfolgen muß, da hier die zyklische Einmessung (ARMW-Frames) beginnt.

```
__inline void __SyncControl(
                        ULONG Period,
                        ULONG SyncCycles)
{
        //Loop through all stations
        for (int i=0; i<__StationNum; i++)
        {
                //Activate sync unit
                Ecat64SyncControl(
                        &__pUserList[i],
                        Period * SyncCycles * 1000,   //Sync0 cycle time [nsec]
                        0,                            //Sync1 cycle time [nsec]
                        20*1000,                      //Sync0 cycle shift [nsec]
                        0,                            //Sync1 cycle shift [nsec]
                        TRUE,                         //Sync0 pulse flag
                        FALSE,                        //Sync1 pulse flag
                        FALSE);                       //Sync PDI control
        }
}

...

        //Change state to PRE OPERATIONAL
        if (ERROR_SUCCESS == Ecat64ChangeAllStates(AL_STATE_PRE_OP))
        {
                ULONG DriftTimePerMsec = 0;

                //Init DC immediately after cyclic operation has started
                //and get static master drift per msec (nsec unit)
                if (ERROR_SUCCESS == Ecat64ReadDcLocalTime())
                if (ERROR_SUCCESS == Ecat64CompDcOffset())
                if (ERROR_SUCCESS == Ecat64CompDcPropDelay())
                if (ERROR_SUCCESS == Ecat64CompDcDrift(&DriftTimePerMsec))
                {
                        //Check DC quality
                        ULONG MaxSysTimeDiff = 0;
                        Ecat64CheckDcQuality(&MaxSysTimeDiff);

                        //Activate DC sync unit
                        __SyncControl(REALTIME_PERIOD, SYNC_CYCLES);
```