Fieldbus Device Driver Interface

Document Version 0.1

01.05.2007

CONTENT

1 Overview

1.1 Layered IO-Framework Reference Model

Any driver software to access industrial IO according to the definitions of the OSADL belongs to one of the four layers of the I/O framework reference model:

System Interface (SI)

This interface accesses hardware and makes hardware entities available for the layers above in uniform way. It manages entities like interrupts, mapped memory etc. This layer is the only one, which runs partially in kernel space.

Low Level Device Interface (LLDI)

This interface supplies uniform access to IO-interface hardware with the same profile, in order allow upper layers to have only one implementation for a specific IO-protocol. For example the LLDI should provide a uniform access to all CAN interfaces, to allow hardware independent implementations of CANopen or DeviceNet Stacks. Other typical profiles are Ethernet interfaces, Sercos interfaces, serial interface, Hischer fieldbus card, etc. The different profiles do not have necessarily the same interface function definitions, but should follow the same patterns (for example the socket pattern) as far as possible.

Fieldbus Device Driver Interface (FDDI)

This interface allows applications to access IO-subsystems in a uniform way. It manages IO interfaces, IO devices, process data and IO channels.

Logical Device Interface (LDI)

This interface supplies application entities in a uniform way, independent of the underlying IO-interface. A typical entity supplied by the LDI is a servo drive.

1.2 Entities of the FDDI

The FDDI interface handles four different entities:

IO-Interfaces

IO-interfaces represent the software (that means the driver or protocol stack), which implements a specific I/O-protocol. The instance of an IO-interface is bound to a specific IO-Interface hardware supplied by the LLDI in the initialisation phase of the system. Operations supported for IO-interfaces and instances of IO-Interfaces are:

- Enumeration (divers and driver instances)
- Instantiation
- Information (name, version, required LLDI profile)
- Configuration (setup of parameters and expected IO-devices)
- Scan (list of the actual connected devices)
- Diagnosis/Operation state
- Commands (start, stop, reset, identify)
- Enumeration of configured devices (IO-devices)
- Enumeration of process data transport units (IO-transport units)
- Enumeration of IO-channels

IO-Devices

IO-devices are the configured devices of an IO-interface. If there is a physical device matching an IO-device it represents the physical device. There can be devices, which do not match with physical devices at a given time and there can be physical devices ,which are not associated with any IO-device. The IO-interface must do the matching between configured and physical devices at

configuration time and can support the matching at runtime if required. Operations supported for IO-devices are:

- Information (logical ID, physical ID, type ID, name)
- Diagnosis/operation state
- Asynchronous read/write services
- Commands (start, stop, reset, identify, save param)
- Enumeration of IO-channels

IO-Transport Units

IO-transport units are chunks of process data, transferred by the underlying low level device at the same time (from the application point of view). IO-transport units are **not** necessarily bound to IO-devices. An IO-transport unit can represent only a part of a device or can cover more than one device. Examples for I/O-transport units are CANopen PDOs, EtherCAT FMMUs, the complete input and the complete output process image of a intelligent field bus interface card. Any I/O transport operation is done with IO-transport units. The number, size and layout of all IO-transport units is determined by the configuration of the IO-interface.

The process memory contained in I/O-transport units are implemented in a managing layer, which is used by the LLDI, FDDI and the application layers above in common. This management layer handles access control and buffering.

Operations supported for transport units are:

- Information (ID, time stamp, direction: in/out, sync attribute)
- Access (get address to read, get address to write, release)
- Registration to receive event
- Explicit send
- Enumeration of IO-channels

IO-Channels

An IO-channel represents a single IO-signal. The data/value of a channel is stored on a certain offset in the IO-transport unit, to which the channel belongs. The following operation is available for IO-channels:

- Information (ID, Type/Size, IO-transport unit, IO-device, offset in memory of transport unit)

All information delivered can be assumed to be constant in a given configuration.

1.3 Typical usage scenario of the FDDI

To clarify the usage of the FDDI the typical sequence of operations is explained:

At system initialisation:

Available IO-Interfaces are enumerated and instantiated with the available IO-interfaces supplied by the LLDI.
For example a CANopen IO-interface is bound to the first CAN interface of the system and a DeviceNet IO-Interface is bound to the second CAN interface of the system.

At application initialisation:

The configuration of an IO-interface is done. The source of the configuration can be either a file or a explicit configuration. After the configuration all IO-devices, IO-transport units and IO-channels are defined for the specific interface. Before configuration just a few operation with the device are possible like scan or identify.

At application runtime:

The enumeration operations are used to get the I/O-channels. They are accessed through the IO-transport unit, which contained them. To access them a read or write address to the memory buffer of

the transport unit is retrieved. By using the offset and size of the channel, the value is read or written. After the access the pointer must be released. During the access it is guaranteed that no other layer of the IO-framework and no other client of the FDDI changes the process data.

If the IO-interface and the application are not strictly coupled the channels are simply accessed. If the IO-interface triggers the application, the registration to the receive event of I/O-transport units can be used to implement an event driven application. If the application triggers the I/O-interface the “explicit send” operation of the IO-transport unit can be used. In which way the transfer of the I/O-transport units of the IO-interface are triggered is determined by the configuration of the I/O-interface.

The state of the IO-devices attached to an IO-interface can be processed by enumeration them and use the diagnosis function of the IO-device.

2 API Interface Definitions

This chapter describes the access of applications and above layers to the FDDI. The implementation an registration of FDDIs is described in chapter 3.

2.1 IO-Interface

3 Implementing FDDI Drivers

Revision History

Version	Description	Author	Date
0.1	Overview	Dieter Hess	01.05.07