MOPGF027

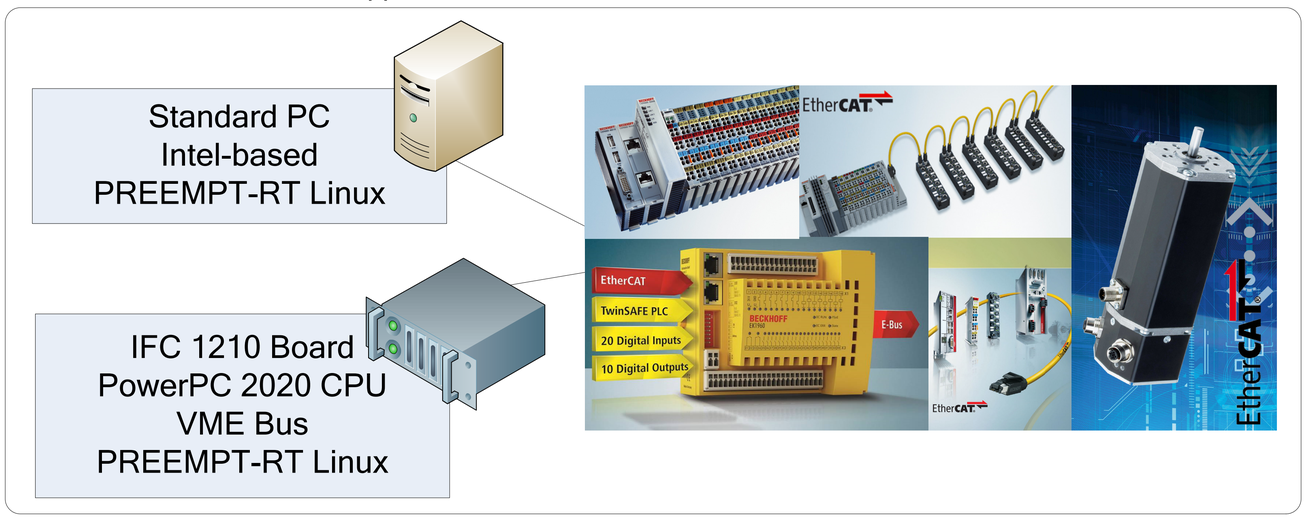
Real-time EtherCAT Driver for EPICS and Embedded Linux at Paul Scherrer Institute (PSI)

Dragutin Maier-Manojlovic, Paul Scherrer Institute (PSI), Villigen, Switzerland

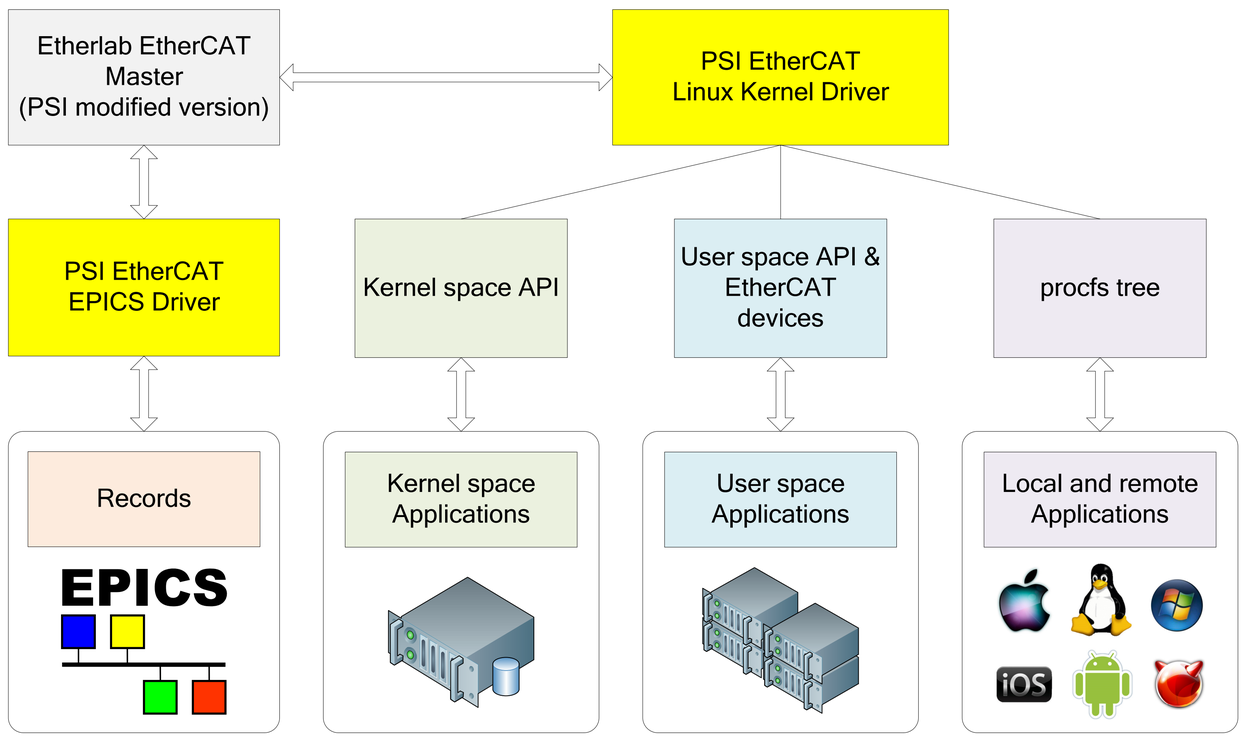
Abstract

EtherCAT bus and interface are widely used for external module and device control in accelerator environments at PSI, ranging from undulator communication, over basic I/O control to Machine Protection System for the new SwissFEL accelerator. A new combined EPICS/Linux driver has been developed at PSI, to allow for simple and mostly automatic setup of various EtherCAT configurations. The new driver is capable of automatic scanning of the existing device and module layout, followed by self-configuration and finally autonomous operation of the EtherCAT bus real-time loop. If additional configuration is needed, the driver offers both user- and kernel-space APIs, as well as the command line interface for fast configuration or reading/writing the module entries. The EtherCAT modules and their data objects (entries) are completely exposed by the driver, with each entry corresponding to a virtual file in the Linux procfs file system. This way, any user application can read or write the EtherCAT entries in a simple manner, even without using any of the supplied APIs. Finally, the driver offers EPICS interface with automatic template generation from the scanned EtherCAT configuration.

Overview - PSI EtherCAT support can be used on standard PCs or IFC 1210 Boards



Structure - PSI EtherCATdrivers support EPICS, kernel-, userspace and remote applications



Usability - Access EtherCAT data from any OS/application, local or remote

PSI EtherCAT Linux kernel driver provides an easy way to access and configure the EtherCAT data, using **procfs trees** created on-the-fly. Each procfs tree includes entries representing EtherCAT Modules, SMs, PDOs and PDO entries, as they appear in the user-defined EtherCAT Domain.

By accessing the "files" in the procfs tree, representing the different levels of each EtherCAT Module, it is possible to **read or write the current value(s)** of each entry or even multiple entries at once, both from applications or using any CLI..

Aside from the entries themselves, assorted related data is available to applications or CLI scripts through the PSI EtherCAT procfs trees, such as:

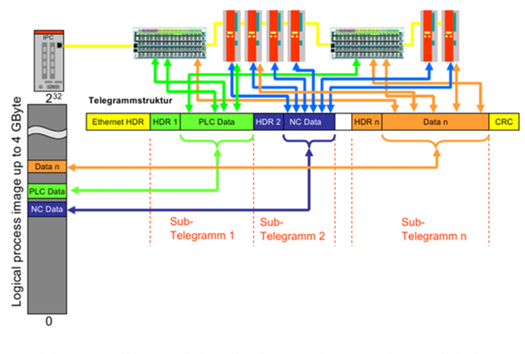
- concise **maps of the domains**, with entries and their values, optionally with maps of EPICS records connected (and their values)
- **rate/frequency**, to read and change the scan rate for domain on-the-fly
- **lists, states and counts** of domains, entries, and other objects
- **slave-to-slave communication** transaction maps and current values

Easy setup - Automatic scanning or manual configuration, or a mix of both!

Both PSI EtherCAT and Linux driver support **automatic scan and configuration of EtherCAT Modules**. There is no need to prepare XML files or perform any other setup in advance.

Modules can also be configured **statically**, either from EPICS or any other application or from the CLI. The static configuration will work even if the modules are not reporting their own configuration (SII missing or wrong).

Entries can also be added or removed using **on-the-fly-configuration**, if needed.



Versatility - Multi-domain and multi-freq

Each Domain can contain arbitrary number of PDO Entries, and **each has its own, real-time scan rate**, varying from multiple kHz, down to sub-Hz ranges.

Scan for different data at different rates/frequencies!

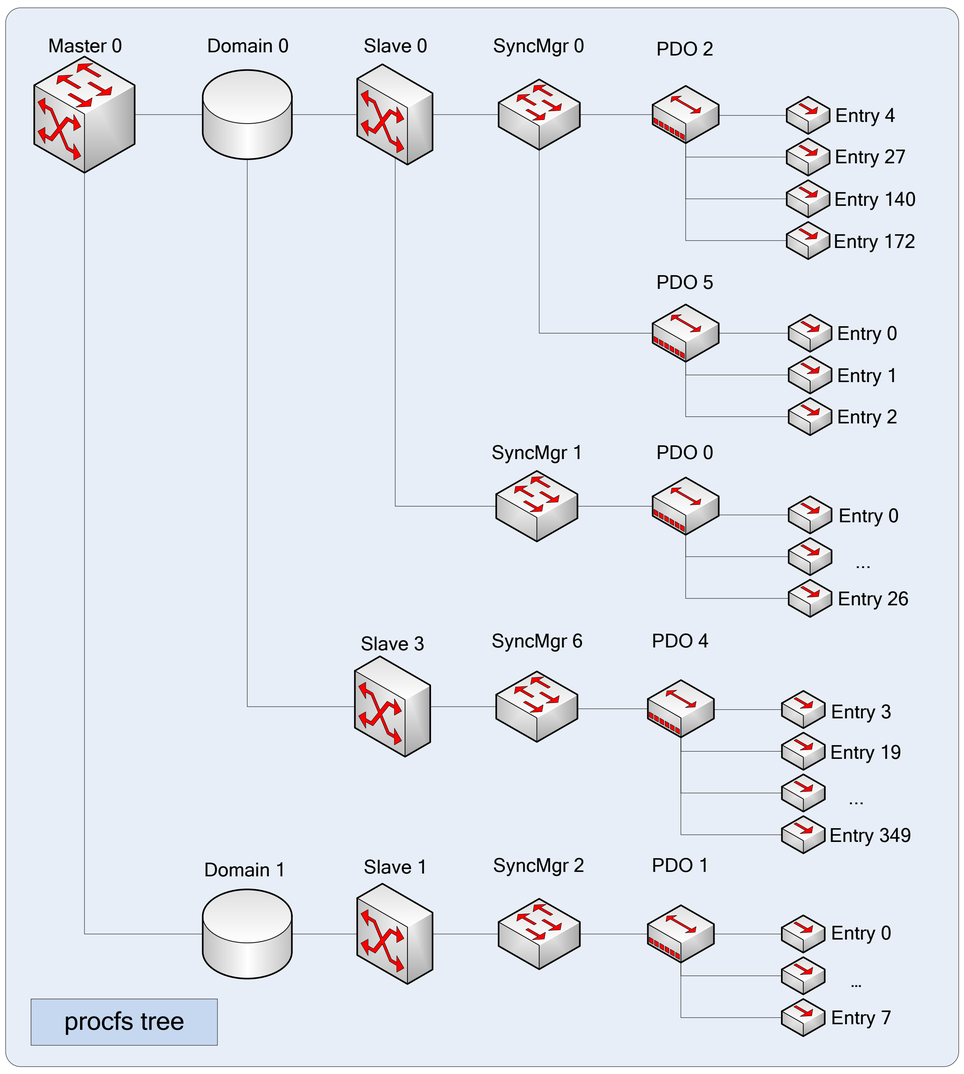
EPICS Control System - Full EPICS support and many useful extensions

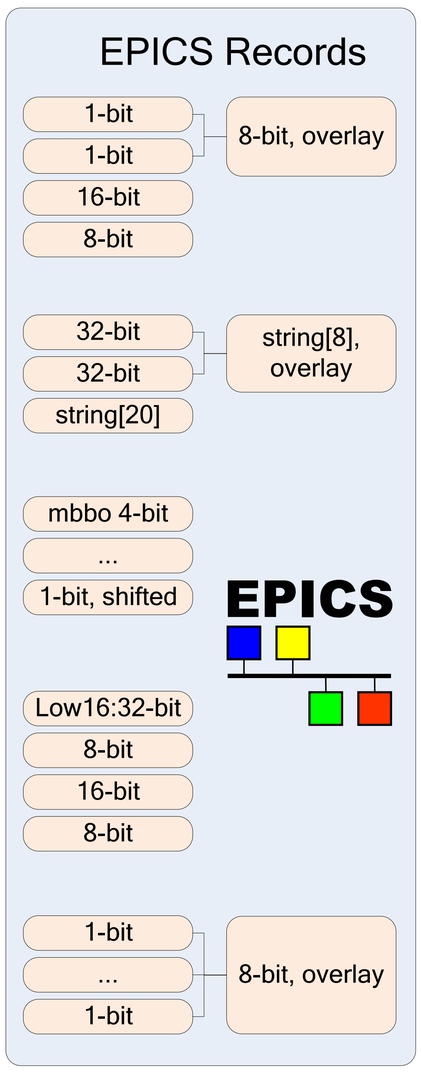
PSI EtherCAT EPICS driver supports all standard EPICS record types, and allows for interrupt-driven, real-time triggering of read/write operations for PDO Entries.

PSI addressing style (Slave.SM.PDO.Entry, i.e. s3.sm4.p0.e136) as well as a lot of addressing extensions and modes are available, such as:

- **forced offset** (.onnn, e.g. s0.sm0.p0.e0.o4 for +4 offset)
- **forced bit extraction** (.bnnn, e.g. s0.sm0.p0.e0.b3 for bit 3 of that entry)
- **domain register addressing** (r0-rnnn) for relative addressing
- **slave register addressing** for relative addressing of entries inside a slave module (.lrnnn) on any level, from SM to PDO to Entry
- **support for string in/out records** for string extraction from anywhere in the Domain
- **support for aai/aao records** for array-style extraction of data
- **type override** - freely change the type of the Entry being read or written from and to records, using any type supported by EPICS (regDev and/or PSI EtherCAT Driver style, e.g. t=float or t=uint32,...)

Finally, all of the above extended **addressing modes can be mixed as needed**, without any restriction. So it is possible to have a PDO Entry, but to shift the address being read, then to extract a single bit from the value read, and finally to force type change from bit (integer) to a float, for example.





Flexibility - Easy setup of programmable modules (EL6692 and similar)

Both PSI EtherCAT and Linux drivers support **easy setup of programmable modules**. For example, to create new EL6692 entries, simply add the following:
ecat2cfgEL6692 <bridge_nr> in <num_of_bits>
...and/or...
ecat2cfgEL6692 <bridge_nr> out <num_of_bits>

Slave-to-slave - Built-in support for real-time slave-to-slave communication

Any number of **slave-to-slave** transactions can be added to allow for **real-time communication** without any additional software. Simply list all the slave-to-slave transactions, and that's it!
ecat2sts r8 r0
ecat2sts r2.b0 r0.b6
ecat2sts s2.sm0.p1.e0 s1.sm0.p1.e0
ecat2sts s3.sm3.p0.e10.b3 s4.sm2.p1.e0.b7