Building Door Control

Dieter Kuchenbecker, Effeff Fritz Fuss GmbH & Co. KGaA

Business buildings use doors with several electronic and electromechanical safety and security devices. These doors are for safety (escape route doors), doors for security areas, door interlock systems and automatic doors for convenience. The connection of various devices is today a challenge having as result the right door functionality. Some devices are also dependent from others, so the general function of the door depends also from the right coordination and control of those devices. Another problem is the wiring, some devices needs at least ten wires and if some of them are mounted on the door leaf, the problem is to carry the cable to the door frame.

For those reasons, a door device bus network is necessary. In building automation (e.g. access control), the door can be seen as a black box with defined functionality. The configuration and extension of those devices to the door functionality will become to be easy and flexible using a bus network.

Background

Since the antiquity mechanical locks were used to protect material assets. Fundamentally there were no changes until today. To lock a door, mechanical devices are still used. Electromechanical strikes were invented for convenience to unlock a door that is far away. Today in business buildings there are much more different devices mounted around a door to increase security, safety and convenience, e.g. security locks, access control, card readers, panic bars, escape route systems and door operators for a comfortably door pass of handicapped persons.

Problematic

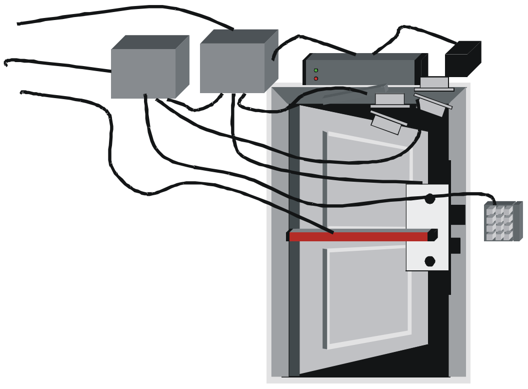
Figure 1: Devices around a door in a conventional manner

It is very difficult to connect different devices together to fulfill tasks of an access controlled door or special door conditions and dependencies between the devices are very critical. Normally the devices have more than ten wires to connect because of some micro switches inside, therefore it is not possible to get the right task without an additional relay box. So there is a need to reduce the wire numbers and increase the intelligence of the devices itself to make installation much easier.

CAN-Bus Network

A device bus network is used to reduce wiring, increase flexibility and simple retrofit of new devices. A gateway is the physical and logical interface to an upper level system, e.g. a building management.

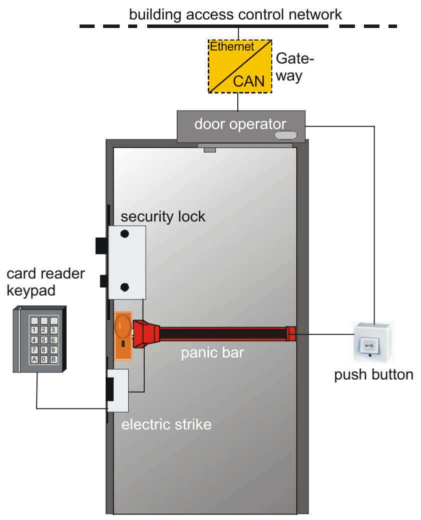

Figure 2: Door device bus network

Then, the door can be seen as a black box with a defined functionality. Also the configuration can be done over the bus network.

Requirements

As seen in Figure 2, all the devices must have a small electronic circuit board inside, it must be very cheap and robust concerning EMC/EMI. CAN fulfills these physical requirements. Many of the micro controllers have CAN inside, so there is no need for an additional CAN controller. The wiring must be very easy with a free bus topology. The data transfer should be secure and a node guarding is required. CANopen is a protocol, that covers all the logical requirements. Finally a very easy putting into operation like Plug & Play is necessary. This requires, that the logical connections between the devices have to be established automatically. Every device must be able to determine which potential communication partner is present in the network. Dependent on this information, the logical connections are established.

Node-ID

Every node of the CAN bus system needs a node ID for addressing the message. Normally it will be set with dip switches or other switches, mounted on the printed circuit board. Because of very small devices like an electric strike, it is not possible to mount dip switches on board. And to reduce faults in adjusting the IDs, an automatism is used to adjust the node IDs. For such an automatism, some preconditions are determined:

- same baud rate for all devices
- every device must have a unique individual number
- every device must be connected to the same network
- simultaneous power up of all devices
- during the procedure, all devices will be transmitters and receivers of the same CAN message

Node-ID acquisition procedure

The acquisition procedure is divided into three steps, the start-up synchronizing, the claiming process and the claiming message at the end of a successful node-ID acquisition.

Device start-up synchronization

The problem in real systems is the different start-up time of different devices. During the startup time, activities like initializing ROM, RAM and other peripherals are preformed. Obviously the time spent for these actions depend strongly on the implementation. The start-up synchronization compensates the different timings in a way that all connected devices are guaranteed to have the same starting point in time.

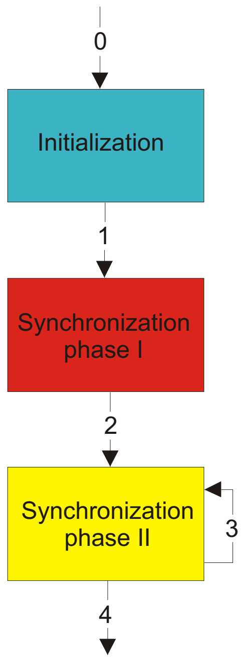

Figure 3: State machine synchronization process

The state machine for the synchronization comprises the following states:

- Initialization – Initializing the hardware
- Synchronization phase I – device is initialized and the CAN controller is already active, no messages are evaluated
- Synchronization phase II – device is aware of any synchronization message received over the bus

The state machine for the synchronization comprises the following transitions:

0 After power-on or reset, the device starts here.

1 CAN controller is initialized and the device enters the synchronization phase I.

2 After the synchronization phase I is elapsed (fixed time for all devices), the synchronization message is transmitted over the bus.

3 The synchronization message was received over the bus. The phase II is restarted again.

4 The time for synchronization phase II is elapsed (fixed time for all devices), the claiming procedure starts.

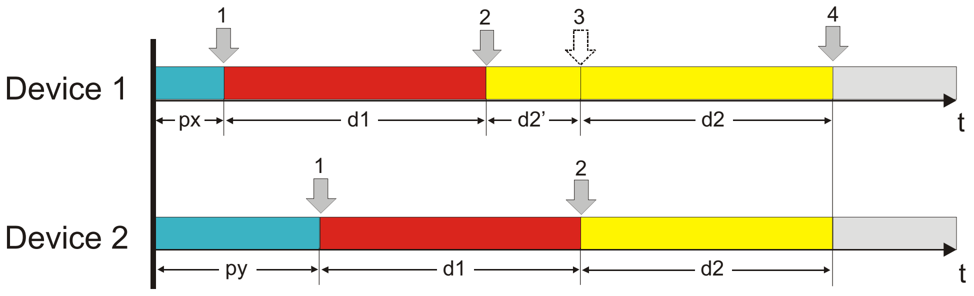

Figure 4: Synchronization process

px –	start up time device 1, depends to internal initialization
py –	start up time device 2, depends to internal initialization
d1 –	duration of synchronization phase I, global parameter
d2 –	duration of synchronization phase II, global parameter
1 –	starting phase I from device 1 / 2 after internal initialization
2 –	transmission of synchronization message from device 1 / 2
3 –	reception of synchronization message from device 2
4 –	both devices are internally synchronized

Claiming process

sync –	claiming cycle synchronization
am1 –	arbitration message 1
am2 –	arbitration message 2
...	
amn –	arbitration message n
cm –	claiming message
opt –	optional: additional claiming messages

After all devices are synchronized, all devices starts to claim a node-ID using the claiming cycle synchronization (sync) and sending the first arbitration message (am1), which depends on the Vendor-ID and the individual number as stored in the device. Any device detecting a higher prior arbitration message than it is trying to transmit, leave the current claiming cycle. At the end of the arbitration (arbitration message n, amn), there is only one device left, claiming a node-ID. Then it transmits the claiming message (cm) containing the claimed node-ID and other information for Plug & Play functionality. This claiming message will be received by all the devices. If there is no additional claiming message (opt), a new claiming cycle starts with the remaining devices. The "winner" of this cycle will also send a claiming message. This process continues until the last device has claimed its node-ID. That is, that all devices in the network know, how many devices are connected to the network and which node-IDs are used. The method of the claiming cycle relies on time slots (see Figure 5). For every device present in the network, one claiming cycle is necessary. Every claiming cycle consists of a synchronization slot, *n* arbitration slots and at least one claiming

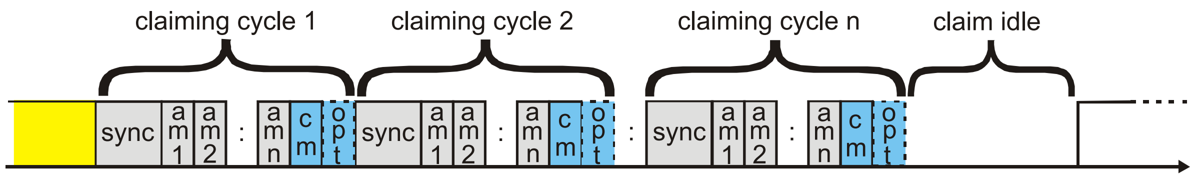

Figure 5: Claiming cycles

message slot. Additional claiming message slots are possible. After claiming, every claimed device will wait for the claiming idle time to find out, if it was the last claiming cycle. The last claimed device send the NMT zero message to boot-up the network.

Running system

After boot-up of the network, all devices will send the state of their application data to update all connected devices. Then the application data exchange is event-based.

Communication principle

The communication is based on the producer/consumer relationship. The PDO communication is done by the first transmit PDO (180_h + node-ID) and up to 127 receive PDOs (180_h + node-ID). The first transmit PDO will be received by all physical devices in the network except the transmitter itself and is filtered through the object mask. The object mask decides if the data contents have to be evaluated further or have to be discarded.

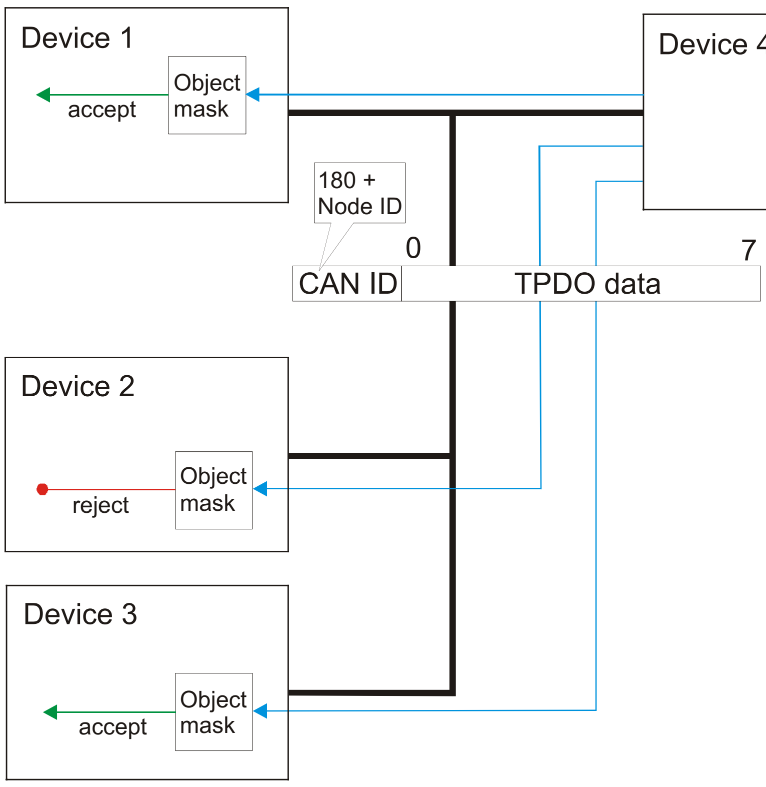

Figure 6: Communication model

Virtual devices

The application data is grouped in so called "virtual devices". Every virtual device represents one functionality of the physical device and every physical device covers in minimum one virtual device. Normally, a physical device covers a set of virtual devices because of different functionality's. The transmit PDO contains the object of exactly one virtual device, therefore the communication between the physical devices is virtual device based.

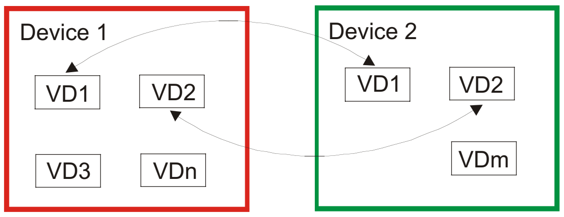

Figure 7: Physical device with virtual devices

As shown in Figure 7, only the virtual devices "talk" together. According to this, it is necessary to connect physical devices together, that have in minimum one of the same virtual device implemented.

Every virtual device contains four object entries (Table 1).

The first virtual device object is the real application data object to distribute over the bus (one functionality of the physical device). This is e.g. the sensor data of a device. Every change of the sensor data will cause a transmit PDO and will be received by all other devices with the receive part of this virtual device (second object of the virtual device), filtered by the object mask (third object of the virtual device).

Index	Object
6xx0$_h$	application data (transmit data)
6xx1$_h$	data collection (receive data)
6xx2$_h$	object mask
6xx3$_h$	configuration
	...

Table 1: Virtual device structure

The second virtual device object is the data collection for the same virtual device application data of other physical devices. This is organized in a node-ID based array and is an image of all same existing virtual device data. The application may evaluate the collection data to make decisions.

The third virtual device object is the object mask to accept or reject the received application data of other physical devices. This is organized in a node-ID based bit array and is normally changed by configuration. The object mask has the advantage to channel application data.

The fourth virtual device object is for all configuration data of the virtual device. It contains times, count values, distances, threshold values, etc. to configure the functionality of the virtual device, respectively the physical device. This is the area for the customer to make the device work as he want. Pre-defined default values gives the device a standard behavior.

Virtual devices gives the physical device a very clear and easy to understand data structure.

Future perspectives

Uplifting the physical in- and outputs (application data) to logical in- and outputs, the system flexibility raises and key words like “intelligent door” or “intelligent room” becomes more and more reality. This can be an access control to a room or a room zone with complex identification technologies (fingerprint-reader, palm-reader, iris scanner, face- or speech-identification, etc.). Furthermore this technology simplifies the centralized monitoring and configuration of such a system, e.g. to forward disturbances or to change the functionality of a door dynamically. Building management will be easier, because every door system can be seen as a black box with defined functionality and can simply be influenced by logical commands.

Dieter Kuchenbecker
Effeff Fritz Fuss GmbH & Co. KGaA
Bildstockstr. 20, 72458 Albstadt
Phone: 07431 / 123-112
Fax: 07431 / 123-65112
dieter.kuchenbecker@effeff.com
www.effeff.com