WELMEC 7.1
(Issue 1)

WELMEC

European cooperation in legal metrology

Software Requirements on the Basis of the Measuring Instruments Directive

October 1999

WELMEC is a cooperation between the legal metrology services of the Member States of the European Union and EFTA. This document is one of a number of Guides published by WELMEC to provide guidance to manufacturers of measuring instruments and to notified bodies responsible for conformity assessment of their products. The Guides are purely advisory and do not themselves impose any restrictions or additional technical requirements beyond those contained in relevant EC Directives. Alternative approaches may be acceptable, but the guidance provided in this document represents the considered view of WELMEC as to the best practice to be followed.

SOFTWARE REQUIREMENTS ON THE BASIS OF THE MEASURING INSTRUMENTS DIRECTIVE (MID)

CONTENTS

1 Introduction **6**

1.1 Background and Scope 6
1.2 Conception 7

2 Terminology **8**

2.1 Program code 8
2.2 Legally Relevant Software 8
2.3 Changes of Software 10
2.3.1 Unintentional changes 10
2.3.2 Intentional changes (corruption, misuse) with simple common software tools 10
2.3.3 Intentional changes (corruption, misuse) with special sophisticated software tools 10
2.4 Protection of Software 11
2.4.1 Protected software 11
2.4.2 Audit trail 11
2.5 Interfaces 12
2.5.1 Hardware interface 12
2.5.2 Protective interface 12
2.5.3 Software interface 12
2.5.4 Protective software interface 12
2.6 Data security 13

3 Essential Software Requirements **14**

4 Definition of Levels **16**

4.1 Software Protection Level 16
4.2 Software Examination Level (Type Examination or Design Examination) 17
4.3 Degree of Software Conformity 18

5 Technical Features of Measuring Instruments and Systems **20**

5.1 Hardware Configuration 20
5.2 User Interface (Shell) 20
5.3 Software Loading 20
5.4 Software Structure 21
5.5 Software Environment 21
5.6 Fault Detection 21
5.7 Longterm Storage of Measurement Values 21
5.8 Measuring principle 21

6 Interpretation of the Essential Software Requirements for Selected Measuring Instruments and Systems **23**

6.1 Example A: Simple Stand-Alone Measuring Instrument 23
6.1.1 Description of the Instrument 23
6.1.2 Legal Classification 24
6.1.3 Technical Classification 25
6.1.4 Interpretation of the Essential Software Requirements 25

6.2 Example B: PC-based, Modular Complex Measuring System 31
6.2.1 Description of the System 31
6.2.2 Legal Classification 32
6.2.3 Technical Classification 33
6.2.4 Interpretation of the Essential Software Requirements 33

7 References and Other Literature **47**

Annex I **48**

Proposal for the Assignment of Levels (see Chapter 4) 48

1 Introduction

1.1 Background and Scope

The Measuring Instruments Directive (MID) [1] will contain "Essential requirements" (Annex I) for measuring instruments used for legal purposes. Some of these essential requirements can be directly applied to the software controlling these instruments:

(i) Annex I, No. 13 [1]:
"Software that is critical for metrological characteristics shall be identified as such and shall be secured. Its identification shall be easily available. Evidence of an intervention shall be available for a reasonable period of time."

(ii) Annex I, No. 14 [1]:
"Measurement data and metrologically important parameters stored or transmitted shall be adequately protected against accidental or intentional corruption."

Other essential requirements can be applied to both hardware and software of a measuring instrument, e.g.:

(iii) Annex I, No. 7 [1]:
"A measuring instrument shall have no feature likely to facilitate fraudulent use, whereas possibilities for unintentional misuse shall be minimal."

(iv) Annex I, No. 11 [1]:
"The metrological characteristics of a measuring instrument shall not be inadmissibly influenced by the connection to it of another device, by any feature of the connected device itself or by any remote device that communicates with the measuring instrument."

(v) Annex I, No. 27 [1]:
"A measuring instrument shall be designed so as to allow ready evaluation of its conformity with the requirements of this directive."

It has been the experience with the Directive 90/384/EEC [2] for non-automatic weighing instruments (NAWI) that these kinds of essential requirements need a uniform interpretation with regard to software, in order to avoid an unequal treatment of customers by the various European Notified Bodies. The result of respective discussions in WELMEC WG2 was the publication of the WELMEC Guide 2.3 "Guide for Examining Software (Non-automatic Weighing Instruments)" [3] in 1995 which, since 1997, was applied to automatic weighing instruments as well. The guide 2.3 primarily deals with systems that contain free programmable computers.

Based on the experiences with the WELMEC Guide 2.3 for weighing instruments, and due to the growing importance of software in legal metrology, the new WELMEC Working Group 7 "Software" started its work in 1996 as the successor of the former WG7 on Peripheral Equipment, Interfaces and Microcomputers. The scope of the new WG7 "Software" is to harmonise the type approval practice of the European Notified Bodies with respect to the software of measuring instruments covered by the MID.

The Guide in hand is the result of the 2 years work of the new WELMEC WG7. It tries to make the reader aware of the fact that only testing the metrological performance of an instrument without especially taking care of the software controlling this instrument is in many cases no longer adequate for modern, microprocessor-controlled or even PC-based measuring instruments, as it is substantially the software and its integrity that determines the metrological properties and reliability of an instrument. As the Guide covers very different categories of measuring instruments it can give only the basics of software examination. It is intended to be successively amended by specific annexes for each kind of measuring instrument similar to the specific annexes contained in the MID.

This Guide is intended to support a uniform software examination in Europe and to make the result of an examination estimable for the manufacturer. The guide is, however, not mandatory, even for those instruments that are covered by the MID.

1.2 Conception

The guide contains in chapter 2 a summary of the most fundamental terminology used.

In chapter 3 "Essential software requirements" are derived from the MID, Annex I. These are very close to the essential requirements of the MID. For practical applications it is necessary to interpret and further detail these requirements taking into account the requirements in the instrument-specific annexes of the MID, the various fields of applications of measuring instruments and technical aspects like the hardware and software configuration of an instrument.

It has been the experience in type approval practice that different kinds of measuring instruments are not treated equally within a country and that the same kind of instrument is treated differently in different countries without an obvious objective reason. Therefore in chapter 4 the facts or criteria that make up the different evaluation of the instruments have been identified as:

- the strength of protection of the software against changes
- the intensity of examination of the software at type approval
- the degree of conformity between the software implemented in a verified instrument and the approved software

These facts and criteria levels are defined and assigned to groups/categories of measuring instruments as a guiding, non-binding principle.

Chapter 5 describes the technical features of measuring instruments and measuring systems that have to be taken into account for the software examination. The possible hardware and software configurations are presented as "cases" that are later referred to in chapter 6.

In chapter 6, examples of the software examination for 2 typical measuring instruments and measuring systems are given:

- A simple stand-alone built-for-purpose instrument with a protective interface
- A complex PC-based, modular measuring system

The examples contain a description of the instruments, their legal and technical classification and an interpretation of the essential software requirements along with comments and additional information that may be useful for a uniform software examination. Moreover, the required software documentation is detailed. Because of the reasons mentioned above, the guide does not yet contain, however, complete collections of detailed measuring instrument specific requirements and examples.

Finally, chapter 7 contains a list of references and other literature which might be of help for the interested reader.

2 Terminology

2.1 Program code

Source code. Readable program code produced in human readable form, in general by the aid of a text editor. [14]

Executable code. Sequence of binary numbers that are read and interpreted by the central processing unit (CPU). It is only intelligible for a human reader if he uses tools like debuggers, disassemblers or re-compilers. A text editor is useless for this purpose. [14]

2.2 Legally Relevant Software

Software that realizes functions or properties of a legally controlled measuring instrument as defined in the MID, Article 1. The legally relevant software comprises program parts and data that form the software subject to legal control.

Legally relevant program parts

Parts of the program code that perform functions subject to legal control. Figure 2-1 shows the legally relevant parts of a program system realized as subroutines above the dividing line. Additionally there are subroutines that are not legally relevant below the line. The arrows show which subroutine is called by another (tip of the arrow) and which subroutine is calling. Instead of subroutines, the components of the program code can also be formed by complete executable programs that call each other via the operating system.

Notes:

a) *This software structure is not prescribed but it may offer advantages (see 4.3 and interpretation in 6.1.4 and 6.2.4). Furthermore the technical concept of software separation described here is state of the art in software engineering, also known as structured or modular programming or object oriented programming, and it is the inherent principle in most of the programming languages (like C/C++, Java, Delphi, Visual Basic, ...).*

b) *For conformity level "middle" (see section 4.3) the legally relevant program part may consist of a fixed part and other parts. These parts may have different legal software identifications.*

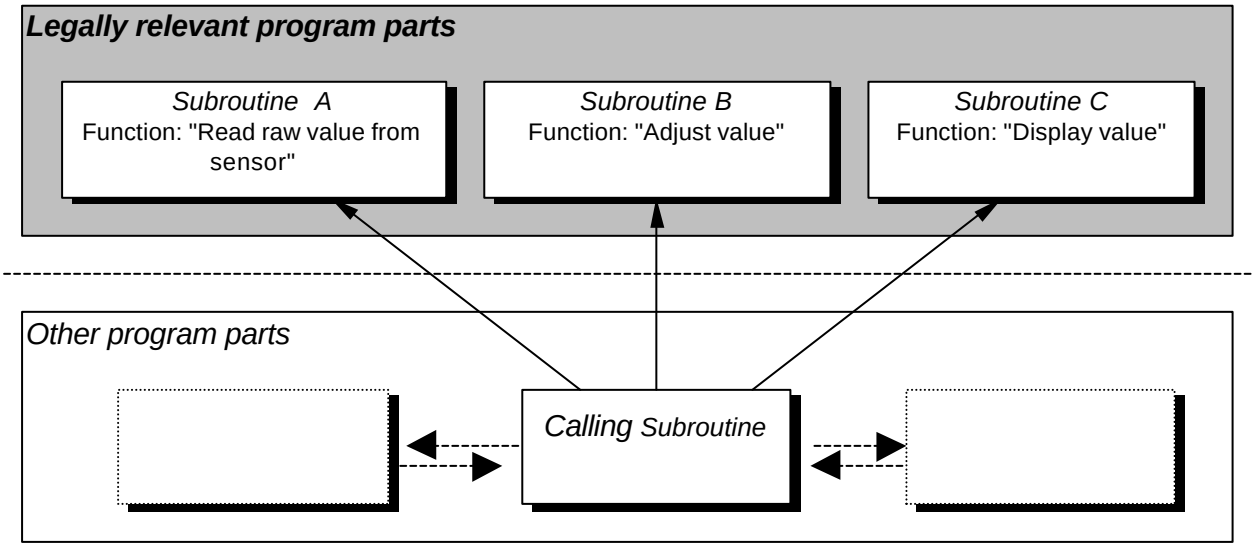


Figure 2-1: One example for legally relevant subroutines realizing legally relevant functions, and other program parts being separated

Legally relevant data

The legally relevant data can be distinguished into the following types of parameters and data:

- ***Type-specific parameters*** which depend on the special type of instrument only. Type-specific parameters are fixed at type approval of the instrument. In practice they are integrated into the program code.
- ***Device-specific parameters*** which depend on the individual device or instrument. Device-specific parameters comprise ***adjustment parameters*** (e.g. sensitivity, other adjustment or correction parameters) and other metrological parameters like ***configuration parameters*** for the measuring instrument (e.g. measuring range, scale division, units of measurement).

  Note: *Normally device-specific parameters have to be secured.*
- ***Settable parameters*** are manually entered data. They are allowed to be set or modified by the user.
- ***Variable values*** comprise the ***processed measurement values*** that are under control of legally relevant program parts (i.e. that are members of the data domain of such a program part) and ***final measurement values*** that often can be freely accessed by any software. Additionally there are ***auxiliary variables*** that e.g. contain ***commands*** for controlling the functions and the data flow of the legally relevant program parts, that realize ***counters*** for events etc.

Examples of legally relevant functions and data are given in Table 2-1.

Programs and subroutines usually have a *data domain*. A data domain consists of all variables and constants a program or subroutine can access by reading or writing to. Either the domain is owned by one program, subroutine or object alone and no other can write to it or even read it, or the data domain is shared with other program parts that all have read or write permission.

Figure 2-2 shows the data domain of a legally relevant program part (above the dividing line) and another domain with several variables that belong to other program parts. The parameters and variables mentioned above belong to the data domain of the legally relevant program part and write access to the variables by not legally relevant programs is impossible. Only the access to the import interface variable is not limited. The arrows show the dataflow from one data element to another.

Notes:

a) If software is designed to be separated according to Figure 2-1, then also the data domains must be separated into those subject to legal control and those that are not.

b) While data - e.g. final measurement values - are stored in files or transmitted, they are not within the data domain of a legally relevant program. (In this case the essential software requirements have to be interpreted in another way than in the case described above, if these data will be used for legal purposes (see 5.7 and example B, 6.2.4, ER2.2)).

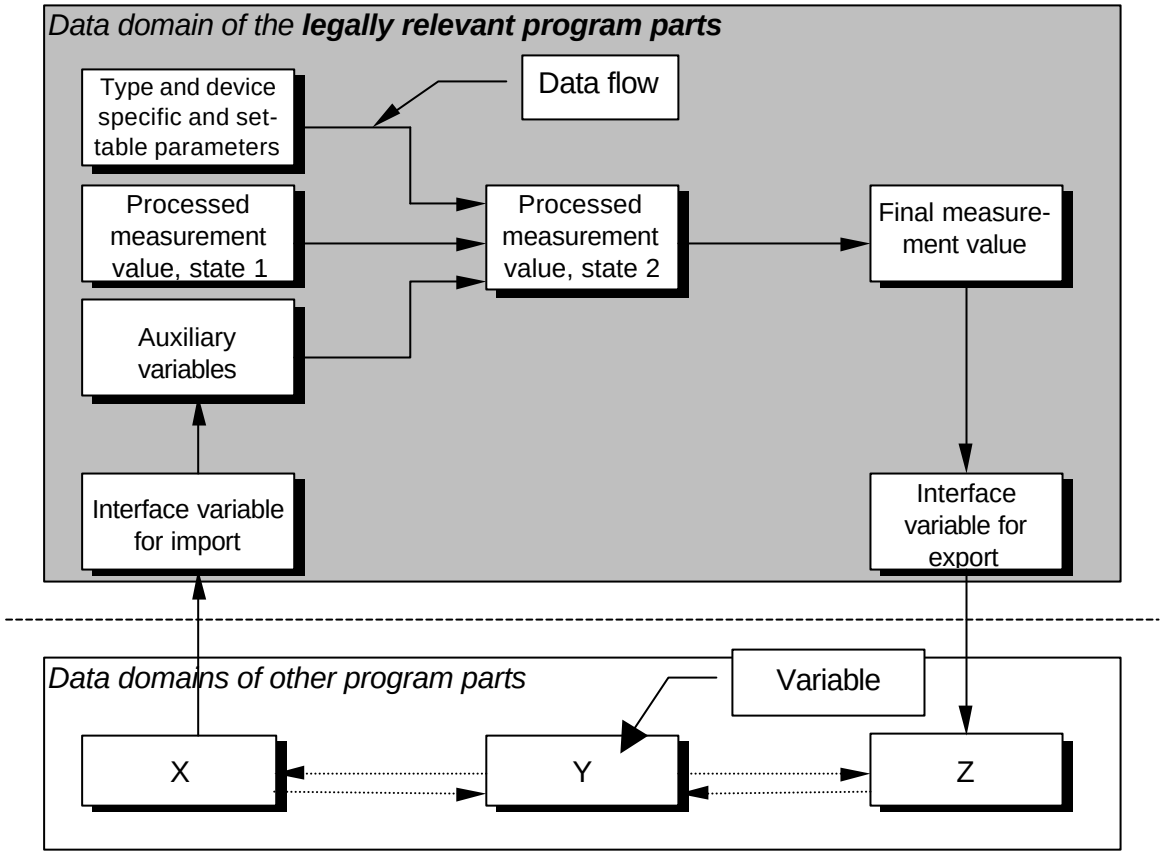


Figure 2-2: One example for a typical dataflow diagram of software with a separated legally relevant part with different types of parameters and variables

2.3 Changes of Software

2.3.1 Unintentional changes

Changes of program parts or data subject to legal control that happen by accidental physical or software effects (crashes, virus infections) or that are unintentionally performed by the user of an instrument.

2.3.2 Intentional changes (corruption, misuse) with simple common software tools

Changes with software tools and know-how commonly available to the general public. All kinds of text editors, for instance, are regarded as simple common software tools, whereas e.g. debuggers or disc editors are not.

2.3.3 Intentional changes (corruption, misuse) with special sophisticated software tools

Manipulation or simulation of the legally relevant software that is performed using software tools not available to the general public and that require special know-how. All kinds of e.g. debuggers, disc editors or software developing tools, for instance, are regarded as sophisticated software tools.

Legally relevant function	***Type specific parameter***	***Device specific parameter***	***Settable parameter***	***Variables***
Algorithm for calculating the final measurement value	Corrections for non-linearity	Sensitivity	Preset tare	Final measurement value as displayed
		Units of measurement		Intermediate measurement value
		Digital resolution, verification scale interval		
		Measurement range (Max and Min value)		
Stability analysis for a measurement value	Time constant			Status signals (e.g. zero indication, stability of equilibrium)
Counting impulses for cumulative measurements		Impulse factor		Counter variable
Calculating maximum	Length of the buffer	Length of the measurement period		Buffer for the values of all measurement periods
				Intermediate maximum value
Self checking routines	Nominal values for checking result[1])		Activation mode: on demand / cyclic	Flags (OK-FAIL)
Price calculation for direct sales to the public			Unit price	Price to pay
Rounding algorithm				Intermediate measurement value

Table 2-1: Examples of legally relevant functions, parameters and data.

2.4 Protection of Software

2.4.1 Protected software

Software, i.e. program code and data, a change of which either is not possible or is detected and made evident, e.g by sealing or an audit trail.

2.4.2 Audit trail

A software counter and/or information record of the changes to the device-specific parameters. An audit trail can be realized e.g. as an ‘Event counter’ or as an ‘Event logger’:

- **Event counter.** A non-resettable counter (legally relevant variable, see above) that increments once each time a special operational mode of the instrument is entered and one or more changes are made to device-specific parameters or other legally relevant data.

[1]) May be device specific in certain cases.

- **Event logger.** A file containing a series of records where each record contains data that describe the kind and time of an event e.g. a change to a device-specific parameter with an identification of the parameter that was changed, the time and date when the parameter was changed and the new value of the parameter. Program parts that realize event logging and files that contain the event data are regarded as legally relevant and have to be secured accordingly.

2.5 Interfaces

2.5.1 Hardware interface

Electrical input and/or output of a device for interchanging data or signals with other devices. These can either be instruments, modules (components) of an instrument or peripheral devices.

The term 'interface' comprises all mechanical, electrical and logic properties at the data interchange point and the meaning of the transmitted data and instructions [5] (ISO 7498).

2.5.2 Protective interface

An interface is defined as being protective

- if only a defined set of allowed parameters, data and functions of the legally relevant software part can be influenced or released via this interface

  and

- if it is not possible to introduce into an instrument (or module of an instrument) instructions or data intended or suitable to:
  - display data that are not clearly defined and could be mistaken for a measurement result
  - falsify displayed, processed or stored measurement results or other legally relevant data (e.g. unit price, price-to-pay, unit of measurement in case of direct sales to the public).
  - adjust or configure the instrument or inadmissibly change any type or device specific parameter
  - corrupt the legally relevant program code of the instrument.

2.5.3 Software interface

If parts of software exist besides the legally relevant parts, these parts may be separated in a sense that they communicate via a software interface. Communicating software parts interchange data via certain variables (or files) that they can both access (read or write to). These interface variables and the program code that writes the data to or reads the data from the interface variables form the software interface. (The interface variables correspond to the lines of a hardware interface).

Interface variables can be realized as e.g. global program variables, as function parameters or as data files.

2.5.4 Protective software interface

A software interface between the legally relevant software part and other software parts consisting of variables or data files is defined as being protective

- if only a defined set of allowed parameters, data and functions of the legally relevant software part can be influenced or released via this interface and
- if both parts exchange information only via this interface, i.e. not via any other link.

Variables and program code of a protective software interface are part of the legally relevant software.

2.6 Data security

Authenticated program. Program code that is trusted by the user and customer (both parties involved) to be identical with the approved one. It is either supplied by someone who is authorized and who is responsible that the program code is identical (or conforms) with the approved one OR its identity / conformity with the approved one is (legally) verified.

Authenticated data. Transmitted data in a complex measuring system the origin of which can be verified by the receiver

OR

in the case of measurement values stored in a memory with public access for later use: values that can be clearly assigned to a certain measurement.

Authentification method. Method to enable everyone involved to verify that programs or data are authentic.
Example: Generation of an electronic signature for the relevant data or files by an authenticated program before storage or transmission. On receiving or reading: Recalculation of the electronic signature and comparison of the result with the nominal value with an authenticated program used by a person involved.

Checksum. Addition of all bytes of a program code or a data set. A modulo addition is often used in order to get a result with a fixed number of figures.

Here a checksum is often used as a simple hash code. A hash code is the result of an arithmetic combination of all bytes of a program code or a data set. The result of the hashing algorithm comprises only some bytes, and the algorithm is such that any modification of the program code or data with a high probability leads to another result.

(**Electronic**) **signature.** A signature of a file (program code or data) is generated in two steps: firstly a hash code is calculated (see "checksum" above) and secondly the hash code is encrypted.[2])

The signature is normally added to the program code or data set it has been generated from.

Software identification. Method to verify the authenticity and integrity of a software.

OR

A number or a string of characters that is assigned to a certain reference software.[3])

Legal software identification. Software identification that is assigned to the legally relevant software.[4])

One acceptable technical solution for a legal software identification is the "ABC method" that consists of three parts:

- *Part A* is the code fixed by the manufacturer of the instrument. This part indicates, under his responsibility, each legally relevant change in software.
- *Part B* is formed by an algorithm which is part of the legally relevant software and which forms a number which will automatically change if there has been a change in the device specific parameters.
- *Part C* in the same way as part B but now over the program code which is covered by the actual identification code ABC.

Software integrity. The software is identical with a correct reference version (e.g. the approved one); it has not been modified intentionally or unintentionally.

---

[2]) Here in certain cases a simple all-in-one solution for hashing and encrypting is accepted as a technical solution: the "cyclical redundancy check" CRC [11,12] with a secret start value.

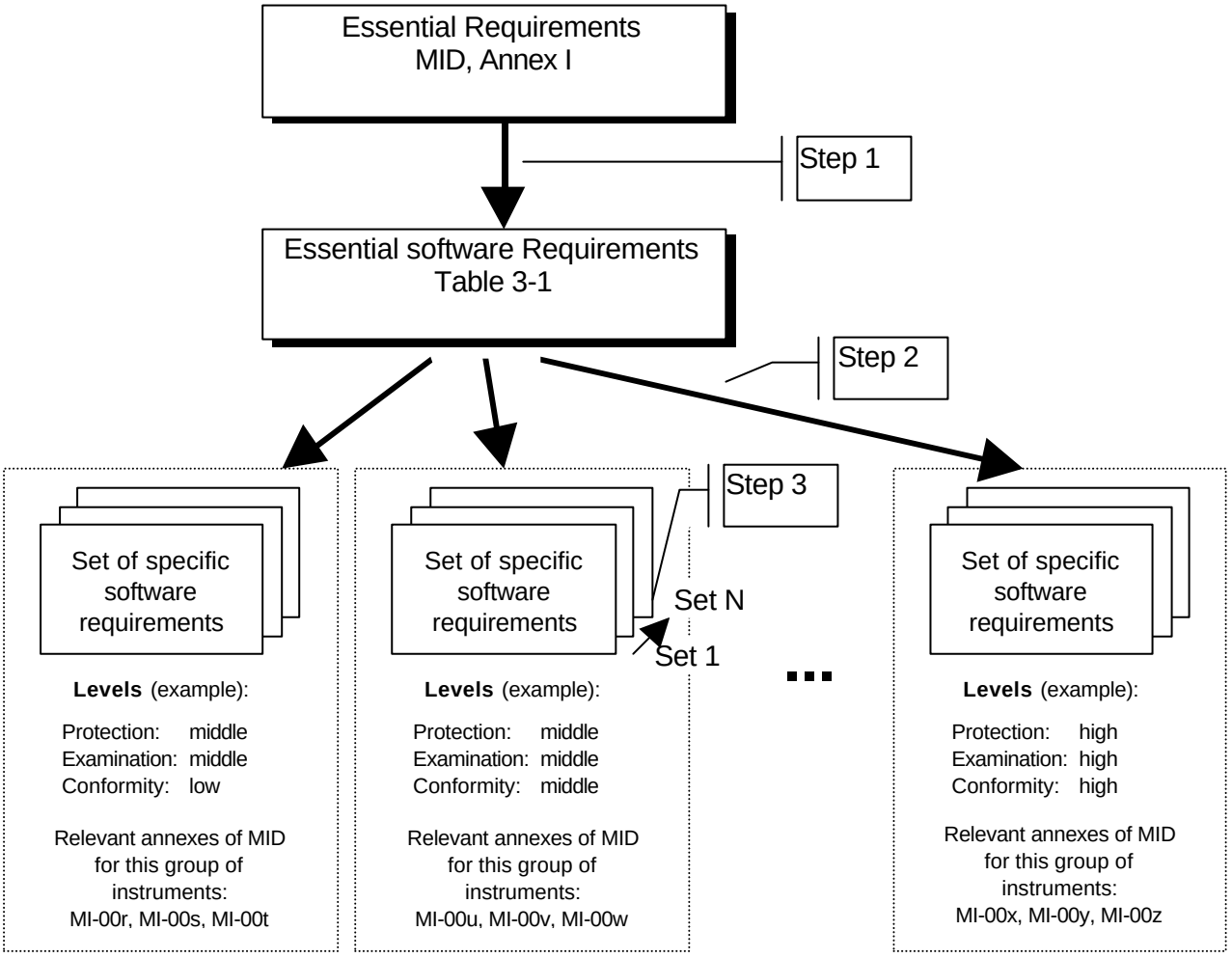
[3]) This can be the version number of the software.

[4]) The identification may be separated into several parts.

3 Essential Software Requirements

The basis of this guide is the European Measuring Instruments Directive MID [1]. Annex I of this directive contains essential requirements that have been interpreted with regard to the *software* of measuring instruments. The result of this interpretation is the 5 subjects with 11 *essential software requirements* that are listed in Table 3-1. These requirements have a very general scope and for real practical use they have to be more detailed. On the other hand there are many different fields of application and many possible technical solutions for measuring instruments. To avoid defining a great number of detailed requirements that only apply to some special technical solution, and that make no sense or lead to confusion for most of the other applications, a stepwise approach for a system of tailored software requirements has been chosen.

The first step is shown in this chapter: the derivation of the essential software requirements from the requirements of the MID (see Figure 3-1). The next steps are explained in chapters 4 and 5.



Step 1: Deriving Essential Software Requirements from the MID.
Step 2: Defining groups of measuring instruments for which the *same levels* of the three criteria protection, examination and conformity can be chosen.
Step 3: Interpret the essential software requirements for each group and define sets 1 to N of specific software requirements taking into consideration the *technical features*.

Figure 3-1: Derivation of specific software requirements from the MID (see chapters 4 and 5)

Requirement number	***Essential Software Requirements***[5]***)***	Reference to MID Annex I [6])
	Software Design and Structure	
ER1.1	The software of a measuring instrument shall be designed so as to allow ready evaluation of the conformity of its legally relevant functions to the requirements of this guide.	27
ER1.2	The legally relevant software shall be designed in such a way that it is not inadmissibly influenced by other software.	7, 11, 13
ER1.3	The legally relevant software shall be designed in such a way that it is not inadmissibly influenced via the interfaces of the device.	7,11
	Software Protection	
ER2.1	Legally relevant programs and data shall be protected against accidental or unintentional changes.	11, 13, 14, 15, 26
ER2.2	Legally relevant programs and data shall be protected against corruption or intentional changes by unauthorized persons.	11, 12, 13, 14, 15, 26
ER2.3	Only the approved and verified software is allowed to be used for legal purposes. It shall be clear and unambiguous that a presentation of a result is generated by a legally relevant program.	7, 8, 18, 21, 23
ER2.4	Functional defects that can falsify measurement values in software controlled hardware shall be detected as far as possible. When detected they shall be acted upon.	1, 6
	Software Conformity[7])	
ER3.1	The software shall not inadmissibly be modified after type approval.	13, 27
ER3.2	For the verification of conformity an identification of the legally relevant software and suitable instructions shall be available.	16, 17, 27
	Testability	
ER4.1	The functionality of the instrument shall be testable. ***Note:*** *Testability means that it is possible to verify the conformity of the instrument with the requirements of the MID and of this guide.*	7, 27
	Documentation for Type Approval	
ER5.1	The legally relevant software, including its hardware and software environment, shall be suitably documented.	Annex IV

Table 3-1: Essential software requirements

[5]) ***Note:*** These requirements cover some features of the hardware of the measuring system, too.

[6]) ***References to*** **MID/3 Annex I (Essential Requirements):**

(No. 1)	*Allowable Errors*
(No. 6)	*Reliability*
(No. 7, 8)	*Suitability*
(No. 11, 12, 13, 14, 15)	*Protection against corruption*
(No. 16, 17, 18, 21)	*Information to be borne by and to accompany the instrument*
(No. 23)	*Indication of result*
(No. 26)	*Further processing of data to conclude the trading transaction*
(No. 27)	*Conformity evaluation*

[7]) ***Note:*** Here the conformity with the approved pattern (software) or with applicable requirements is meant.

4 Definition of Levels

It has been the experience in type approval practice that different kinds of measuring instruments are not treated equally within a country and that the same kind of instrument is treated differently in different countries without an obvious objective reason. It was tried by the working group to identify the facts or criteria that lead to the different evaluation of the instruments in type approval. For these facts or criteria three levels have been defined and the working group will chose and fix (as a proposal) one level of each fact for a certain kind of instrument or a certain field of application and by that will harmonize the type approval.

The facts and criteria that have an impact on the different treatment of instruments in the sense described above are:

- The strength of ***protection*** of the software against changes
- The intensity of ***examination*** of the software at type approval
- The degree of ***conformity*** between the software implemented in a verified instrument and the approved software

In this chapter (4) the levels for these facts and criteria are defined and in annex I the different kinds of instruments according to the annexes of the MID are grouped to relevant fields of application and the levels are assigned to each group as a guiding, non-binding principle.

The benefit of defining and fixing levels is that now a comprehensible and well-founded interpretation of the essential software requirements is possible. This is the second step in Figure 3-1. In chapter 6 two examples for the interpretation are described in detail.

Besides the described facts and criteria there is another aspect that has to be taken into consideration: the technical features of the measuring system. Depending on these features the essential software requirements have to be interpreted in different depth and ways. This is the third step in developing specific software requirements shown in Figure 3-1. The classification of an instrument according to its technical features is discussed in chapter 5, and in chapter 6 an attempt is made to demonstrate this by two examples.

Notes:

a) Steps 2 and 3 are not performed completely. This is future work for specialists in the various kinds of measuring instruments.

b) Though only levels for two subjects of the essential software requirements (chapter 3) are defined, it turns out that the interpretation of essential requirements of the other subjects has to be interpreted according to these levels. E.g. for a high level of protection it can be necessary to interpret the requirement on "Software design and structure" in a way that it is not possible to realize an open system.

4.1 Software Protection Level

The software protection means adequate measures against accidental or intentional corruption. The software protection level has an impact on the technical solution and therefore mainly addresses the manufacturer, ie. the software developer. The definition of the protection levels gives an answer to the questions:

- How strong must the protection against misuse of the instrument be?
- Which tools used by the intruder can be expected?

The definitions of the protection levels are:

Low: There is no protection of the software against intentional changes required.

Middle: The legally relevant software is protected against intentional changes with simple common software tools (text editors).

High: The legally relevant software is protected against intentional changes with special sophisticated software tools (debuggers and hard disc editors, software developing tools) i.e. protection level according to the state of the art in data security like e.g. for financial transactions.

Notes:

a) In the following the definition of different protection levels only applies to protection against intentional changes. As for unintentional changes no levels are defined and the essential software requirements are interpreted and the instrument is tested according to the state of the art.

b) If it is to his advantage, the manufacturer is free to fulfil the requirements of the higher protection level rather than the assigned one.

c) The customary method of securing/sealing, making an inadmissible intervention evident, is equivalent to software protection means for the levels middle and high.

4.2 Software Examination Level (Type Examination or Design Examination)

The software examination level mainly addresses the Notified Body responsible for type approval. The definition of the examination levels gives an answer to the questions:

- What resources have to be employed for the examination?
- Which kind of tests have to be performed?
- What size of documentation of the instrument is necessary for the examination?
- What are the consequences for the applicant?

The definitions of the examination levels are:

Low: The software functions are verified by a normal type examination test. The documentation of legally relevant parts and functions of the instrument supplied by the manufacturer is needed mainly to understand the usage of the instrument and to be able to test it. Emphasis is laid on the *results* of the practical metrological examination test and the test of the handling.

For some technical features that are not covered by metrological examination tests (e.g. the protectiveness of interfaces) a declaration by the manufacturer is accepted that the software controlling the measuring instrument to be type approved does fully comply with the documentation supplied and that there are no functions other than the documented ones.

Consequence for the applicant: The operating manual and only a technical documentation that the manufacturer supplies, with no special software documentation, are required as necessary for the normal type examination.

Middle: In addition to the normal type examination tests (see *"Low"*) the software is examined on the basis of a description of the software functions supplied by the manufacturer. It is verified whether the documented functions are complete and consistent.

For PC-based instruments or open measuring systems with possible user access, practical tests (spot checks) with the program are conducted in order to check e.g., whether all protection measures are effective and whether commands and the identification of the legally relevant software operate as documented.

Consequence for the applicant: The documentation submitted for type examination shall also cover the software.[8])

High: In addition to the metrological tests and test of the handling of the system (see "Low" and "Middle") the legally relevant software is tested using its source code. The subject of the code examination can be e.g. the realization of an algorithm, the filtering of the input via an interface or whether the software separation is realized correctly.

---

[8]) Examples are given in sections 6.1.4 and 6.2.4 (see ER5.1 "Comments on examination level„)

Consequence for the applicant: The source code of the software has to be examined.

Notes:

a) *The level is only addressing the depth of the software examination. In any case the metrological properties of the instrument are evaluated by conducting the normal metrological performance test.*

b) *If it is advantageous, the manufacturer and notified body can agree on a higher examination level rather than the assigned one.*

4.3 Degree of Software Conformity

The degree of software conformity and the capability of the software of being checked at verification are of importance for all parties involved, i.e. for the manufacturer, the Notified Body responsible for type approval and the appropriate authorities.

It is a problem of industrial production of measuring systems subject to legal control to keep the relevant and approved features of the product unchanged during its lifecycle. On the one hand demands arise in the course of time to correct or improve the product or fit it to given facts. On the other hand an approval is based on the precondition that properties remain constant. The following definition of three levels for the conformity of specimen and pattern tries to cover this spectrum. Conformity in this sense comprises the aspects:

- Which modifications are allowed after type approval or design examination without additional approval?
- Which modifications have to be announced to the notified body or design examiner by the applicant?
- How can the conformity be checked?
- Is it necessary to deposit the approved version of the software?

The definitions of the conformity levels are:

Low: The implemented software of each individual instrument is *in conformity* with the approved documentation. Regardless of minor corrections of the source code the functionality remains identical to this documentation:

- Modifications of the legally relevant software are allowed as long as the documented functions and characteristics of the approved instrument remain unchanged. The NB[9]) must, however, be informed. Changes of *documented* functions and characteristics require additional approval by the NB and a new legal software identification.
- Modifications of the part not subject to legal control are allowed without informing the NB as long as the software separation is observed and exclusively the approved software interface is used.
- At verification the conformity with the approved software is checked by a legal software identification that is mentioned in the type approval certificate.
- The approved software documentation is kept under control of the NB. Additionally the complete program code (executable code) of the measuring instrument may exceptionally be deposited.
- Detailed documentation of the operating system is not necessarily required.

[9]) NB - Notified Body or design examiner

Middle: The implemented software of each individual instrument is *in conformity* with the approved documentation. Regardless of minor corrections of the source code the functionality remains identical to this documentation. In special cases depending for instance on the technical features (see e.g. section 5.8, case (z), Complex measurement), a part of the legally relevant software is defined and fixed at type approval, that shall be *identical* to the implemented software of each individual instrument[10]):

- Any modification of the *fixed* legally relevant software part automatically leads to a new legal software identification. The NB grants an additional approval in that case.
- For the other parts of the legally relevant software, modifications are allowed as long as the documented functions and characteristics of the approved instrument remain unchanged. The NB must, however, be informed. Changes of *documented* functions and characteristics require additional approval by the NB and a new legal software identification.
- Modifications of the part not subject to legal control are allowed without informing the NB as long as the software separation is observed and exclusively the approved software interface is used.
- At verification the conformity with the approved software is checked by a legal software identification (signature) that is mentioned in the type approval certificate. The identification may contain a part that is calculated from the fixed legally relevant executable code.
- The approved software documentation and the complete program code (executable code) of the measuring instrument are kept under control of the NB in an appropriate form e.g. paper or electronic media.
- In general, detailed documentation of the operating system is not required.

High: The *entire software* of each individual instrument *is identical* to the approved software:

- Because of the identity, modifications of any part of the software automatically lead to a new legal software identification. The NB gives an additional approval in this case.
- At verification the conformity with the approved software is checked by a legal software identification (signature) that is mentioned in the type approval certificate.
- The approved software documentation and the complete program code (executable code) of the measuring instrument are kept under control of the NB.
- The software environment shall not be changed.

Note: *If it is to his advantage, the manufacturer is free to fulfil the requirements of the higher conformity level rather than the assigned one.*

[10]) Reasons for fixing a software part could be for example: During the approval examination the instrument has been tested by expensive or extensive measurements that the applicant is not able to repeat by himself after the modification of the software.

5 Technical Features of Measuring Instruments and Systems

This guide is intended to be applicable to all kinds of measuring instruments. As for the requirements in chapter 3 this is true because of their common definition. In practice more detailed requirements are necessary and the essential software requirements need further interpretation depending on the hardware and software configuration of the measuring instrument or system to be type approved.

In chapter 4, levels for three criteria have been defined that have to or will be fixed. Unlike this, it is not necessary to define and conclude levels for the technical features as they can be observed and classified objectively. In the following a system of several "cases" is proposed for classifying the hardware configuration and the software features assigned by the manufacturer. The various "cases" are referred to in chapter 6 where sets of specific requirements are formulated (see also third step in Figure 3-1). Only two sets are shown in section 6.1 and 6.2 but these are rather typical ones and can serve as examples for further requirement sets.

Note: *Some technical features described below may not be acceptable for certain measuring instruments or legal fields of application, respectively. Acceptable features will be selected later in appendices specific for the various measuring instruments (to be published later). Here the all relevant sets of specific requirements will be contained.*

Note: *Theoretically a great number of sets of specific requirements are possible. However, in practice the number of really different technical configurations is much smaller. Most of the simple configurations (see 6.1) are very similar to each other and therefore only a small number of sets of specific requirements is needed.*

5.1 Hardware Configuration

The variability of the hardware of measuring systems is represented by 5 basic configuration models, cases (a) to (e), see Figure 5-1. The modules or devices shown in this figure can be realized as built-for-purpose devices - normally cases (a) to (d) - or as non-built-for-purpose devices - normally case (e) - the latter may be personal computers, workstations or even mainframes.

5.2 User Interface (Shell)

The user shell consists of input media (e.g. keyboard, mouse) and output media (e.g. display, video monitor or printer).

(f) User shell always in operating mode subject to legal control.

(g) User shell can be switched from operating mode subject to control to operating mode not subject to control and vice versa.
(The user may, for instance, stop the measuring program, start a text processor and then start the measuring program again.)

(h) Free user shell with operating modes subject to control and operating modes not subject to control in parallel.
(There is, for instance, one window in a windows operating system that represents the user interface subject to control.)

5.3 Software Loading

(i) No loading possible, programs are invariable (firmware, usually stored in a non-volatile memory, e.g. in a non-detachable, soldered EPROM).

(j) The manufacturer fixes all of the programs subject to control and all of those not subject to control that are loadable. Loading can be realized by changeable storages (CD-ROM, etc) or by downloading via interface from a server (to hard disc drive, Flash ROM, EEPROM etc).

(k) Any program can be loaded. Loading can be realized by changeable storages (floppy disc, CD-ROM etc) or by downloading via interface from a server (to hard disc drive, Flash ROM, EEPROM etc).

5.4 Software Structure

(l) The software is subject to legal control as a whole and is not intended to be modified after approval.

(m) Parts of the software are subject to legal control. Other parts that are not legally relevant are intended to be modified after approval.

See Figure 2-1 and Figure 2-2.

5.5 Software Environment

(o) The software environment is invariable. The whole of the instrument's software has been constructed for the measuring purpose.

(p) The software subject to control is embedded into an environment like a standard operating system that is not especially constructed for the measuring purpose.

5.6 Fault Detection

(q) The presence of a defect is obvious or can simply be checked or there are hardware means for fault detection.

(r) The presence of a defect is not obvious and cannot be easily and simply checked using devices apart from the instrument itself and there are no hardware means for fault detection.

5.7 Long-term Storage of Measurement Values

(s) No long-term data storage of measurement values in the system.

(t) Measurement values are stored in the system for later legal use.

5.8 Measuring principle

5.8.1 Time Dependence

(u) Cumulative measurement (e.g. counter, fuel dispenser)

(v) Single independent measurement

5.8.2 Repeatability

(w) Repeatable measurement

(x) Non-repeatable measurement

5.8.3 Complexity

(y) Simple, straightforward, or static measurement

(z) Complex or dynamic measurement

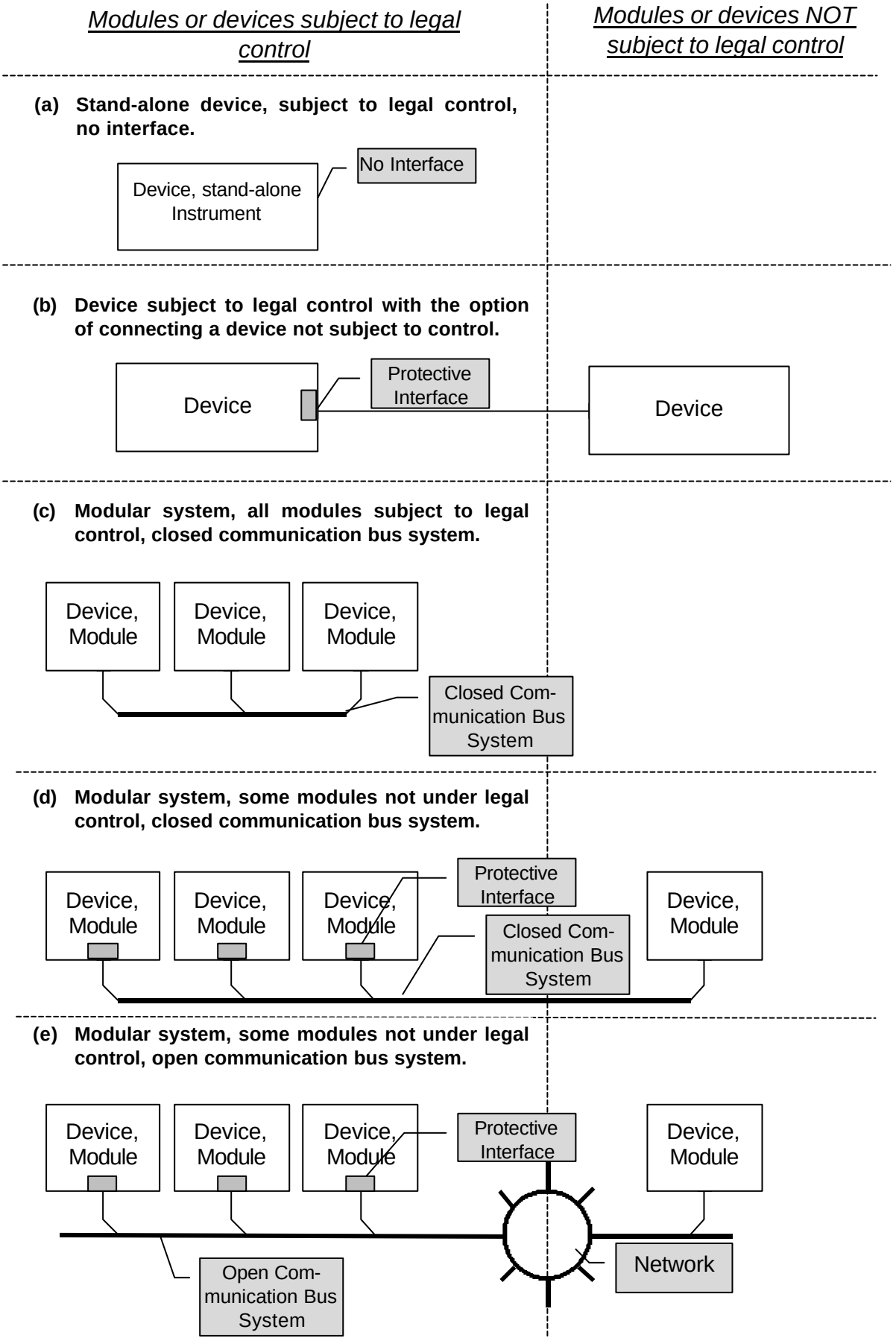


Figure 5-1: Example of possible hardware configurations of microprocessor-controlled or PC-based measuring instruments and systems

6 Interpretation of the Essential Software Requirements for Selected Measuring Instruments and Systems

The concept of the draft is a stepwise approach: After applying the levels of protection, examination and conformity to the kinds of measuring instruments and their fields of application (see chapter 4), for each field of application and measuring instrument sets of specific requirements have to be defined that take into consideration the various technical properties of instruments. At each examination the right set of specific requirements has to be chosen by the examiner depending on the technical features of the instrument that is to be examined.

To illustrate what is meant by the interpretation of the Essential Software Requirements (see 3, abbreviated ***ER***), two examples of technical realizations of measuring systems are discussed in the following. This is not a substitute for a complete set of specific annexes for each kind of measuring instrument that have to be published later. However, it is intended by the choice of these examples to already cover a big part of the spectrum of possible technical solutions.

In this section the software of the example systems is classified according to chapters 4 and 5, i.e. the technical impacts and the non-technical conditions for the interpretation of the essential software requirements are demonstrated (software classification). With the non-technical conditions the levels of protection, examination and conformity are meant (see 4.1 to 4.3).

The examples are

A) a simple measuring stand-alone instrument, realized as a built-for-purpose device with all components within a housing and

B) a complex PC-based measuring system with various components connected by a network.

Note: *Up to now the kinds of measuring instruments and their fields of application have not been assigned to certain levels of the non-technical conditions (protection level, examination level, level of conformity). Therefore* ***all*** *levels are discussed in the following even if the interpretation of a certain general requirement under some conditions is only hypothetical. By doing so it will be easier to define the levels for each kind of instrument and field of application.*

6.1 Example A: Simple Stand-Alone Measuring Instrument

In principle this example stands for a broad variety of instruments used for commercial transactions like fueling points, taximeters etc.

6.1.1 Description of the Instrument

Let the simple stand-alone measuring instrument be a built-for-purpose device. The instrument is characterized by the following general technical features (Figure 6-1):

- *Closed housing. All components of the instrument are within the housing; sealing possible.*
- *The instrument consists of a sensor (transducer, including analogue electronics), further analogue components (e.g. A/D converter), a microprocessor board and an LC display.*
- *The device has a hardware interface that is intended for connecting a peripheral device not subject to legal control.*
- *The software is stored in a non-volatile memory (*non-detachable *Flash ROM, EEPROM, EPROM or PROM).*
- *The entire software is not intended to be changed after type approval. There is no software separation of legally relevant program parts and other parts realized.*
- *Fault detection: checksum calculation over the memory contents.*

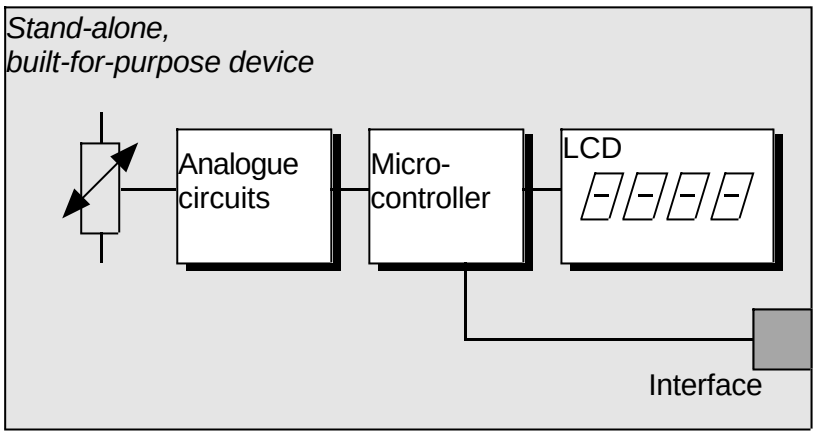


Figure 6-1: *Example A* - Simple Stand-Alone Measuring Instrument

6.1.2 Legal Classification

Normally the legal classification would follow Table A-1 in Annex 1. As the levels have not yet been selected, in the following interpretations ***all*** levels for "protection, examination and conformity" are taken into consideration.

Note: To make the interpretations and comments comparable to example B (see 6.2) the following cases have been selected according to the proposal for the category of "Measuring Instruments used for Commercial Transactions" (see Table A-1):

Software protection level: ***middle***

Software examination level: ***middle***

Degree of software conformity: ***low***

These levels and the resulting interpretations are marked with grey background in section 6.1.4 like this note.

6.1.3 Technical Classification

According to chapter 5 the instrument can be classified as follows:

Feature	Case	Explanation
Hardware Configuration	*b*	Device subject to legal control with the option of connecting a device not subject to control.
User Interface (Shell)	*f*	The user shell is always in operating mode subject to legal control.
Software loading	*i*	No loading possible, programs are invariable (firmware, usually stored in non-volatile memory).
Software structure	*l*	The entire software is subject to legal control and is not intended to be modified after type approval.
Software environment	*o*	The software environment is invariable. The whole of the instrument's software has been designed for the measuring purpose.
Fault detection	*r*	The presence of a defect is not obvious and cannot be easily and simply checked using devices apart from the instrument itself and there are no hardware means for fault detection.
Long-term storage of Measurement values	*s*	No long-term storage of measurement values in the system intended.
Measuring principle	*v, w, y*	Single independent, repeatable, simple and static measurement

In the following sections the cases are simply abbreviated by, for instance, (*b*).

6.1.4 Interpretation of the Essential Software Requirements

ER1.1: The software of a measuring instrument shall be designed so as to allow ready evaluation of its conformity to the requirements of this guide.

In this example of a simple stand-alone instrument, the manufacturer of the software does not intend to change the software after type approval (l)[l] . In this case the design or structure of the software (whether separated or not) is not important for the objectives of the type examination. The ER1.1 needs no further interpretation.

Comments on examination level:

As the design and structure of the software of this example system is not relevant for the examinations discussed in the following, there is no need to examine it in any level.

[l]) The entire software is subject to legal control and is not intended to be modified after type approval.

ER1.2: The legally relevant software shall be designed in such a way that it is not inadmissibly influenced by other software.

This requirement is met independently of the software structure, because no other software exists besides the legally relevant software in the instrument (I, o)[o] and the software cannot be loaded (i)[i].

Comment on examination level:

As no other software exists besides the legally relevant software, it is supposed that the normal metrological examination tests of the instrument are sufficient to judge the software and no additional software examination concerning the structure of the software is necessary even if levels *middle* or *high* are stipulated.

Comment on conformity level:

The software structure has an impact on the reliability of the conformity during the lifecycle of the software. For the simple configuration of this example (i, I, o), ER1.2 needs no further interpretation. However, ER3.1 must be taken into consideration.

ER1.3: The legally relevant software shall be designed in such a way that it is not inadmissibly influenced via the interfaces of the device.

In this example the device has an interface (b)[b], and any device not subject to legal control may be connected. If it can be proven that the interface is protective, it doesn't need to be sealed.

Comments on protection level:

Low: The interface doesn't need to be sealed, even if it is not proven to be protective.

Comments on examination level:

Low: The manufacturer declares that the interface is protective, i.e. that neither the measurement values nor the functions of the instrument can be influenced by commands or data transmitted to the instrument via the interface. No special test of the interface software is performed in this case.

Middle: The manufacturer supplies a complete description of commands and parameters received via the protective interface, including a declaration of completeness of this description.

It has to be verified in the examination, on the basis of this documentation, that all the data received via the interface do not inadmissibly influence the measuring instrument.

High: It has to be verified on the basis of the source code that all the data received via the interface do not inadmissibly influence the measuring instrument.

ER2.1: Legally relevant programs and data shall be protected against accidental or unintentional changes.

There are two reasons for inadmissible changes: physical effects and false handling by the user. If this instrument is tested according to the regulations in question (EMC, temperature, humidity etc), accidental changes of data or programs need not be taken into consideration. As for false handling by the user, in this example the user interface is always in an operating mode subject to legal control and the software subject to control is isolated (f, I, o)[f] . Unintentional changes could only happen by inadmissible properties of the software subject to control (e.g. it must not be possible to change device specific parameters unintentionally).

---

[o]) The software environment is invariable. The whole of the instrument's software has been designed for the measuring purpose.

[i]) No loading possible, programs are invariable (firmware, usually stored in non-volatile memory).

[b]) Device subject to legal control with the option of connecting a device not subject to control.

[f]) The user shell is always in operating mode subject to legal control.

Comments on protection level:

Protection level in the sense used here refers to *intentional* manipulations (see ER2.2).

Comments on examination level:

Low: The handling of the instrument has to be tested practically with help of the operating manual.

Middle: The correctness and consistency of the instrument's handling is analysed on the basis of the documentation (operating manual and special software documentation) in addition to the practical tests.

High: The source code of the software has to be tested to check whether malfunctions because of false handling are possible additionally to the tests mentioned above.

ER2.2: Legally relevant programs and data shall be protected against corruption or intentional changes by unauthorized persons.

As the user interface is always in an operating mode subject to legal control and the software subject to control is isolated (f, l, o), *intentional* changes could only happen by inadmissible properties of the software itself (e.g. it must not be possible for the user to change device specific parameters).

Comments on protection level:

Low: No special protection measures against corruption are required.

Middle/high: Either the housing of the instrument has to be secured, or the program and data memory must be secured against unauthorized removal.

Comments on examination level:

Low: All operations have to be practically tested on the basis of the operating manual: no data and no program must be changeable via the user interface.

Middle: In addition to the test mentioned above, all protection measures mentioned in the documentation have to be tested practically to check whether they function as documented.

High: In addition to the tests mentioned above, the software of the user shell has to be analysed to check whether only the defined set of operations is possible and all other handlings are blocked by the software.

ER2.3: Only the approved and verified software is allowed to be used for legal purposes. It shall be clear and unambiguous that a presentation of a result is generated by a legally relevant program.

The instrument in this example is a built-for-purpose device that has restrictive technical features (f, i, l, o). It is technically not possible to change the operating mode. Therefore the presentation of measurement values and other functions can be easily marked unambiguously as legally relevant by seals, verification marks or imprints.

Comments on protection level:

Middle/high: Program and data memory must be protected against unauthorized removal.

Comments on conformity level:

Low: The manufacturer is allowed to correct the program code without changing the legal software identification. As far as legally relevant software parts are concerned, the Notified Body must, however, be informed in any case. At verification the appropriate authority or a responsible person checks by the legal software identification that the software implemented in the instrument is in ***conformity*** with the approved software.

Middle: As the instrument performs simple, straightforward measurements, the same as for level low applies.

High: The manufacturer implements exactly the same software in each individual instrument without any modification. At verification the appropriate authority or a responsible person checks by the

legal software identification (signature) that the software of the instrument is ***identical*** with the approved software.

The user can rely on the ***verification mark*** that the presentation of the measurement values is generated by the approved program.

ER2.4: Functional defects that can falsify measurement values in software controlled hardware shall be detected and acted upon.

In the example some kinds of functional defects are detected and the software realizes the appropriate reaction (r)[r] .

Comments on examination level:

Low: The instrument is tested practically with the help of the operating manual. As functional defects happen rather seldomly, the failure detection mechanism normally isn't tested.

Middle: The failure detection mechanism described in the documentation is checked by simulating suitable failures.

High: The failure detection mechanism is tested as in case middle. Additionally, other failures are simulated and the reaction of the instrument is judged.

ER3.1: The software shall not inadmissibly be modified after type approval.

What kind of modifications are admissible depends on the level of the required conformity level:

Comments on conformity levels:

Low: The implemented software of each individual instrument is in conformity with the approved documentation. Regardless of minor corrections of the source code the functionality remains identical to the technical documentation:

- Modifications of the software are allowed as long as the documented functions and characteristics of the approved instrument remain unchanged. The NB must, however, be informed. Changes of documented functions and characteristics require additional approval by the NB and a new legal software identification.
- At verification the conformity with the approved software is checked by a legal software identification that is mentioned in the type approval certificate.
- The approved software documentation is kept at the NB. Additionally the complete program code (executable code) of the measuring instrument may exceptionally be deposited.

Middle: For a built-for-purpose device with a simple, straightforward measurement principle (y) and where the entire software is subject to control as in this example (f, i, l, o), the same as for level *low* applies.

High: The entire software of each individual instrument is identical to the approved software:

- Because of the identity, modifications of any part of the software automatically lead to a new legal software identification. The NB gives an additional approval in this case.
- At verification the conformity with the approved software is checked by a legal software identification (signature) that is mentioned in the type approval certificate.
- The approved software documentation and the complete program code (executable code) of the measuring instrument are kept at the NB.

[r]) The presence of a defect is not obvious and cannot be easily and simply checked using devices apart from the instrument itself and there are no hardware means for fault detection.

ER3.2: For the verification of conformity an identification of the legally relevant software and suitable instructions shall be available.

It depends on the level of the required conformity level how the conformity of the individual instrument is checked:

Comments on conformity levels:

Low: The implemented software of each individual instrument is in conformity with the approved documentation. Regardless of minor corrections of the source code the functionality remains identical to the technical documentation:

- At verification the conformity with the approved software is checked by a legal software identification that is mentioned in the type approval certificate. The legal software identification may be displayed either on demand or automatically on start up or cyclically).

Middle: For a built-for-purpose device with a simple, straightforward measurement principle (y) and where the entire software is subject to control as in this example (f, i, l, o), the same as for level *low* applies.

High: The entire software of each individual instrument is identical to the approved software:

- At verification the conformity with the approved software is checked by a legal software identification (signature) that is mentioned in the type approval certificate.

ER4.1: The functionality of the instrument shall be testable.

As for the metrological parts of the software, this requirement is met because the normal metrological performance test of the complete instrument and of its functions is possible.

Comments on examination level:

Low: Only the metrological parts of the instrument's software are tested by the normal practical examination test. Other features of the software that are not covered by these tests don't need to be made testable by the manufacturer. It is sufficient that he declares that these untested features conform with the requirements (protectiveness of an interface, failure detection and reaction etc.).

Middle: In addition to the normal type examination tests (see *"Low"*) the software is examined on the basis of a description of the software functions supplied by the manufacturer. It is verified by practical tests whether the documented functions are complete and consistent.

High: The source code has to be supplied. The metrological performance test is still not obsolete because it is very effective. However, parts of the software can be tested either "manually" (well-known methods: code inspection, walk-through etc.) or by the aid of software analyzing tools. Typical examples for such practical tests are the protectiveness of interfaces, the separation of software into parts etc.

ER5.1: The legally relevant software including its hardware and software environment shall be suitably documented.

For a built-for-purpose device with the entire software subject to control as in this example (f, i, l, o), at least the following documentation has to be supplied by the manufacturer:

Comments on examination level:

Low: The operating manual and a technical documentation are supplied by the manufacturer. No additional special software documentation is required. The documentation should contain the manufacturer's declarations about some features of the instrument that are not tested (e.g. that an interface is constructed to be protective) and the legal software identification.

Middle: In addition to the documentation of level low, the special software documentation shall comprise:

- detailed description of all legally relevant software functions, legally relevant parameters that determine the functionality of the instrument
- description of the measuring algorithms (e.g. price calculation and rounding algorithms)
- legal software identification
- complete description of commands and parameters via the protective interface, including a declaration of completeness of this description
- reference to the requirements of this guide
- operating manual

High: In addition to the documentation of level "middle", the source code (as a file) has to be supplied by the manufacturer together with some auxiliary documentation like :

- logic diagram of the software (e.g. flow chart or Nassi-Shneidermann diagram)
- detailed description of the functions of each legally relevant software module
- description of data structures (transmitted data sets)

6.2 Example B: PC-based, Modular Complex Measuring System

The measuring system described in this example can for instance be found in applications like automatic rail-weighbridges, dimensional measuring instruments often in combination with weighing systems, point of sale (POS) systems, etc.

6.2.1 Description of the System

Let the complete measuring system consist of several components (modules) connected by an open network. The system can be characterized by the following features (Figure 6-2):

- *"Sensor modules" consist of sensor, analogue electronics, A/D converter, microcontroller and digital interface to the network but do not have an indication.*
- *There is a "central device" realized by a personal computer. Its monitor is used as indicator for the final measurement values and for stored values.*
- *Each sensor module transmits measurement values to the central device and receives commands from it via the network.*
- *The central device realizes a data storage for legal purposes.*
- *The central device has a windows operating system.*
- *The legally relevant functions of the central device are realized by a program that is loaded from the hard disk drive of the computer. It is compiled to a so called library[11].*
- *The legally relevant software in the central device receives the measurement values from the sensor modules, displays them in a window, stores them for later legal use and exports them to other programs not subject to legal control.*
- *The measured goods cannot be measured statically, i.e. the measuring process is dynamic and complex. (Applies e.g. to automatic rail-weighbridges. For dimensional measuring and static weighing the process is simple and static)*

In the following only the central device is considered. The sensor modules can be treated similarly to the measuring instrument discussed in example A.

[11]) ***Note:*** A (dynamic) software library is a collection of subroutines (or classes in an object oriented language) that can be used by any program (application). The library can be produced separately from the application software. The internal structure of the library is hidden from the programmer of the application.

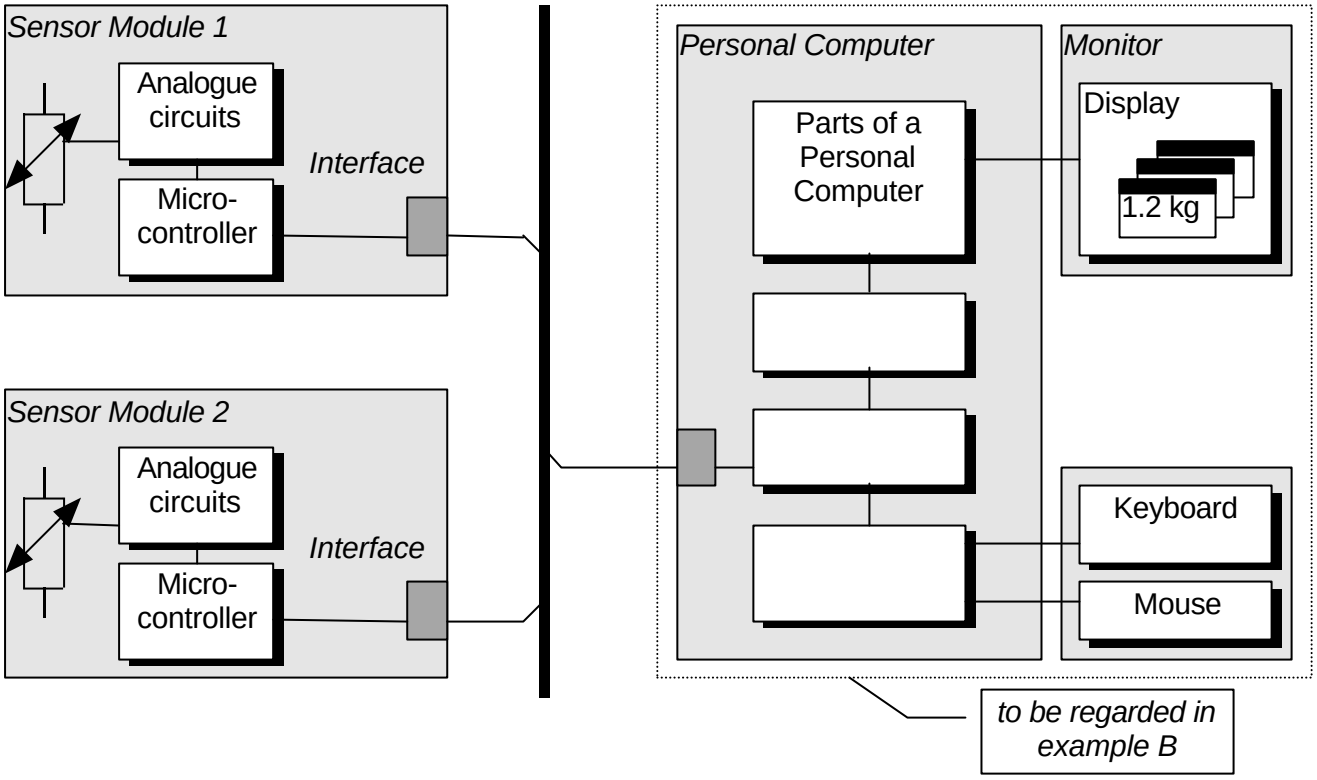


Figure 6-2: *Example B:* PC-based, Modular Complex Measuring System

6.2.2 Legal Classification

Normally the legal classification would follow Table A-1 in Annex 1. As the levels have not yet been selected, in the following interpretations of ***all*** levels for "protection, examination and conformity" are taken into consideration.

Note: As example B is a rather complex system, a lot of interpretations of the general requirements are necessary. To make the interpretations and comments more comprehensible, the following cases have been selected according to the proposal for the category of "Measuring Instruments used for Commercial Transactions" (see Table A-1):

Software protection level: ***middle***

Software examination level: ***middle***

Degree of software conformity: ***low***

These levels and the resulting interpretations are marked with grey background in section 6.2.4 like this note. They are in line with the requirements of the WELMEC guide 2.3 that is already applied to weighing instruments.

Note: *If the stipulated conformity level or protection level is "high", the technical solution of this example will not be suitable to meet these levels for some requirements (ER1.1, ER1.2, ER2.3 etc.).*

6.2.3 Technical Classification

The classification according to chapter 5 here only applies to the central device (personal computer) in the example.

Feature	*Case*	*Explanation*
Hardware configuration	*e*	Modular System, some modules not under legal control, open communication bus system.
User interface (shell)	*h*	Free user shell with operating modes subject to control and operating modes not subject to control in parallel.
Software loading	*k*	Any program can be loaded. Loading can be realized by changeable storages (floppy disc, CD-ROM etc) or by downloading via interface from a server (to hard disc drive, Flash ROM, EEPROM etc).
Software structure	*m*	Parts of the software are subject to legal control. Other parts that are not legally relevant are intended to be modified after approval
Software environment	*p*	The software subject to control is embedded into an environment like a standard operating system that is not especially constructed for the measuring purpose.
Fault detection	*r*	The presence of a defect is not obvious and cannot be easily and simply checked using devices apart from the instrument itself and there are no hardware means for fault detection.
Long-term storage of measurement values	*t*	Measurement values are stored in the system for later legal use.
Measuring principle	*v, x, z*	Single, non-repeatable, complex measurement

In the following sections the cases are simply abbreviated by, for instance, (k).

6.2.4 Interpretation of the Essential Software Requirements

The interpretation here only applies to the central device (personal computer) in the example.

ER1.1: The software of a measuring instrument shall be designed so as to allow ready evaluation of its conformity to the requirements of this guide.

In this example the manufacturer intends to change parts of the software that are not under legal control after type approval (m). Therefore it is necessary to separate the software into two parts (as is sound programming praxis today): One part shall contain all program modules that perform legally relevant functions or that influence legally relevant parameters and data. All program modules that are allowed to be changed, form the other part(s).

Comments on examination level** (ER1.1**):

Low: The design and structure of the software cannot be tested by the normal metrological examination tests. The manufacturer *declares* (without supplying substantiating documentation) all required properties of the software being implemented correctly (e.g. software separation, protectiveness of software interface, intended modifications only in the legally non-relevant part). No examination verifying this declaration is performed.

Middle: The design and structure of the software (software separation, software interfaces etc.) are examined on the basis of a description of the software functions supplied by the manufacturer.

It is verified whether the documentation contains all functions to form the measuring instrument and whether these are defined correctly and consistently.

High: The design and structure of the software (software separation, software interfaces etc.) are examined using the source code in addition to the steps of level middle.

Comments on conformity level** (ER1.1)**:

Low: The manufacturer *declares* that the implemented software of each individual instrument is, and will be, *in conformity* with the approved documentation: The separation of the legally relevant software parts from other parts will be preserved for all versions of the legally non-relevant part of the software to be realized in the future.

The manufacturer shall inform the NB about modifications concerning the separation of the software.

The approved software documentation is kept at the NB. Additionally the complete program code (executable code) of the measuring instrument may exceptionally be deposited.

Middle: The manufacturer shall keep the software part subject to legal control *identical* to the approved one.

Modifications of the part not subject to legal control are allowed without informing the NB as long as the software separation is observed. The manufacturer declares that the software separation will be preserved for all versions of the legally non-relevant part and he shall inform the NB about modifications concerning the separation of the software.

The approved software documentation and the complete program code (executable code) of the measuring instrument are kept at the NB.

High: It is not admissible that the software is modified at all. The design and structure described in the example is ***not admissible*** if conformity level high is stipulated!

ER1.2: The legally relevant software shall be designed in such a way that it is not inadmissibly influenced by other software.

In order to realize data flow between the two software parts and not to violate the separation (see ER1.1), a protective software interface shall be realized between the legally relevant software part and the software not subject to control. This interface comprises

- the interaction between the software parts (e.g. subroutine calls) and
- the data flow between the parts.

In this example the software subject to control is compiled into a library and programs not subject to control can call certain functions of this library to get data or to control some functions (object oriented style). The software interface is realized by the parameters of the called subroutines.

The legally relevant software part shall be designed in a way that the legally relevant functions, parameters and data are also not influenced by the software environment. As loading of software is not restricted technically on a personal computer as in this example, the "software environment" can be any program that is running in parallel. In this example the multitasking operating system (p) is assigned for protecting the functions of the relevant software against inadmissible influences by the software environment.

Measures must be taken to prevent the protective software interface from being circumvented either by the user (see comments on protection level and ER2.2) or by the programmer of the software not subject to legal control (see comments on conformity level). Additional preventions may be necessary to protect the presentation of the measurement values (see ER2.3).

Comments on protection level** (ER1.2)**:

Low: No protection against circumvention of the software interface or influences from the software environment required.

Middle: See ER2.2.

High: Under the stipulated technical and legal conditions, the technical solution of this example is not suitable to realize a high protection level against tampering!

Comments on examination level** (ER1.2)**:

Low: The design and structure of the software cannot be tested by the normal metrological examination tests. The manufacturer *declares* (without supplying substantiating documentation) that all required properties of the software are implemented correctly (e.g. software separation, protectiveness of software interface, intended modifications only in the legally non-relevant part). No examination verifying this declaration is performed.

Middle: The software interface is examined on the basis of the software documentation supplied by the manufacturer. It is verified

- whether the software interface is protective i.e. that all documented commands and data input via the interface variables to the legally relevant software part do not inadmissibly influence it and whether the manufacturer has declared that no other commands than the documented ones are accepted
- whether measures are taken that the software interface isn't likely to be circumvented (compilation of the legally relevant software into a library with an appropriate documentation for the application programmer would fulfil this)
- whether the operating system is able to protect the legally relevant software from influences of the software environment (a multitasking operating system would fulfil this, however, see ER2.2)

High: Additionally to the steps of level middle, the software interfaces are examined using the source code.

Comments on conformity level** (ER1.2)**:

Low: The manufacturer *declares* that the software interface between the software of each individual instrument is protective.

The manufacturer supplies a documentation of the software interface for the application programmer. Besides a description of the usage of the interface, this documentation contains the restrictions the programmer has to observe in order to guarantee that the interface is not circumvented.

The manufacturer shall inform the NB about modifications concerning the software interface between the software parts.

The approved documentation of the software interface is kept at the NB. Additionally the complete program code (executable code) of the measuring instrument may exceptionally be deposited.

Middle: The manufacturer shall keep the software that realizes the protective interface *identical* to the approved one.

The manufacturer produces a documentation of the software interface for the application programmer. Besides a description of the usage of the interface this documentation contains the restrictions the programmer has to observe in order to guarantee that the interface is not circumvented.

Modifications of the part not subject to legal control are allowed without informing the NB as long as the protective software interface is not circumvented. The manufacturer shall inform the NB about modifications concerning the software interface.

The approved software documentation and the complete program code (executable code) of the measuring instrument are kept at the NB.

High: It is not admissible that the software is modified at all. The design and structure described in the example is ***not admissible*** if conformity level high is stipulated!

ER1.3: The legally relevant software shall be designed in such a way that it is not inadmissibly influenced via the interfaces of the device.

The measuring system in the example consists of several sensor modules that are connected to a central device by an open bus. Data exchange between the sensor modules and the central device via the bus is necessary in order to get the final measurement result. There are three possible reasons for inadmissible influences to the measurement result:

1. *the software in the central device could have inadmissible features that could be activated and controlled by inputs via the network interface,*
2. *the data received by the central device could have been influenced or corrupted on their way from the sensor modules via the network,*
3. *the data received by the central device from the network could have been generated by a sender other than a verified sensor module.*

As for no. 2 data transmission is addressed (see ER2.2). As for no. 3 tampering of the system is addressed (see ER2.2). No. 1 deals with the properties of the software that controls the interface of the device. To fulfil the ER1.3 in this sense the interface must be protective; see the following comments.

***Comments on protection level** (ER1.3):*

Low: No protection against tampering of transmitted data or influencing the legally relevant software via the network interface is required.

Middle/high: If the software controlling the interface only lets pass commands that cannot inadmissibly influence the legally relevant functions and data of the software, no tampering via the interface is possible; it is protective. If the programmer realizes the interface software in this way, the technical solution of this example is suitable to guarantee level *middle and high* of protection against tampering.

As for protection of data in an open network see ER2.2.

***Comments on examination level** (ER1.3):*

Low: It cannot be tested by the normal metrological examination tests whether interfaces are protective. The manufacturer *declares* (without supplying substantiating documentation) that no command can be received via the interface that inadmissibly influences legally relevant functions or data of the software. No examination verifying this declaration is performed.

Middle: The interface is examined on the basis of the software documentation that

- *defines and documents the functions that can be controlled via the interface,*
- *defines and documents the parameters that can be set or changed via the interface.*
- *specifies the functions controlled and parameters set that are legally relevant.*

It is examined whether the interface is protective i.e. that all documented commands and data input via the interface to the device do not inadmissibly influence its functions and data, and whether the manufacturer has declared that no other commands than the documented ones are accepted.

High: Additionally to the steps of level middle, the software that controls the interface is examined using the source code.

***Comments on conformity level** (ER1.3):*

Low: The manufacturer *declares* that no command can be received via the interface that inadmissibly influences legally relevant functions or data of the software.

Middle/high: The manufacturer shall keep the software that controls the protective interface identical to the approved one. He has to inform the NB about modifications concerning the software controlling the interface.

ER2.1: Legally relevant programs and data shall be protected against accidental or unintentional changes.

The following effects could lead to accidental or unintentional changes in the example system:

- *physical (electro magnetic, temperature, humidty etc.) effects within the device*
- *electro magnetic effects in the transmission channel*
- *software crashes, viruses*
- *unintentional loading, editing and storing of the program file with a text editor*
- *inadmissible properties of the software subject to control (e.g. it must not be possible to change device specific parameters unintentionally)*

Comments on protection level** (ER2.1)**:

Protection level in the sense used here refers to *intentional* manipulations (see ER2.2).

Comments on examination level** (ER2.1)**:

Low: The manufacturer *declares* (without supplying substantiating documentation) that measures are taken to detect accidental changes (within the devices as well as in the transmission channel) and to suitably react to them. No examination verifying this declaration is performed.

As for unintentional changes the handling of the user shell is practically tested with the help of the operating manual.

Certificates of tests according to the regulations in question (EMC, temperature, humidity etc.) are required.

Middle: The measures for detecting changes of data and programs are examined on the basis of the software documentation supplied by the manufacturer. It is verified whether

- a self checking algorithm is described (In this example: the program checks its integrity automatically e.g. by calculating a checksum over the executable code, comparing it with a nominal value and stopping if the code has been modified.)
- the transmission protocol enables the receiving program to detect accidental changes in the data set transmitted by the sensor modules to the central device (If a protection against *intentional* changes is realized, this requirement ER2.1 concerning accidental or unintentional changes is covered, too.)
- the handling of the user shell has to be documented completely by the manufacturer

As for unintentional changes the handling of the user shell is practically tested with the help of the operating manual.

Certificates of tests according to the regulations in question (EMC, temperature, humidity etc.) are required.

High: Additionally to the steps of level middle, the software that realizes the data transmission and the user shell is examined using the source code.

ER2.2: Legally relevant programs and data shall be protected against corruption or intentional changes by unauthorized persons.

Protection of program code *(ER2.2)*

The central device has an open user shell (h) and tools like editors can be loaded.

Comments on protection level** (ER2.2, program code)**:

Low: No protection measures against tampering are required.

Middle: The legally relevant program must be protected against intentional changes with simple common software tools (text editors). In this example the program checks its integrity automatically e.g. by calculating a checksum over the executable code, comparing it with a nominal value, and stopping if the code has been modified. It is supposed to be difficult enough for an intruder to modify the program code, find the checksum in the code, calculate a new one for the modified code and replace the old one only by aid of a text editor.

It is not possible to influence the running program by the aid of a text editor.

High: The legally relevant software must be protected against intentional changes with special sophisticated software tools (debuggers and hard disc editors, software developing tools) i.e. protection level according to the state of the art in data security like e.g. for financial transactions.

The technical solution of this example would ***not*** be suitable to fulfil this protection level. Additional hardware units in the personal computer of the central device would be necessary to stop debugging (tracing) and to guarantee the integrity of the legally relevant program code.

Comments on examination level** (ER2.2, program code)**:

Low: The manufacturer *declares* (without supplying substantiating documentation) that measures are taken to detect intentional changes and to suitably react on them. No examination verifying this declaration is performed.

The handling of the user shell is practically tested with the help of the operating manual.

Middle: The measures for detecting changes of the legally relevant program are examined on the basis of the software documentation supplied by the manufacturer. It is verified whether

° a self checking algorithm is described (In this example: the program checks its integrity automatically e.g. by calculating a checksum over the executable code, comparing it with a nominal value, and stopping if the code has been modified.)

The software is practically tested. Especially the detection of code modifications is tested with help of a text editor.

High: Additionally to the steps of level middle, the software that realizes the detection of intentional changes and the user shell is examined using the source code.

Protection of type specific parameters *(ER2.2)*

Type specific parameters are normally part of the program code. In this case all comments on the "Protection of program code" (see above) apply. If type specific parameters are stored separately from the program code the comments on "Protection of device specific parameters" (see below) apply.

Protection of device specific parameters *(ER2.2)*

There is one difference between type and device specific parameters: in contrast to the constant type specific data there must be a possibility for adjusting the device specific data before legal verification. Adjusting must not be possible for the user and other unauthorized persons after legal verification. ER2.2 therefore has to be interpreted slightly differently from the interpretation concerning the program code, type specific parameters within the program code etc.

Comments on protection level** (ER2.2, device specific parameters)**:

Low: No protection measures against tampering are required.

Middle/high: Though for this level a text editor is supposed to be the only tool for tampering too, this is not enough because of the possibility of adjusting device specific parameters. It is necessary to seal the adjusting equipment mechanically or by electronic sealing *(definition according WG2, to be added)*. It is technically ***not*** possible to realize this in a standard personal computer like in this example.

In this example all device specific parameters are stored in the sensor modules where they can easily be secured (similar to example A, 6.1).

Comments on examination level** (ER2.2, device specific parameters)**:

Low: The manufacturer *declares* (without supplying substantiating documentation) that no device specific parameters are stored within the central device of this example. No examination verifying this declaration is performed.

The software is practically tested with the help of the operating manual to check how the device specific parameters are set and whether securing is possible.

Middle: The manufacturer documents all device specific parameters. He describes where they are stored and how they can be secured.

In the examination it is verified on the basis of the documentation that these parameters cannot be adjusted or changed by the user or other unauthorized persons.

The user shell is practically tested. Especially the way device specific parameters are set has to be checked.

High: Additionally to the steps of level middle, the software is examined using the source code. Especially those parts that are responsible for storing the device specific parameters are examined. This software part must be blocked by any hardware means.

Protection against circumventing a software interface *(ER2.2)*

In this example a protective software interface is realized. Circumventing the interface *by the user* enables him to influence functions of the legally relevant software part or to change or modify variables or parameters that are not allowed to be set. See the items above concerning protection of program code, data and parameters.

Protection of transmitted data *(ER2.2)*

In this example, measurement values are transmitted via a network and received by the central device for final processing (e). Transmitted data must be protected for two reasons:

- *the data received by the central device could have been influenced or corrupted on their way from the sensor modules via the network (the data have lost their integrity),*
- *the data received by the central device from the network could have been generated by a sender other than a verified sensor module (the data are not authentic).*

Comments on protection level** (ER2.2, transmitted data)**:

Low: No protection measures against tampering are required.

Middle: *Integrity.* The legally relevant transmitted data must be protected against intentional changes with simple common software tools (text editors). This can be realized e.g. by an electronic signature (see 2.6) or by encryption.

The security level depends on the algorithm and key length of the signature (or encryption). An acceptable solution for the protection level middle would be e.g. the CRC [11, 12] algorithm with a key / signature length of 2 bytes for each data set with one measurement value.

Authenticity. The receiver of measurement values or other legally relevant data must be able to check whether the data have been sent by an authorized sender and whether the data are actual. An acceptable solution for the protection level middle would be e.g.

- *the registration of the addresses of all legal senders on the net in the receiver, combining the sender's address with the measurement value, and then, after transmission, checking in the receiver whether the address is valid*
- *combining a time stamp with the measurement value, then transmitting it and checking in the receiver whether it is actual*

All relevant data necessary to finally process or re-verify a measurement value, including signature, sender address, time stamp etc. must be grouped in one data set and the signature must cover all fields of the data set.

Data that are detected as having been corrupted must not be used.

The key used for checking or generating the signature must be treated as legally relevant data.

High: The legally relevant software must be protected against intentional changes with special sophisticated software tools (debuggers and hard disc editors, software developing tools) i.e. a protection level according to the state of the art in data security like e.g. for financial transactions.

The same as for level middle applies, however, the signature algorithm and key length mentioned above are too weak. An acceptable solution for a signature algorithm would be e.g. DEA[12] with a minimum key length of 128 bits.

But even if the algorithm and key meet the level high, the technical solution of this example would ***not*** be suitable to reach this protection level because the receiver (central device) is a standard personal computer with no appropriate protection means (see comment on ER1.2, protection level high).

Comments on examination level *(ER2.2, transmitted data)**:***

Low: The manufacturer *declares* (without supplying substantiating documentation) that measures are taken to detect intentional changes of data sets received from another module and to suitably react on them. No examination verifying this declaration is performed.

Middle: The measures for detecting changes of transmitted legally relevant data are examined on the basis of the software documentation supplied by the manufacturer. It is verified whether

- *a suitable signature algorithm and sufficient key length is realized*
- *all data necessary to finally process and to protect transmitted measurement values are combined to a data set (necessary fields in the data set are e.g. measurement value, address of the sender, time stamp and current number of the measurement)*
- *the signature key cannot be read or explored by aid of a text editor*

Practical test: A data set is falsified and sent to the central device. The reaction to this error is checked.

High: Additionally to the steps of level middle, the software that realizes the detection of intentional changes and the signature generation is examined using the source code.

Protection of long-term stored data *(ER2.2)*

In this example measurement values are stored by the central device for later legal use (t). There is a program subject to control for presenting stored values to the user. It enables him to find and clearly assign any earlier measurement result to a certain stored data set (within the appointed space of time).

Comments on protection level *(ER2.2, long-term stored data)**:***

Low: No protection measures against tampering are required.

[12]) Specification of Algorithm DEA in [10]

Middle: *Integrity.* The legally relevant stored data must be protected against intentional changes with simple common software tools (text editors). This can be realized e.g. by an electronic signature (see 2.6) or by encryption.

The security level depends on the algorithm and key length of the signature (or encryption). An acceptable solution for the protection level middle would be e.g. the CRC [11, 12] algorithm with a key / signature length of 2 bytes for the data set with one measurement value.

Authenticity. The user of the stored measurement values must be able to assign each value to a certain measurement. An acceptable solution for the protection level middle would be e.g.

° *combine an ID like a unique (current) number with the measurement value*

° *combine a time stamp with the measurement value*

All relevant data necessary to re-verify a measurement value, including signature, file id-number, time stamp etc. must be grouped in one data set and the signature must cover all fields of the data set.

Data that are detected as having been corrupted must not be used.

The key used for checking the signature must be treated as legally relevant data.

High: The legally relevant software must be protected against intentional changes with special sophisticated software tools (debuggers and hard disc editors, software developing tools) i.e. protection level according to the state of the art in data security like e.g. for financial transactions.

The same as for level middle applies, however, the signature algorithm and key length mentioned above are too weak. An acceptable solution for a signature algorithm would be e.g. DEA[13] with a minimum key length of 128 bits.

But even if the algorithm and key meet the level high, the technical solution of this example would ***not*** be suitable to reach this protection level because the program for signing and verifying a data set runs on a standard personal computer (central device) with no appropriate protection means (see comment on ER1.2, protection level high).

Comments on examination level *(ER2.2, long-term stored data)**:***

Low: The manufacturer *declares* (without supplying substantiating documentation) that measures are taken to detect intentional changes of stored data sets and to suitably react on them. No examination verifying this declaration is performed.

Middle: The measures for detecting changes of stored legally relevant data are examined on the basis of the software documentation supplied by the manufacturer. It is verified whether

° *a suitable signature algorithm and sufficient key length is realized*

° *all data necessary to protect a stored measurement values are combined to a data set (necessary fields in the data set are e.g. measurement value, file id-number, time stamp of the measurement)*

° *the signature key cannot be read or explored by aid of a text editor*

Practical test: A stored data set is falsified in the central device using a text editor. The reaction to this error is checked.

High: Additionally to the steps of level middle, the software that realizes the detection of intentional changes and the signature generation is examined using the source code.

ER2.3: ***Only the approved and verified software is allowed to be used for legal purposes. It shall be clear and unambiguous that a presentation of a result is generated by a legally relevant program.***

If a standard personal computer with a windows operating system like in this example is part of the measuring system, two problems have to be solved to meet ER2.3:

[13]) Specification of Algorithm DEA in [10]

- *a program other than the approved one could be loaded either by the manufacturer when installing or by an unauthorized person when the system is in use (k)* [k]
- *programs other than the approved one could control the windows on the screen, and the presentation of the measurement values could be disturbed or inhibited (h, p)* [h p]

Installing of the program code subject to legal control *(ER2.3)*

The manufacturer shall only install the approved software on the system for the legal purpose.

Comments on conformity level** (ER2.3)**:

Low: The manufacturer is allowed to correct the program code without changing the legal software identification. As far as legally relevant software parts are concerned (in the example: the library with subroutines subject to legal control) the Notified Body must, however, be informed in any case. At verification the appropriate authority or a responsible person checks by the legal software identification that the software implemented in the instrument is in ***conformity*** with the approved software.

Middle: The legally relevant part of the implemented software (in the example: the library) has to be identical to the approved software. At verification the appropriate authority or a responsible person checks by the legal software identification (e.g. signature) that the software implemented in the instrument is ***identical*** with the approved software.

The user can rely on the ***sealings*** and ***verification mark*** that the approved program has been installed.

High: If conformity level high is stipulated, the technical solution of this example system is ***not*** suitable. The entire software including the legally non-relevant parts has to be identical with the approved software, and modifications of software parts after type approval are not admissible.

Exchanging of the program code subject to legal control after verification *(ER2.3)*

The software of a standard personal computer like the central device of the example system can be freely loaded even by the user (k).

Comments on protection level** (ER2.3)**:

Low: No protection measures against changing and substituting of the approved program are required.

Middle: It is assumed that the only tool used for tampering with the system is a text editor, but not a compiler. A compiler would however be necessary to write a new program that has similar functions to the approved one. It is assumed that it is a criminal act to write such a program and substitute it for the approved one. Therefore no measures are required to inhibit loading of programs. (As for tampering with the code of the approved program and of data and parameters see ER2.2.)

Note: *If the manufacturer produces programs similar to the approved one, he shall not install them on a system to be legally verified and not make them available to the user of the approved system.*

High: If conformity level high is stipulated, the technical solution of this example system is ***not*** suitable.

Comments on conformity level** (ER2.3)**:

Low/middle: The user and the verification officer or the person responsible can verify whether the approved software is loaded and running by comparing the indicated legal software

---

[k]) Any program can be loaded. Loading can be realized by changeable storages (floppy disc, CD-ROM etc) or by downloading via interface from a server (to hard disc drive, Flash ROM, EEPROM etc).

[h]) Free user shell with operating modes subject to control and operating modes not subject to control in parallel.

[p]) The software subject to control is embedded into an environment like a standard operating system that is not especially constructed for the measuring purpose.

identification with that registered on the main plate of the device or in the type approval certificate.

High: If conformity level high is stipulated, the technical solution of this example system is ***not*** suitable.

Identifying the legally relevant presentation *(ER2.3)*

In this example several programs can run in parallel. It is possible that not only the presentation of the legally relevant program is seen on the screen of the personal computer (h, p). Some restrictions must be observed in order to give priority to the legally relevant presentation.

Comments on protection level** (ER2.3)**:

Low: No protection measures against indicating falsified measurement values are required.

Middle: It is assumed that a text editor is used for tampering with the system. It cannot be excluded that a presentation of (falsified) values is generated with a modern (window) text editor. Therefore technical measures must be taken in the program subject to control to prevent this. There are three measures an acceptable technical solution should realize:

- *Measurement values received from the sensor modules are only processed by the program part subject to control, and no access is given to other programs as long as the measurement values are not yet indicated (or stored in a long-term storage subject to control). At the moment they are displayed and/or stored they can be exported to program parts not subject to control.*
- *The program subject to control generates a window on the screen for presentation of the relevant data that is always on top, overwrites all other windows and is refreshed in certain time intervals. If the window is not on top any more, processing of measuring values stops.*
- *The window for presenting the measurement values has to be designed in a way that it cannot be mixed up with a window generated by a text editor. There must be a copy of the window generated by the program subject to control in the operating manual*.

Note: It is assumed to be a criminal act to write a program that is able to process and indicate the measurement values instead of or in parallel to the approved program.

High: If conformity level high is stipulated, the technical solution of this example system is ***not*** suitable.

Comments on examination level** (ER 2.3)**:

Low: The manufacturer *declares* (without supplying substantiating documentation) that measures are taken to force the window of the program subject to control always on top and that measurement values are not exported to other programs until they have been displayed or stored. No examination verifying this declaration is performed.

Middle: The measures for protecting the legally relevant presentation are examined on the basis of the software documentation supplied by the manufacturer. It is verified whether

- *measurement values are processed only by the legally relevant program until they have been indicated or stored*
- *the window for presentation of the measurement values is always on top*
- *the design of the window is not similar to a window of a text editor*

Practical test: It is practically tested that the measurement window cannot be suppressed and is always on top as long as measurement values are processed.

High: Additionally to the steps of level middle, the software that realizes e.g. the refreshing of the measurement window is examined using the source code.

ER2.4: Functional defects that can falsify measurement values in software controlled hardware shall be detected and acted upon.

In the example some kinds of functional defects are detected and the software realizes the appropriate reaction (r)[r].

Comments on examination level:

Low: The instrument is tested practically with the help of the operating manual. As functional defects happen rather seldomly, the failure detection mechanism normally isn't tested.

Middle: The failure detection mechanism described in the documentation is checked by simulating suitable failures.

High: The failure detection mechanism is tested as in case middle. Additionally other failures are simulated and the reaction of the instrument is judged.

ER3.1: The software shall not inadmissibly be modified after type approval.

What kind of modifications are admissible depends on the level of the required conformity level:

Comments on conformity levels:

Low: The implemented software of each individual instrument is in conformity with the approved documentation. Regardless of minor corrections of the source code the functionality remains identical to the technical documentation:

- Modifications of the legally relevant software are allowed as long as the documented functions and characteristics of the approved instrument remain unchanged. The NB[14]) must, however, be informed. Changes of documented functions and characteristics require additional approval by the NB and a new legal software identification.
- Modifications of the part not subject to legal control are allowed without informing the NB as long as the software separation is observed and exclusively the approved software interface is used.
- The approved software documentation is kept at the NB. Additionally the complete program code (executable code) of the measuring instrument may exceptionally be deposited.

Middle: The legally relevant part of the implemented software of each individual instrument is identical to the approved software:

- Because of the identity, modifications of the legally relevant software lead to a new legal software identification. The NB gives an additional approval in this case.
- Modifications of the part not subject to legal control are allowed without informing the NB as long as the software separation is observed and exclusively the approved software interface is used.
- The approved software documentation and the complete program code (executable code) of the measuring instrument are kept at the NB.

High: The entire software of each individual instrument is identical to the approved software:

- Because of the identity, modifications of any part of the software lead to a new legal software identification. The NB gives an additional approval in this case.
- The approved software documentation and the complete program code (executable code) of the measuring instrument are kept at the NB.

ER3.2: For the verification of conformity an identification of the legally relevant software and suitable instructions shall be available.

---

[r]) The presence of a defect is not obvious and cannot be easily and simply checked using devices apart from the instrument itself and there are no hardware means for fault detection.

[14]) NB - Notified Body or design examiner

An instruction for the user and verification officer must be provided that explains how to indicate the legal software identification number. It depends on the level of the required conformity level how the conformity of the individual instrument is checked:

Comments on conformity levels:

Low: The implemented software of each individual instrument is in conformity with the approved documentation. Regardless of minor corrections of the source code the functionality remains identical to the technical documentation:

- At verification the conformity with the approved software is checked by a legal software identification that is mentioned in the type approval certificate. The legal software identification may be displayed either on demand or automatically on start up or cyclically).

Middle: The legally relevant part of the implemented software of each individual instrument is identical to the approved software:

- At verification the conformity with the approved software is checked by a legal software identification (signature) that is mentioned in the type approval certificate.

High: The entire software of each individual instrument is identical to the approved software:

- At verification the conformity with the approved software is checked by a legal software identification (signature) that is mentioned in the type approval certificate.

ER4.1: The functionality of the instrument shall be testable.

As assumed in the description of the example system, a complex measurement process is realized here. The metrological tests are difficult and cannot be repeated very often.

Comments on examination level:

Low: The manufacturer supplies the results of measurement series, a description of the conditions during the measurements and a declaration that these measurement values have been made with the software version that is to be approved. The measurement results are checked by the examiner. Other features of the software that are not covered by these measurements don't need to be made testable by the manufacturer. It is sufficient that he declares that these untested features conform with the requirements (protectiveness of an interface, failure detection and reaction etc.).

Consequence for the manufacturer/applicant: Sufficient measurements have to be performed and compared to nominal values. The results shall be documented. The arrangement of the measuring equipment and the conditions during the measurements shall be documented.

Middle: The metrological input signals for the interpreting parts of the software are simulated by a special test device or by test software. The results that are calculated by the software of the instrument are treated as if they were real measurement values. In this example there is no special interface necessary to enter simulated values because the communication bus is suitable to connect a simulator and to enter simulated data sets.

Some practical tests on the basis of the special software documentation are performed in addition to these simulator tests. From the result of this analysis the examiner can derive additional tests on the real instrument (e.g. test the functioning of the failure detection and reaction).

Consequence for the manufacturer/applicant: The instrument shall be equipped with one or more interfaces for monitoring measurement signals or data streams or for entering simulated signals or data streams. If need be, he shall make available a suitable simulator device or program.

High: The metrological input signals for the interpreting parts of the software are simulated by a special test device or by test software. The results that are calculated by the software of the instrument are treated as if they were real measurement values. In this example there is no special interface necessary to enter simulated values because the communication bus is suitable to connect a simulator and to enter simulated data sets.

The source code has to be supplied. Still the metrological simulator performance test is not obsolete because it is very effective. However, parts of the software can be tested either "manually" (well-known methods: code inspection, walk-through etc.) or by the aid of software analyzing tools. Typical examples for such spot check tests are the protectiveness of interfaces, the separation of software into parts etc.

Consequence for the manufacturer/applicant: The instrument shall be equipped with one or more interfaces for monitoring measurement signals or data streams or for entering simulated signals or data streams. If need be, he shall make available a suitable simulator device or program.

ER5.1: The legally relevant software including its hardware and software environment shall be suitably documented.

For modules of a measuring system like in this example (technical class see 6.2.3), at least the following documentation has to be supplied by the manufacturer:

Comments on examination level:

Low: The operating manual and a technical documentation is supplied by the manufacturer. No additional special software documentation is required. The documentation should contain the manufacturer's declarations about some features of the instrument that are not tested (e.g. that an interface is constructed to be protective) and the legal software identification.

Middle: In addition to the documentation of level low the special software documentation shall comprise:

- detailed description of all legally relevant software functions, legally relevant parameters that determine the functionality of the instrument
- description of the measuring algorithms (e.g. price calculation and rounding algorithms)
- description of the menus and dialogues
- legal software identification
- complete description of commands and parameters via the protective interface, including a declaration of completeness of this description
- complete description of commands and parameters via the protective software interface, including a declaration of completeness of this description
- description of data sets of stored or transmitted data
- necessary characteristics of the operating system and of the hardware of the computer
- reference to the requirements of this guide
- operating manual

High: In addition to the documentation of level "middle" the source code (as a file) has to be supplied by the manufacturer together with some auxiliary documentation like

- logic diagram of the software (e.g. flow chart or Nassi-Shneidermann diagram)
- detailed description of the functions of each legally relevant software module
- description of data structures (transmitted data sets)

7 References and Other Literature

[1] Measuring Instruments Directive (MID/3), 1998, Working Document, European Commission - III.D.2

[2] Council Directive 90/384/EEC on the harmonization of the laws of the Member States relating to non-automatic weighing instruments. Official Journal of the European Communities, L 189, Vol. 33, 20.7.1990, 1-16

[3] Guide for Examining Software (Non-automatic Weighing Instruments), WEMEC 2.3, 1995

[4] IEC 65(Sec)183 Software Documentation, 1994

[5] ISO 7498-1 to -4, Information technology - Open Systems Interconnection, 1989 - 1997

[6] ISO/IEC 9126 Information technology; Software product evaluation, October 1994

[7] ISO/IEC 12119 Information technology; Software packages; Quality requirements and testing, August 1995

[8] Information Technology Security Evaluation Criteria (ITSEC), June 1991, Version 1.2, Document COM(90) 314, Luxembourg

[9] ISO 8731-1:1987 Banking - Approved algorithms for message authentication - Part 1 : DEA

[10] ISO 10126-2:1991 Banking - Procedures for message encipherment - Part 2: DEA algorithm

[11] ITU-T (formerly CCITT) V.42

[12] ISO/IEC 13239 Information technology - Telecommunication an Information exchange between systems -High-level data link control (HDLC) procedures, 1996

[13] Security of computerized instruments, Jean-François Magana, OIML Bulletin Volume XL, Number 3, July 1999

[14] ISO 2382-1, Information technology, Part 1: Fundamental Terms, 1993
ISO 2382-7, Information technology, Part 7: Computer Programming, 1989

Annex I

Proposal for the Assignment of Levels (see Chapter 4)

The actual edition of the draft MID contains specific annexes for 11 measuring instruments. These instruments have been grouped in Table A-1 into 4 categories. This is an attempt to suitably combine instruments with regard to the future development of special software guides (see Chapter 3, Fig. 3-1).

The table contains a proposal for the assignment of levels (as defined in chapter 4) to the different categories. Further subdivision of the categories may turn out to be necessary in the course of the work of WG7 and its expert subgroups.

The differentiation of the risk of fraud will be based on the subjective assessment of the subgroup experts rather than on objective criteria. Possible criteria are (see [13]):

- *potential gain of tampering with the computer program,*
- *the penalty incurred against the potential gain,*
- *the probability of the detection of the fraud,*
- *the number of people being involved in the fraud,*
- *the number of measuring instruments manufactured,*
- *the type of customers and users of the instruments.*

The assignment of levels is intended to be a guiding principle rather than a binding regulation. Exceptions may be defined for special applications or for special kinds of measuring instruments. A final proposal for an assignment will not be given before enough experience with the guide has been gathered and an agreement between all member countries and all organizations and associations involved has been found.

Note: *The proposed levels for software conformity may later be discussed and defined depending on the modules B+D and H1 of the MID, especially reflecting the existence of appropriate software quality assurance measures.*

Category	***MID Annex***	***Risk of Fraud***	***Software Protection Level***	***Software Examination Level***	***Degree of Software Conformity***
Supply to the customer by mains	MI-001, MI-002, MI-003, MI-004	middle	*middle*	*middle*	*middle*
		high	*high*	*middle*	*middle*
Commercial transactions and services	MI-005, MI-006, MI-007, MI-009	middle	*middle*	*middle*	*low*
		high	*high*	*middle*	*middle*
Evidential measurement	MI-010	-	*high*	*high*	*high*
Environment, safety, health	MI-011	middle	*middle*	*middle*	*low*

MI-001 Water meters
MI-002 Gas meters
MI-003 Active electrical energy meters and measurement transformers
MI-004 Heat meters
MI-005 Measuring systems for the continuous and dynamic measurement of quantities of liquids other than water
MI-006 Automatic weighing instruments
MI-007 Taximeters
(MI-008 Material measures, no software, not relevant)
MI-009 Dimensional measuring instruments
MI-010 Evidential breath analysers
MI-011 Exhaust gas analysers

Table A-1: Proposal for the assignment of levels (as defined in sections 4.1 to 4.3) for different categories of measuring instruments