INTERNATIONAL STANDARD

ISO 20242-1

First edition
2005-10-15

Industrial automation systems and integration — Service interface for testing applications —

Part 1: **Overview**

Systèmes d'automatisation industrielle et intégration — Interface de service pour contrôler les applications —

Partie 1: Vue d'ensemble

Reference number
ISO 20242-1:2005(E)

PDF disclaimer

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 20242-1 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration*, Subcommittee SC 5, *Architecture, communications and integration frameworks*.

In addition to this part, ISO 20242 is envisaged to consist of several more parts dealing with:

- Resource management service interface;
- Virtual device service interface;
- Device capability profile template;
- Application program service interface;
- Conformance test methods, criteria and reports.

Introduction

The motivation for this International Standard stems from international automotive industries and their suppliers to facilitate the integration of automation and measurement devices, and other peripheral components for this purpose, into computer based applications. It defines rules for the construction of device drivers and their behaviour in the context of an automation and/or measurement application.

The main goal of ISO 20242 is to provide users with:

- independence from the computer operating system;
- independence from the device connection technology (device interface/network);
- independence from device suppliers;
- the ability to certify device drivers with connected devices and their behaviour in the context of a given computer platform;
- independence from the technological device development in the future.

ISO 20242 will not involve the development of new device families or the use of special interface technologies (networks). It encapsulates a device and its communication interface to make it compatible with other devices of that kind for a given application.

Industrial automation systems and integration — Service interface for testing applications —

Part 1: Overview

1 Scope

This part of ISO 20242 provides an overview of the particularities of this International Standard and its use in the computer aided testing environment.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

2.1

application program

resource designed to help users perform a specific task

NOTE In this International Standard, an application program does any task necessary to run a computer-aided test station running, with the special requirement that communication with peripheral devices is done.

2.2

application program service interface

interface to communicate with a coordinator

NOTE This will be specified in another part of ISO 20242.

2.3

communication object

existing object which may be accessed with a communication function to read or write a value

2.4

coordinator

program with a specified interface to handle the access of an application program to one or more device drivers and to manage real-time application aspects, synchronization and events

2.5

device capability description

text file containing information about the capabilities of virtual devices in a defined format (i.e. structure, syntax)

NOTE This will be specified in another part of ISO 20242.

2.6

device driver

program with an ISO 20242-specified interface containing service functions that call the platform adapter to access physical devices

2.7

interface

<device capability description> keyword identifying a class for the description of device functions

NOTE A device function inside the device driver is an instance of such an interface.

2.8

interface driver

program handling the data transfer via a peripheral interface

2.9

module

ISO 20242-defined keyword, identifying a class for the description of virtual devices

NOTE A virtual device inside the device driver is an instance of a module.

2.10

platform adapter

program with an ISO 20242-specified interface, encapsulating the computer hardware and its periphery and providing services to communicate with connected devices and to use other resources of the computer operating system

2.11

resource management service interface

set of ISO 20242-specified service functions to communicate with a platform adapter

2.12

virtual device

representation of one or more physical devices and/or stand-alone program entities to provide an unambiguous view on the resources of a communication interface

2.13

virtual device service interface

set of ISO 20242-specified service functions to communicate with a virtual device

NOTE These service functions use the Resource Management Service Interface (platform adapter) to access physical devices and/or provide the needed capabilities by contained software tasks.

3 Abbreviations

APSI Application Program Service Interface

ASCII American Standard Code for Information Interchange

CAQ Computer Aided Quality Assurance

CAT Computer Aided Testing

CIM Computer Integrated Manufacturing

CORBA Common Object Request Broker Architecture

DCD Device Capability Description

DCPT Device Capability Profile Template

OOP Object-Oriented Programming

PA Platform Adapter

PDU Protocol Data Unit

RMSI Resource Management Service Interface

XML eXtensible Markup Language

VD Virtual Device

VDSI Virtual Device Service Interface

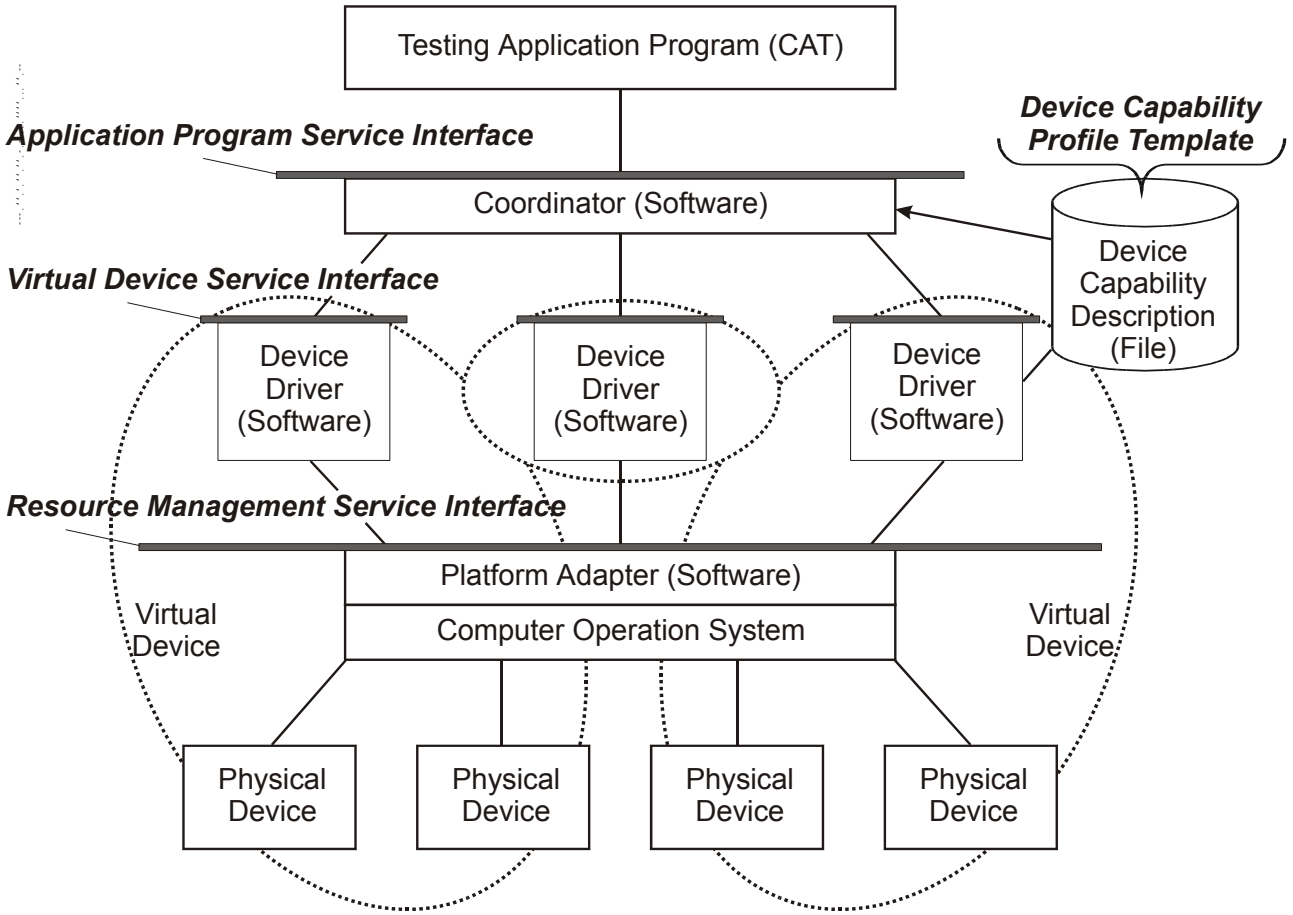

Figure 1 — Service interfaces defined in ISO 20242

4 Service interface concept

4.1 General

An essential function of an ongoing manufacturing operation is its quality assurance to make sure that high quality is designed into the product and that the manufacturing operation produces the desired quality. For control of this quality, test stations are implemented which may be part of the manufacturing process itself or are stand-alone systems. The computer is an extremely effective tool to control test stations, monitor performance of processes, collect and evaluate data, and issue quality reports.

The use of computers in test stations is called Computer-Aided Testing (CAT), which falls under the Computer-Aided Quality Control (CAQ) heading in the Computer Integrated Manufacturing (CIM) terminology. Test stations use measurement and automation devices to acquire data and to control test scenarios (see Figure 1).

The application programs on computers in CAT communicate with automation and measurement devices via interfaces. This part of ISO 20242 defines the Application Program Service Interface (APSI) that contains a list of defined services to access any number of device drivers with included virtual devices.

Device drivers are software modules with another interface defined by this part of ISO 20242, the Virtual Device Service Interface (VDSI). They present the capabilities of peripheral devices and optional software resources inside the driver via virtual devices. The structure of virtual devices and their status regarding communication is captured by a device capability description (DCD). For the creation of the DCD, which is an ASCII file, this part of ISO 20242 defines the Device Capability Profile Template (DCPT).

The computer hardware and the operating system, i.e the platform, is covered by the Resource Management Service Interface (RMSI) that allows a testing application to be platform-independent. A platform adapter (PA) handles the data transfer with peripheral devices and the access to operating system resources.

4.2 Platform adapter

4.2.1 Features

The platform adapter (see Figure 2) is a software module that covers the peripheral interfaces and the operating system resources of a computer and presents them via the Resource Management Service Interface (RMSI). Thus the device driver, which is the user of the RMSI, does not depend on a specific operating system or on specific peripheral interface drivers. If an operating system does not provide the needed resources, a platform adapter will add capabilities.

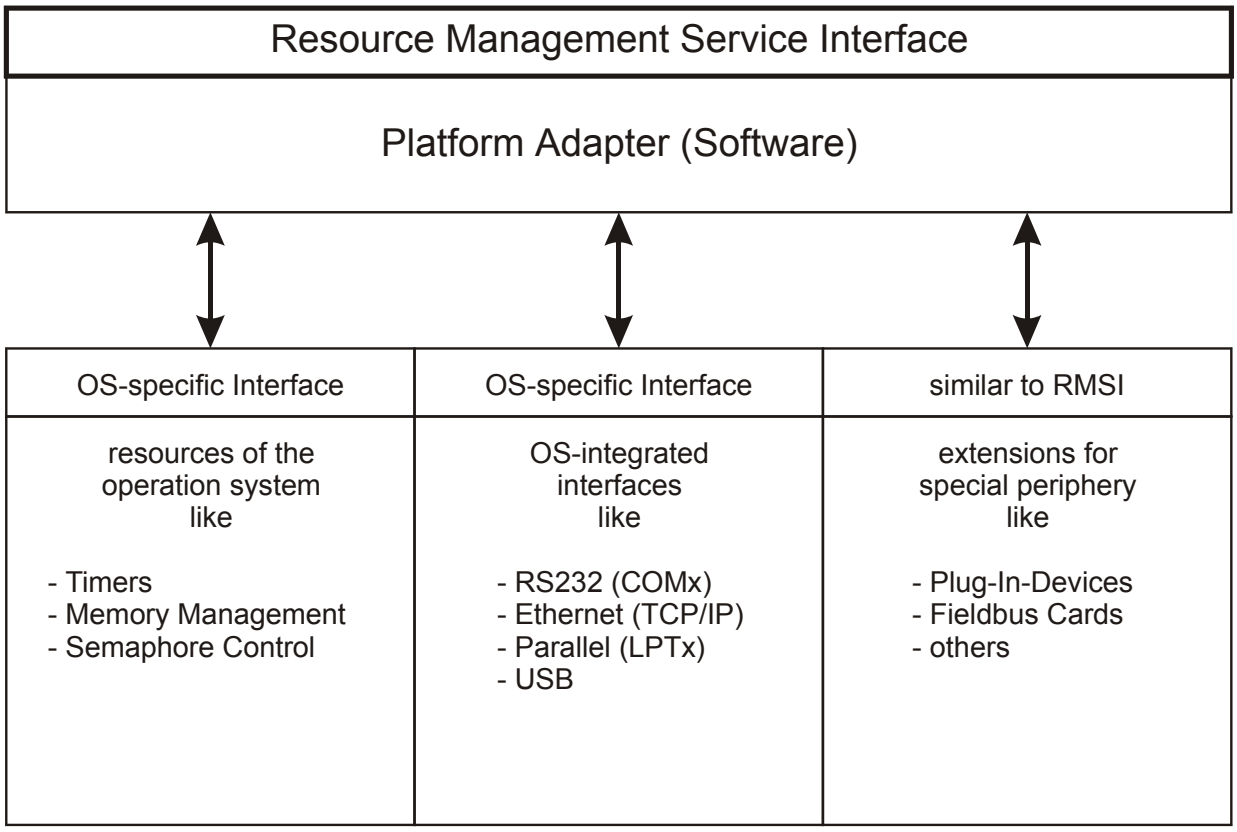

Figure 2 — Access to peripheral interfaces and to resources of the operating system

Extensions are dynamically loadable communication resources that are not available with the operating system. Their interface is the same as RMSI, so the platform adapter only has to route user access from RMSI to the extensions. Some special services of RMSI handle the loading and unloading of extensions.

4.2.2 Device communication aspects

The platform adapter uses the resources of the operating system and/or extensions for the data transfer to physical devices. It depends on the type of interface, which communication layer is presented by the RMSI. This may be the transport layer for typical peripheral interfaces of a computer. But if a platform adapter provides special communication protocols it may present higher layers.

For the communication via fieldbus the protocol procedures in most cases are located on a plug-in-board. It may depend on the resources of the plug-in-board, which communication layer is presented by the platform adapter.

The communication attributes of a platform adapter service are defined by the type name of the interface. ISO 20242 defines a few names for typical computer integrated periphery. If different platform adapters on different operating systems and/or with different extensions use the same type name, the related services will have the same behaviour. For the use of RMSI it makes no difference whether a communication protocol is handled within the platform adapter or within an extension.

4.3 Device driver

4.3.1 Features

The device driver is a program which submits access to communication objects of virtual devices. There is no automatic communication with real devices on such an access, because virtual devices and real devices are not directly mapped (see Figure 3).

Virtual devices are defined from the needs of the testing application . The application requires functionalities that may be presented by one or more physical devices . The functionalities are grouped according to virtual devices that may belong to

- a dedicated physical device,
- a number of physical devices,
- a part of a physical device, or
- a software module inside the device driver or inside a Platform adapter extension.

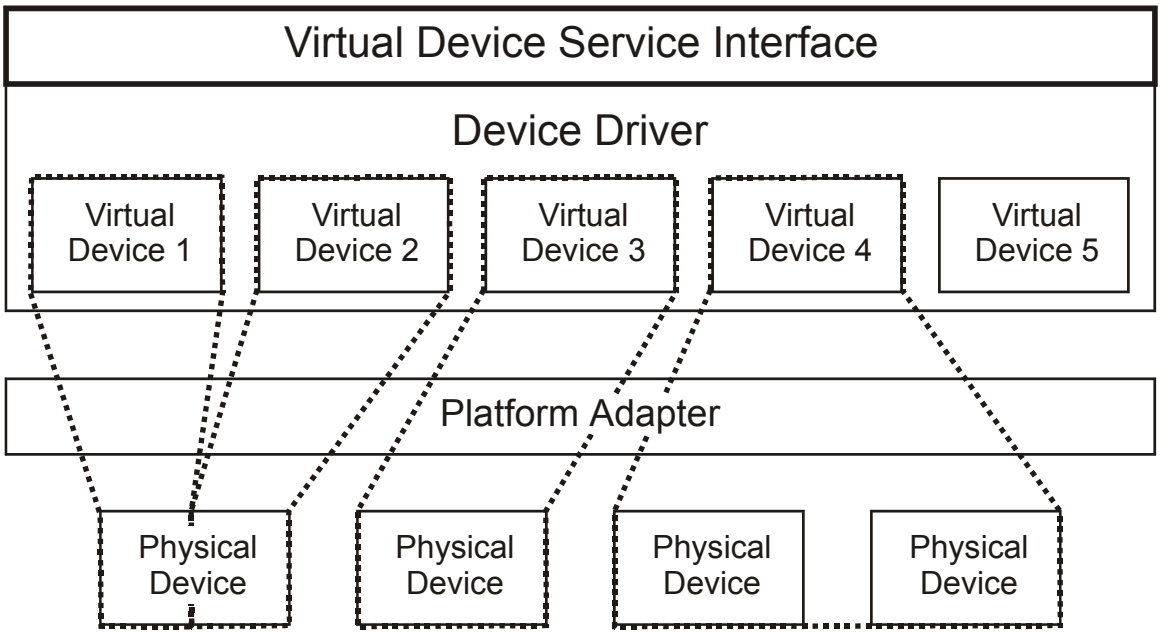

Figure 3 — Mapping of Virtual Devices

4.3.2 Device communication aspects

The real communication, consisting of PDU generators and interpreters in several layers of a communication stack, is not seen by the user of the Virtual Device Service Interface (VDSI). The communication is handled inside the device driver. The communication effort depends on the communication layer presented by the RMSI and the communication demands of the physical devices.

4.4 Device capability description

4.4.1 Features

The device capability description is a file containing ASCII text to describe the structure and behaviour of virtual devices. The syntax and semantics are described with the Device Capability Profile Template (DCPT) that also contains rules for the behaviour of virtual devices related to the use of the VDSI.

Virtual devices (see Figure 4) are built from device functions that characterize the devices' capabilities. Device functions contain communication objects for data exchange and/or executable programs, called operations, for more complex, completed device activities.

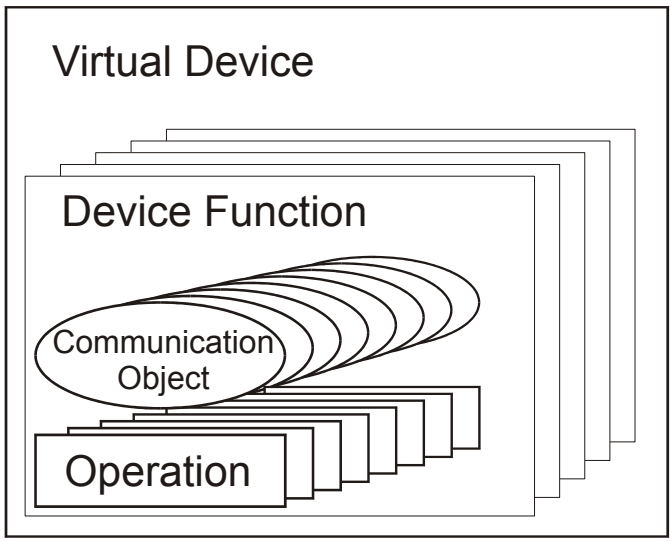

Figure 4 — Internal structure of Virtual Devices

4.4.2 Modules

If more virtual devices of the same kind are used, it is not necessary to describe each of the virtual devices. To handle this, the DCPT defines modules, which are used like classes in object-oriented programming (OOP). VDSI provides a service to instantiate a module, which is creating a virtual device.

It depends on the physical resources, i.e. the number of connected physical devices and the management resources of the device driver and the platform adapter, how many instances of a module can be built.

4.4.3 Interfaces

The class model is also used for device functions by interfaces. The instance of an interface is a device function. VDSI provides a service for the instantiation.

An interface is identified explicitly by a number given with the device capability description. The instantiation of an interface provides a handle to identify the device function.

An interface contains the elements Create Parameters, Parameters, Attributes and Operations (see Figure 5) with the following meaning:

a) Create Parameters — This defines an invariable configuration of the device function (e.g. the type of a signal filter). In OOP terminology, these are parameters for the constructor;

b) Parameters — A parameter can influence the behaviour of a device function (e.g. the frequency limit of a signal filter). Parameters may only be changed while setting up the device function;

c) Attributes — Attributes are the input and/or output data (process data) of a device function (e.g. the input and output connection of a signal filter). Attributes may be accessed at any time they exist;

d) Operations — Operations are provided for more complex completed device activities (procedures). Operations may have start parameters and results.

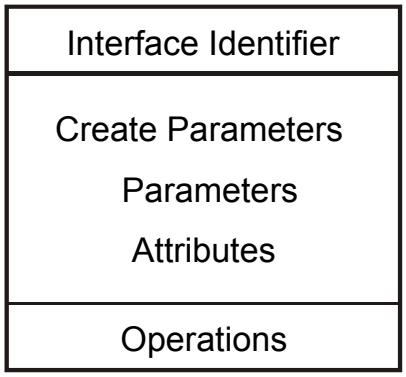

Figure 5 — Elements of interfaces

4.4.4 Communication objects

Parameters and attributes are communication objects. For data access, the VDSI defines services. But these services may not be used before a communication object is prepared with another service. So a testing application may create only the communication objects which are really needed.

4.4.5 Behaviour of virtual devices

After instantiation, virtual devices have specific operating states (see Annex B).

4.5 Coordinator

The coordinator (see Figure 6) is a program that handles the following procedures:

a) Controlling of run time features — The coordinator manages CPU resources (multi-tasking, multi-threading, etc.);

b) Organizing event handling — Events are caused by asynchronous communication and by unsolicited access of the device driver (to present new values or to retrieve regulating variables);

c) Synchronizing different device drivers — The application may access combinations of values (structures) from different device drivers;

d) Multiplexing for several concurrent running device drivers — The testing application may access process data via device drivers and physical devices of different vendors;

e) Interpreting the device capability description and instantiating virtual devices, device functions and communication objects.

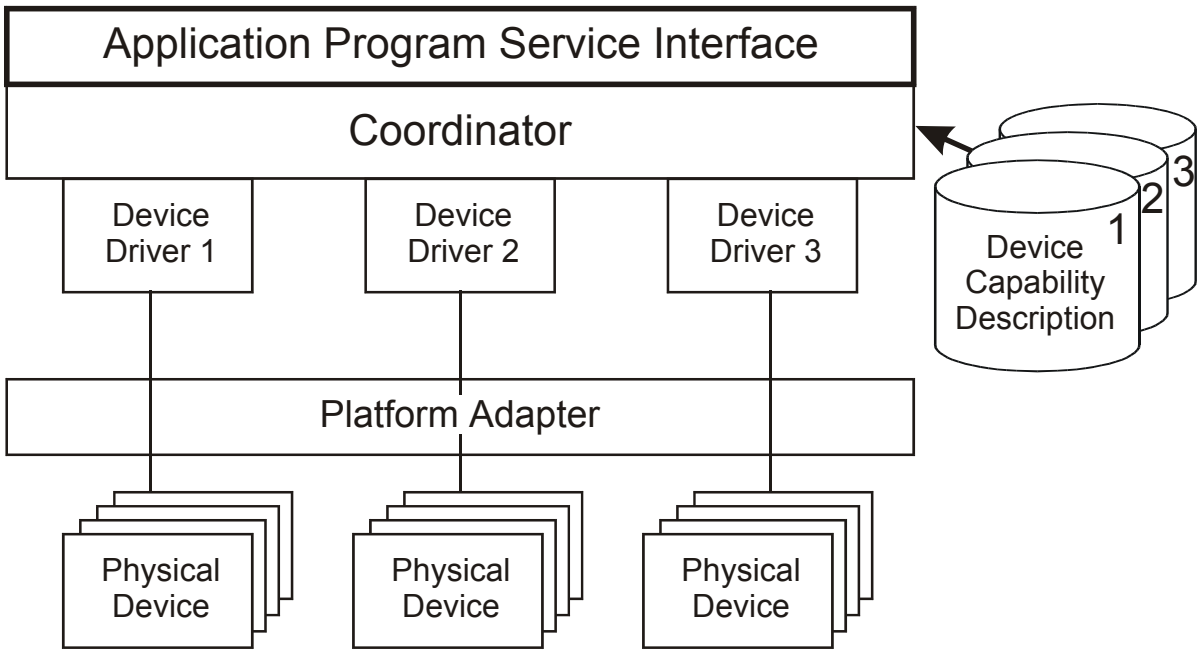

Figure 6 – Coordinator handling different device drivers

The testing application accesses communication objects inside the coordinator via the Application Program Service Interface (APSI). The assignment of these communication objects to the device drivers or even to physical devices is not relevant for the testing application.

Annex A
(informative)

Use case for ISO 20242

A.1 Activities of users

Users manage the testing application and select the computer platform that is used for running the application program. Users provide a coordinator or manage an equivalent property inside the testing application. Based on the application requirements they define the capabilities of virtual devices (modules) and create a related invitation to bid.

It is recommended that ISO 15745-1 be used to describe a profile of the required Virtual Device.

A.2 Activities of vendors

With the profiles of the virtual devices vendors create the device capability description by using the ISO 20242-specified Device Capability Profile Template (DCPT).

After this, vendors select the physical devices for this application. Vendors are free to take one special device developed for this application or any combination of off-the-shelf devices.

The peripheral interface for the chosen devices have to be embedded in a platform adapter for the given computer platform with the ISO 20242-specified Resource Management Service Interface (RMSI). The related development may be commissioned to specialists of the platform or a suitable existing platform adapter is used.

If a Platform adapter for the given platform exists but the interface of a chosen device is missing, the vendor may develop an extension for the new interface, which will be loaded by the existing platform adapter.

The vendor develops the device driver with the ISO 20242-specified Virtual Device Service Interface (VDSI). This device driver, which does not depend on the platform, can be written in C or C++ programming language. If users change platforms (e.g. with updates of operating systems or change of computer hardware), they need only to order a new platform adapter and compile the device driver again.

A.3 Further activities

Based on the device capability description users check their application requirements again and add capabilities if necessary. A refined device capability description is used for customizing the device driver.

Device simulating drivers may be used to test the application. Such simulators are not vendor specific and may be developed as universal tools which are self adjusting by interpreting the device capability description.

Annex B
(informative)

State diagram for virtual devices

To move a virtual device from one state to the next, the operations of the predefined interface *TransitionFct* (transition function) are used. This transition function belongs to an intrinsic virtual device, which is the basic VD of a device driver, called the Control VD.

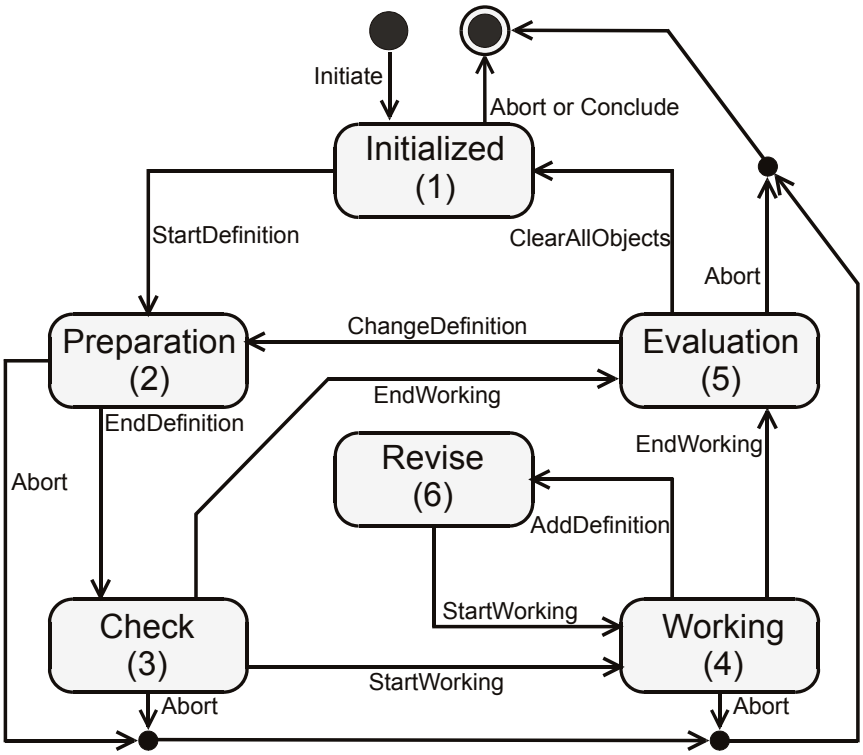

Figure B.1 — Different states of a virtual device

These virtual device states must not be confused with the measurement states of the testing application or the real devices. In most cases, only within the working state of a virtual device the states of the application or the real devices are relevant and will have extra communication objects to handle that. The states of a virtual device represent different stages of adapting it to the testing application by communication.

Annex C
(informative)

Cascading of device drivers

The organisation of functionalities in different device drivers with a Virtual Device Service Interface is a matter of implementation and may be done in a hierarchical structure with several layers (see Figure C.1).

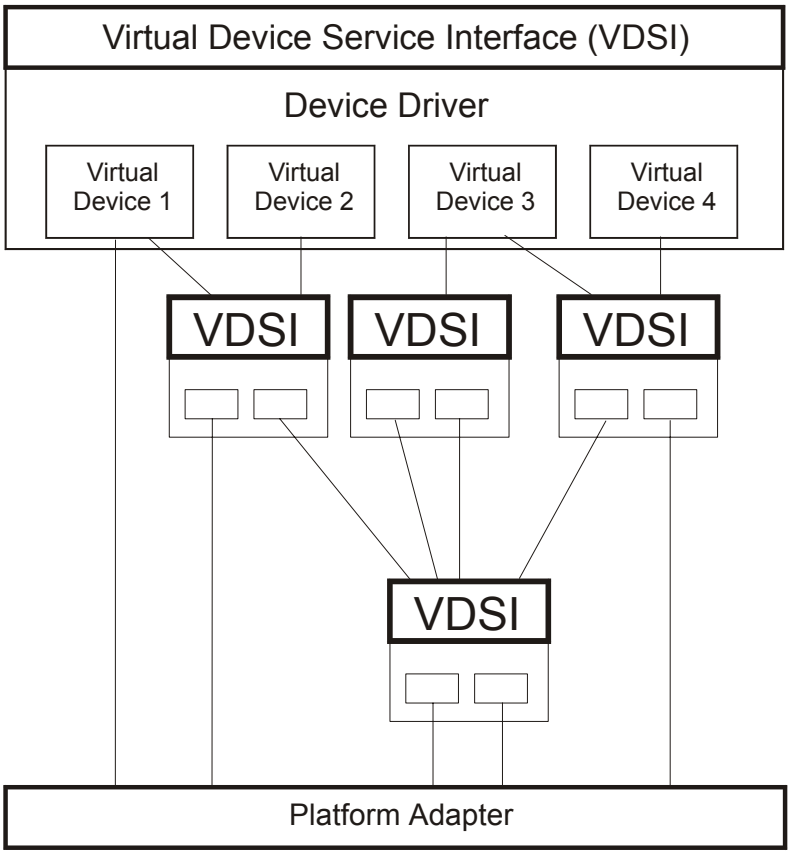

Figure C.1 — Example Cascading of Device Drivers

So there may be device drivers with a VDSI, which use functionalities via another VDSI instead via the Platform Adapter. The cascading of drivers is not seen at the VDSI.

ICS 25.040.40

Price based on 11 pages