SLOVENSKI STANDARD
SIST-TS ES 59008-6-1:2007

01-januar-2007

Zahtevani podatki za polprevodniška integrirana vezja - 6-1. del: Formati za izmenjavo podatkov in podatkovni slovar - Format datoteke DDX

Data requirements for semiconductor die -- Part 6-1: Exchange data formats and data dictionary - Data exchange - DDX file format

Ta slovenski standard je istoveten z: **ES 59008-6-1:1999**

ICS:

31.080.01	Polprevodniški elementi (naprave) na splošno	Semiconductor devices in general

SIST-TS ES 59008-6-1:2007 **en**

EUROPEAN SPECIFICATION

SPÉCIFICATION EUROPÉENNE

EUROPÄISCHE SPEZIFIKATION

ES 59008-6-1

September 1999

English version

Data requirements for semiconductor die
Part 6-1: Exchange data formats and data dictionary
Data exchange - DDX file format

This European Specification was approved by CENELEC on 1999-06-29.

CENELEC members are required to announce the existence of this ES in the same way as for an EN and to make the ES available promptly at national level in an appropriate form. It is permissible to keep conflicting national standards in force.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Ref. No. ES 59008-6-1:1999 E

Foreword

This European Specification has been prepared by the CENELEC BTTF 97-1, Known good die.

It was submitted to the vote during the meeting of BTTF 97-1 and approved by CENELEC as ES 59008-6-1 on 1999-06-29.

The following dates was fixed:

- latest date by which the existence of the ES has to be announced at national level (doa) 1999-11-01

The structure of this European Specification is as follows.

ES 59008 Data requirements for semiconductor die
- Part 1 General requirements
- Part 2 Vocabulary
- Part 3 Mechanical, material and connectivity requirements
- Part 4 Specific requirements and recommendations
  - Part 4-1: Test and quality
  - Part 4-2: Handling and storage
  - Part 4-3: Thermal
  - Part 4-4: Electrical simulation
- Part 5 Particular requirements and recommendations for die types
  - Part 5-1: Bare die
  - Part 5-2: Bare die with added connection structures
  - Part 5-3: Minimally-packaged die
- Part 6 Exchange data formats and data dictionary
  - Part 6-1: Data exchange - DDX file format
  - Part 6-2: Data dictionary

Introduction

This European Specification has been developed to facilitate the selection of unpackaged and minimally-packaged semiconductor die, with or without connection structures in order to save both design and procurement time.

It is a data specification which defines the requirements for :

- product identity
- product data
- die mechanical information
- test, quality and reliability information
- handling, storage and mounting information
- thermal data and electrical simulation data.

This document was prepared by CENELEC Task Force CLC/BTTF 97-1 Known Good Die. Other organisations that helped prepare it were: the ESPRIT GOOD-DIE projects, EECA, Sematech, DPC and EIAJ.

The specification was derived from the work carried out in the ESPRIT 4th Framework project GOOD-DIE. This project was set up to develop a database for the selection of unpackaged and minimally-packaged semiconductor die, with or without connection structures, and for the downloading of information to CAD design stations to facilitate the layout and simulation of MCMs and hybrid circuits. During the early part of the GOOD-DIE project the need was identified for a standard way of presenting information for the selection and procurement of these components.

1. Scope

This series of European Specifications specifies requirements for the exchange of data pertaining to bare semiconductor die, with or without connection structures, and minimally packaged semiconductor die.

This Specification also gives recommendations for general industry good practice in the use of bare die, with or without connection structures, and minimally packaged die.

ES 59008-6 specifies the data format that may be used for the exchange of data which is covered by other parts of this Specification as well as definitions of all parameters used according to the principles and methods of IEC 61360.

Part 6-1 introduces a Device Data Exchange (DDX) format, with the prime goal of facilitating the transferral of adequate geometric data between die manufacturer and CAD/CAE user. The data format has been kept intentionally flexible to permit usage beyond this initial scope.

All such definitions will, in due course, be offered to the IEC 61360 Maintenance Agency for inclusion in the IEC reference collection, see Part 6-2.

2. Normative references

The following standards contain provisions which, through reference in this text, constitute provisions of this part of ES 59008.

ES 59008-1, *Data requirements for semiconductor die -- Part 1: General requirements*

ES 59008-2, *Data requirements for semiconductor die -- Part 2: Vocabulary*

ES 59008-3, *Data requirements for semiconductor die -- Part 3 Mechanical, material and connectivity requirements*

ES 59008-4-1, *Data requirements for semiconductor die -- Part 4-1: Specific requirements and recommendations - Test and quality*

ES 59008-4-3, *Data requirements for semiconductor die -- Part 4-3: Specific requirements and recommendations - Thermal*

ES 59008-4-4, *Data requirements for semiconductor die -- Part 4-4: Specific requirements and recommendations - Electrical simulation*

IEC 61360; series, *Standard data element types with associated classification scheme for electric components*

Die Information Exchange (DIE) Format Reference Manual, Version 1.0.3, 23/11/1994

ISO 8601:1988 (& ANSI X3.30), *Numerical date/time interchange format.*

ISO 6093:1985, *Information Processing - Representation of numerical values in character strings for information interchange.*

3. Definitions

For the purpose of this European Specification, the definitions given in ES 59008-2 apply.

4. Requirements

This Part 6-1 of the specification should be read in conjunction with ES 59008-1: General Requirements and ES 59008-3: Mechanical, material and connectivity requirements. Particular attention should be given to the data requirements for conformancy levels, as defined in ES 59008-3.

5. Device Data eXchange format (DDX) file goals and usage.

To facilitate the transferral of data by electronic media from the device vendor to the end-user for use within a CAD or CAE system, a data file format, **D**evice **D**ata e**X**change, (**DDX**), shall be used. This data file format has been deliberately kept flexible, to permit further enhancements and additions for future use.
It is strongly recommended that **D**evice **D**ata e**X**change files have the three letter **DDX** file extent, and a **D**evice **D**ata e**X**change file shall hereon be referred to as a **DDX** file.

5.1 Data that is to be transferred from a device vendor to a user shall be contained in a single computer readable **DDX** file, and the minimum contents of this file shall suffice a geometric CAD/CAE software design system. The file shall be textually readable, to permit simple manual verification.

5.2 The **DDX** file and its data contents shall be independent of both computer machine and operating system.

5.3 The **DDX** file contents shall include mechanical and interconnectivity information, but may additionally include electrical and functional data. Different levels of conformance are catered for; see ES 59008-3 for details and requirements of these conformance levels.

5.4 The **DDX** file may contain data for one or more devices, and shall be capable of being used as a library file by a CAD/CAE software design system. The file may contain one or more sets of data for the same device type, each having different delivery forms, such as bumped die, bare die, and Chip-Scale packaging.

5.5 The **DDX** file shall be capable of being simply or automatically generated, such as by an ASCII text editor or a spreadsheet.

5.6 The **DDX** file shall be capable of referencing additional external files, such as simulation and thermal model files.

5.7 All data shall be defined in such a way that conversion to or from other exchange formats is possible, such as GDSII and CIF for geometric data of die. As close a compatibility to the existing DIE (Die Information Exchange) data as possible is desired, to facilitate simple translation of partial DIE data files.

5.8 Definitions of parameters shall be in conformity with IEC 61360 (see ES 59008-3, clause 5).

6. DDX file format and file format rules.

The **DDX** file shall be an ASCII compatible text file with suitable line termination. Line termination will depend upon the operating system. DOS/Windows® generally uses a carriage/line-feed <CR/LF> terminator (ASCII 0Dh/0Ah), whereas UNIX® invariably relies solely upon a line-feed <LF> (ASCII 0Ah) terminator, the carriage return <CR> (ASCII 0Dh) being present by implication.

6.1 All data not complying to the data syntax (see 7.3) shall be treated as a remark and, as such, ignored.

6.2 All mandatory data shall be present. Missing data shall be flagged as an error, rendering that data unusable.

6.3 ASCII characters 00h to 7Fh are permitted, ASCII characters 80h to FFh shall be ignored.

6.4 All text data shall be case independent.

6.5 Underscores "_" shall be ignored in a variable or property name, and may be used as intermediate name separators. Underscores are valid within textual string and name data.

6.6 A comma "," shall be used as a data separator.

6.7 All data lines shall be terminated with a semicolon, ";".

6.8 Braces are used to open and close structures or BLOCKs. An open brace "{" shall be used to begin a structure or block, and a close brace "}" shall be used to terminate a structure or block.

6.9 Brackets "()" shall be permitted, then ignored, in numeric data for clarity (e.g. in co-ordinate pairs).

6.10 To accommodate typical spreadsheet CSV (Comma Separated Variable) format outputs, textual data may be inside double quotes **""**, and matching pairs of double quotes shall be ignored.

6.11 Mathematical operations, calculations or formulae shall not be permitted within numeric data.

6.12 Space characters (ASCII 20h) and tab characters (ASCII 09h) shall both be treated as space separators, multiple space and tab characters will syntactically be treated as a single space separator.

6.13 Lines beginning with a hash "#" shall be treated as an intentional comment. All data on that line shall be ignored.

7. DDX file content

7.1 DDX *file content rules*

7.1.1 Data shall only exist within a block structure, referred to as a DEVICE block, and one or more DEVICE blocks, each containing data, may exist within a single file. Each DEVICE block is unique, and shall only contain data relevant to a single device, having a specific device form. All data within each DEVICE block shall be treated as being local and unique only to that block. (See 6.8)

7.1.2 There are two types of parameters use for data, structures and variables, and these parameters shall only exist with a DEVICE block :

7.1.2.1 A structure determines a set or multiple sets of data having different data types.

7.1.2.2 A variable is equated to a single or multiple data of a single data type.

7.1.3 Data types are :

7.1.3.1 **Textual string data,**

All ASCII characters from ASCII 20h to ASCII 7Fh are permitted within textual data, characters including and above ASCII 80h shall be ignored. Consideration may be given to special print and display control characters to permit the printing of underscore or overscore characters. It is advised that textual string data is placed within pairs of double quotes, see 6.10.

7.1.3.2 **Textual name data,**

All names shall be unique, and shall only consist of the following characters from the ASCII character set :-

A-Z a-z 0-9 $ - % & ! @ _ .

When textual name data is used as a file name, it is advised that the characters are limited to 8 characters for the actual file name and to 3 characters for the file extent, with a point "." used as the name/extent delimiter, in line with many common operating systems. It is advised that textual name data is placed within pairs of double quotes, see 6.10.
Note that all textual data is case independent, and spaces are not permitted within a textual name.

7.1.3.3 **Real numeric data,**

Real numeric data shall comply to ISO 6093:1982, and shall consist of the characters :-

0-9 + - . E e

The data values may be signed, and use engineering or scientific notation, but shall not include dimensional units, e.g.

90008, 9000.80, 9.0008E5, -5207, -5.207E3, 0.102, 102E-3

Note that a comma "," is used as a data separator, and therefore shall not be used as a replacement for a decimal point ".".

7.1.3.4 **Integer numeric data,**

Integer numeric data values shall comply to ISO 6093:1982, and only the characters **0** to **9** are permitted. Integers shall be unsigned, and shall not include dimensional units.
For practical purposes, an integer shall be limited to 16-bit resolution, i.e. integer values between and including 0 to 65536 only are acceptable.

7.1.3.5 **Date data,**

Date data values shall comply with ISO 8601:1988 format, Yr2000 compliant, and may include time information as well e.g.

"**YYYY-MM-DD**", "**YYYYMMDD**", "**YYYY-MM-DDTHH:MM:SS**".

7.1.4 No variable identifier or variable name shall be referenced prior to being defined, to permit a single pass parsing.

7.1.5 All units shall belong to the SI system, apart from the geometric unit of the micron (10^{-6}m), the inch and the mil (10^{-3} inch). Only one unit of dimension shall be permitted within a single DEVICE block. Note that the inch and the mil are non-preferred units, and are only present due to continued common usage.

7.1.6 In all co-ordinate data, the **X** co-ordinate shall precede the **Y** co-ordinate, and in all co-ordinate data, the **Y** co-ordinate shall precede the **Z** co-ordinate (i.e. **X,Y** or **X,Y,Z**).

7.1.6.1 The **X** co-ordinate shall be the horizontal axis (numerically left to right),

7.1.6.2 The **Y** co-ordinate shall be the vertical axis (numerically bottom to top), and

7.1.6.3 The **Z** co-ordinate shall be depth axis (numerically near to far).

7.2 DDX *DEVICE block syntax*

DEVICE device_name device_form {
relevant die data ……
}

7.2.1 The **DDX** file may contain one or more DEVICE blocks, all data pertaining to a particular device shall be embedded within the relevant block. (See 6.8 and 7.1.1).

7.2.1.1 A DEVICE block is opened by the **DEVICE** keyword and opening brace "{", (as shown), and

7.2.1.2 the DEVICE block is closed by the matching closing brace "}".

7.2.2 Data not within a DEVICE block structure shall be treated as a remark, permitting the future addition of checksum information, file creation date and historical data etc., within the **DDX** file, without affecting the actual device data.

7.2.3 The **device_name** is the given name by which the device shall be referred, and the **device_form** is the mechanical form of the device to which the block data pertains.

7.2.4 Valid data for the **device_form** variable are

7.2.4.1 **bare_die**,

7.2.4.2 **bumped_die**,

7.2.4.3 **lead_frame_die** and

7.2.4.4 **minimally_packaged_device** (or **MPD**).

Further **device_form** types may be added at a later stage, see IEC 61360-1, XKE004-001, "die type code", for further details.

7.2.5 Only one **DEVICE** block having **device_name** of type **device_form** shall be present within the **DDX** file, but duplication of either **device_name** or **device_form** is permissible.

7.2.6 An example of a typical **DDX** file arrangement of DEVICE blocks :-

```
DEVICE name1 bare_die {
       relevant data for device "name1" as a bare die..
}
DEVICE name1 bumped_die {
       relevant data for device "name1"as a bumped die..
}
DEVICE name2 mpd {
       relevant data for device "name2" as a minimally packaged device..
}
DEVICE name2 bare_die {
       relevant data for device "name2" as a bare die..
}
DEVICE name1 mpd {
       relevant data for device "name1" as a minimally packaged device..
}
DEVICE name3 bare_die {
       relevant data for device "name3" as a bare die..
}
```

In the above example, there are 3 occurrences of a DEVICE block for device "name1", and 2 occurrences of a DEVICE block for device "name2", but each of these DEVICE blocks specify a different device_form. The order or sequencing of the DEVICE blocks has no relevance.

7.3 DDX *data syntax*

Property = value [, value] ;

<**property**>[*equate separator*]<value/variable {*separator* <value/variable> }>[*data terminator*][*line terminator*]

<**property**>	::=	Parameter name
[*space*]	::=	{space character (20h) or tab character (09h)}0+
[*equate separator*]	::=	[space]{equal =}[space]
[*separator*]	::=	[space]{comma ,}[space]
[*data terminator*]	::=	[space]{semicolon ;}
[*line terminator*]	::=	{CR or CR/LF}

e.g.

1. **thickness** = 1000;
2. **Thickness**=470;
3. **geometric_units**=micron;
4. **geometricunits** = micron;
5. **GeometricUnits**= "millimetres";
6. **terminal_type** = T1, Circle, 220;
7. **Terminal_Type** = T2, Rectangle, 200, 250 ;
8. **TerminalType** = T2, O, (200, 250);
9. **TERMINALTYPE** = T2, O, 200 , 250 ;

Thus, **terminal_type, Terminal_Type**, **TerminalType** and **TERMINALTYPE** will all reference the same parameter name.

8. Definitions of DEVICE block parameters

Where a parameter is unique to the device_form, as defined in the DEVICE block, the parameter will be preceded with the following …

DIE_	data parameter is unique to bare die or bumped die form
BUMP_	data parameter is unique to only bumped die
MPD_	data parameter is unique to a minimally packaged device, such as a CSP
WAFER_	data parameter is unique to a die device delivered at wafer level
LEAD_	data parameter is unique to a die device with attached lead frame.

Within the following list of data parameters (clause 8.1 onwards), the following are shown :

- the parameter name, as used syntactically within the DDX file,
- the relevant DET code, this is the IEC 61360 Data Element Type code,
- the parameter type, indicating either a variable or structure data type,
- the parameter function, determining its usage and meaning,
- the parameter value, indicating the type of data expected,
- any parameter limitation, indicating any limitation within the DEVICE block,
- parameter dependencies, highlighting parameters that must have been declared prior to invocation,
- one or more practical examples, and
- any relevant notes.

A brief table of parameters is given in Annex E, and a working example of a full DDX DEVICE block is given in Annex A, with its expected graphical output in Annex B.

To determine the actual conformance level requirements, reference shall be made to ES 59008-3.

Terms and conventions

A point of electrical connection is called a terminal. This may be either a bond-pad for a bare die, and may equally refer to the landing or connection footprint area required by an interconnection medium. It is common convention for die to have the initial terminal, "Pin 1" in the upper left hand corner of the die, and for terminal or pin numbering to continue counter-clockwise.

The X co-ordinate dimensions are for the "width" or horizontal plane with increasing positive values to the right. The Y co-ordinate dimensions are for the "height" or vertical plane, with increasing positive value to the top. The Z co-ordinate dimensions are for "depth", with increasing positive values away from the users view.
As a point of reference, all die components, including bumped die, are generally "viewed" active side upwards.

Summary of general rules

- All valid data must be contained within a DEVICE block.
- Any local or unique parameter, such as a name, must be defined prior to its usage.
- Before any geometric variable is defined, the units of measurement, GEOMETRIC_UNITS, must be defined.
- Before any geometric co-ordinates are defined, the geometric origin, GEOMETRIC_ORIGIN, and the geometric view, GEOMETRIC_VIEW, must be defined.
- Before any TERMINALs are referred to, the TERMINAL_COUNT parameter must have been defined, and the number of TERMINALs shall not exceed the TERMINAL_COUNT value.
- Before any TERMINAL_TYPEs are referred to, the TERMINAL_TYPE_COUNT parameter must have defined, and the number of TERMINAL_TYPEs shall not exceed the TERMINAL_TYPE_COUNT value.

8.1 DEVICE_FORM Parameter

This is defined within the DEVICE block heading by the **device_form** parameter.

Parameter Name	device_form
DET code	XKE004-001, as "die type code"
Parameter Type	Variable, refer to clause 7.2 on DEVICE blocks
Parameter Function	Defines the physical form of the DEVICE.
Parameter Values	Textual string data, as clause 7.1.3.1
Example	`bare_die, bumped_die, MPD`
Notes	Refer to clause 7.2

8.2 DEVICE_NAME Parameter

This is defined within the DEVICE block heading as the **device_name** parameter

Parameter Name	device_name
DET code	XKE002-001, as "die name"
Parameter Type	Variable, refer to clause 7.2 on DEVICE blocks
Parameter Function	Defines the DEVICE's reference name.
Parameter Values	Textual name data, as clause 7.1.3.2
Example	`SN74LS04`
Notes	Refer to clause 7.1.6 and clause 7.2

8.3 VERSION Parameter

Parameter Name	VERSION
Parameter Type	Variable
Parameter Function	Specifies the revision of DDX specification to which this DEVICE block conforms.
Parameter Values	Textual string data, as clause 7.1.3.1
Limitations	Shall be declared only once within a single DEVICE block.
Example	`VERSION = "ES57008, Issue 1.0";`

8.4 MANUFACTURER Parameter

Parameter Name	MANUFACTURER
DET code	XKE036-001, as "firm's name"
Parameter Type	Variable
Parameter Function	Specifies the manufacturer or fabrication house of the device.
Parameter Values	Textual string data, as clause 7.1.3.1
Limitations	Shall be declared only once within a single DEVICE block.
Example	`MANUFACTURER = "Texas Instruments Inc.";`

8.5 FUNCTION Parameter

Parameter Name	FUNCTION
Parameter Type	Variable
Parameter Function	A brief description of the devices' function.
Parameter Values	Textual string data, as clause 7.1.3.1
Limitations	Shall be declared only once within a single DEVICE block.
Example	`FUNCTION = "16 Bit Microprocessor";`
Notes	EN 61360-4, specifies certain functional and application classes that may be used.

8.6 GEOMETRIC_UNITS Parameter

Parameter Name	GEOMETRIC_UNITS
DET code	XKE115-001, as "geometric units"
Parameter Type	Variable
Parameter Function	Specifies the geometric units that shall apply to all geometric values within the DEVICE block.
Parameter Values	Textual string data, as clause 7.1.3.1. One of the following values : • micrometre, or microns, • metre, • millimetre, • inch • mil (1.0E-3 inch)
Limitations	Shall be declared only once within a single DEVICE block.
Example	`GEOMETRIC_UNITS = microns;` `GEOMETRIC_UNITS = mil;`
Notes	The GEOMETRIC_UNITS must be declared before any geometric units are used. The micron is generally the default dimensional unit for die dimensions, and that the inch and mil are non-preferred units. (Refer to clause 7.1.5).

8.7 GEOMETRIC_VIEW Parameter

Parameter Name	GEOMETRIC_VIEW
Parameter Type	Variable
Parameter Function	Specifies the geometric view that shall apply to all geometric shapes within the DEVICE block.
Parameter Values	Textual string data, as clause 7.1.3.1. One of the following values : • TOP meaning active side upwards, and • BOTTOM meaning active side downwards.
Limitations	Shall be declared only once within a single DEVICE block.
Example	`GEOMETRIC_VIEW = top;` `GEOMETRIC_VIEW = bottom;`
Notes	The GEOMETRIC_VIEW must be declared before any geometric shapes are created It would be common for a bare die and packaged part to be viewed as the "TOP" view, whereas bumped die may well be viewed from the "BOTTOM" (i.e. through the substrate).

8.8 SIZE Parameter

Parameter Name	SIZE
DET code	XKE070-001, as "die length", and XKE071-001 as "die width"
Parameter Type	Variable
Parameter Function	Determines the X- and Y- dimensions of the device
Parameter Values	Real X-dimension, Real Y-dimension, as clause 7.1.3.3
Dependencies	GEOMETRIC_UNITS, GEOMETRIC_VIEW
Limitations	Shall be declared only once within a single DEVICE block.
Example	`SIZE = 250, 500.5;`
Notes	Dimension values are in GEOMETRIC_UNITS.