

JESD3-C



Reproduced By GLOBAL
ENGINEERING DOCUMENTS
With The Permission of EIA
Under Royalty Agreement

JEDEC STANDARD

Standard Data Transfer Format Between Data Preparation System and Programmable Logic Device Programmer

JESD3-C

(Revision of JESD3-B)

JUNE 1994

ELECTRONIC INDUSTRIES ASSOCIATION
ENGINEERING DEPARTMENT



NOTICE

JEDEC Standards and Publications contain material that has been prepared, progressively reviewed, and approved through the JEDEC Council level and subsequently reviewed and approved by the EIA General Counsel.

JEDEC Standards and Publications are designed to serve the public interest through eliminating misunderstandings between manufacturers and purchasers, facilitating interchangeability and improvement of products, and assisting the purchaser in selecting and obtaining with minimum delay the proper product for his particular need. Existence of such standards shall not in any respect preclude any member or nonmember of JEDEC from manufacturing or selling products not conforming to such standards, nor shall the existence of such standards preclude their voluntary use by those other than EIA members, whether the standard is to be used either domestically or internationally.

JEDEC Standards and Publications are adopted without regard to whether their adoption may involve patents or articles, materials, or processes. By such action, JEDEC does not assume any liability to any patent owner, nor does it assume any obligation whatever to parties adopting the JEDEC Standards or Publications.

The information included in JEDEC Standards and Publications represents a sound approach to product specification and application, principally from the solid state device manufacturer viewpoint. Within the JEDEC organization there are procedures whereby a JEDEC Standard or Publication may be further processed and ultimately become an EIA Standard.

Inquiries, comments, and suggestions relative to the content of this JEDEC Standard should be addressed to the JEDEC Executive Secretary at EIA Headquarters, 2001 Pennsylvania Ave., N.W., Washington, D.C. 20006.

Published by

©ELECTRONIC INDUSTRIES ASSOCIATION 1994
Engineering Department
2001 Pennsylvania Ave., N.W.
Washington, D.C. 20006

PRICE: Please refer to the current
Catalog of EIA, JEDEC, and TIA STANDARDS and ENGINEERING PUBLICATIONS
or call Global Engineering Documents, USA and Canada (1-800-854-7179)
International (303-397-7956)

Printed in U.S.A.
All rights reserved

PLEASE!
DON'T VIOLATE
THE
LAW!

This document is copyrighted by the EIA and may not be reproduced without permission.

Organizations may obtain permission to reproduce a limited number of copies through entering into a license agreement with the EIA. For information, contact:

EIA Engineering Publications Office
2001 Pennsylvania Ave., N.W.
Washington, D.C. 20006
(202)457-4963

STANDARD DATA TRANSFER FORMAT BETWEEN DATA PREPARATION SYSTEM AND PROGRAMMABLE LOGIC DEVICE PROGRAMMER

CONTENTS

1 INTRODUCTION	1
1.1 Purpose and Scope	1
1.2 Summary of Programming and Testing Fields	2
1.3 Changes to October 1983 Standard	2
1.4 Changes clarifying preload test vectors	2
1.5 Addition of Register Observation Vector	3
1.6 Additions to JESD3-B that implement JESD3-C	3
2 SPECIAL NOTATIONS AND DEFINITIONS	3
2.1 Notation Conventions	4
2.2 BNF Rules and Definition	5
2.3 PLD Register Numbering	5
3 TRANSMISSION PROTOCOL	5
3.1 Protocol Syntax	6
3.2 Computing the Transmission Checksum	6
3.3 Disabling the Transmission Checksum	6
4 DATA FIELDS	6
4.1 General Field Syntax	7
4.2 Field Identifiers	7
5 COMMENT AND DEFINITION FIELDS	7
5.1 Design Specification	8
5.2 Note (N)	8
5.3 Device (D)	8
5.4 Values (QF,QP,QV)	8
6 DEVICE PROGRAMMING FIELDS	9
6.1 Syntax and Overview	9
6.2 Fuse Default State (F).....	10
6.3 Fuse List (L).....	10
6.4 Fuse Checksum (C)	11
6.5 Electrical Fuse Data (E)	12
6.6 User Data (U)	12
6.7 Device Identification (J)	13
7 DEVICE TESTING FIELDS	13
7.1 Syntax and Overview	14
7.2 Default Test Conditions (X)	15
7.3 Test Vectors (V)	15
7.4 Pin Sequence (P)	15
7.5 Test Conditions	16
7.6 Register Preload	19
7.7 Register Observation.....	19

8 PROGRAMMER/TESTER OPTIONS	21
8.1 Security Fuse (G)	21
8.2 Signature Analysis Test (S, R, T)	22
8.3 Access Time (A)	22
9 EXAMPLES	22
9.1 Data File Examples	22
ANNEX	
A1 INTERACTIVE PROGRAMMING	25
A1.1 INTRODUCTION	
A1.1.1 Purpose and Scope	25
A1.1.2 Summary of Reporting and Testing Fields	25
A1.1.3 Changes to Standard No. 3A	25
A1.2 DATA FILE TYPES	
A1.2.1 General	26
A1.2.2 Source (S-File)	26
A1.2.3 Acknowledgement (A-File)	27
A1.2.4 Continuation (C-File)	28
A1.3 FILE REPORTING FIELDS	
A1.3.1 General	30
A1.3.2 Mode Declaration (M Field)	30
A1.3.3 Error messages (ME Field)	30
A1.3.4 Vector failure (MV Field)	31
A1.4 DEVICE TESTING FIELDS	
A1.4.1 General	31
A1.4.2 Super-Voltage Definition (QE Field)	32
A1.5 IMPLEMENTATION CONSIDERATIONS	
A1.5.1 Standard No. 3A	32
A1.5.2 Programming Specifications	33
A1.5.3 Programmer QE field support requirements	33
A1.6 EXAMPLE FILES	
A1.6.1 Voltage reference adjustment	33
A1.6.2 Logic replacement	35

1 INTRODUCTION

1.1 Purpose and Scope

This standard was developed to prevent the proliferation of data transfer formats that occurred with microprocessor development systems. The focus of the standard is on field-programmable devices and their support tools. It is not intended for other types of semicustom logic devices or other types of fabrication or testing equipment.

The standard includes a simple transmission protocol based on traditional PROM formats that allow a device programmer to share a computer serial port with a terminal. This simple protocol is not a complete communication protocol and does not do retries or error correction. If the device programmer has local storage, such as floppy disk, for the programming data this protocol is not required.

This standard defines a data format for transferring the fuse or cell states between the development system and the programmer. It does not define the device architecture such as types of logic arrays or the output macro cell. Also the standard does not define the programming algorithms or the device and technology specific information for accessing the fuses or cells.

Field-programmable logic devices may require more testing than programmable memories, so the standard defines a simple functional testing format. This test vector format is not a general purpose parametric test language.

```
<STX>File for PLD 12S8 Created on 8-Feb-85 3:05PM
6809 memory decode 123-0017-001
Joe Engineer Advanced Logic Corp *
QP20* QF448* QV8*
F0* X0*
L0000 11111011111111111111111111111111*
L0028 10111111111111111111111111111111*
L0056 11101111111111111111111111111111*
L0112 01010111011110111111111111111111*
L0224 01010111101110111111111111111111*
L0336 01010111011101111111111111111111*
V0001 000000XXXNXXXHHHLXXN*
V0002 010000XXXNXXXHHHLXXN*
V0003 100000XXXNXXXHHHLXXN*
V0004 110000XXXNXXXHHHLXXN*
V0005 111000XXXNXXXHLHHXXN*
V0006 111010XXXNXXXHHHHXXN*
V0007 111100XXXNXXXHHLHXXN*
V0008 111110XXXNXXXLHHHHXXN*
C124E*<ETX>8A76
```

Figure 1. Example of a Programmable Logic Device Data File

1.2 Summary of Programming and Testing Fields

The programming and testing information is contained in various fields. The following list gives the field identifier and a description. To comply with the standard the device programmer, tester, and development system must provide and recognize certain fields.

Identifier	Description		
(n.a.)	Design Specification		
N	Note		
QF	Number of Fuses in Device ***		
QP	Number of Device Package Pins ***		
QV	Maximum Number of Test Vectors ***		
F	Default Fuse State *	E	Electrical Data *
L	Fuse List *	U	User Data
C	Fuse Checksum	J	Device Identification
X	Default Test Condition **		
V	Test Vectors **		
P	Pin Sequence **		
D	Device (Obsolete)		
G	Security Fuse		
R,S,T	Signature Analysis		
A	Access Time		
*	Programmer must recognize		
**	Tester must recognize		
***	Development System must provide		

1.3 Changes to October 1983 Standard

The 1983 standard defined the D, F, L, C, V, and P field. Several other fields are now being used, and this new standard formally defines these fields. The new standard is a superset of the 1983 standard.

The sequence and allowed combinations of the various fields have been clarified. One field, the D field, was not clearly defined and this led to conflicting uses. The D field is made obsolete in this standard.

Several changes have been added to the test vectors to defined unspecified vectors and don't care conditions. The test vectors now allow for manufacture independent register preloading.

1.4 Changes clarifying preload test vectors

The B preload test vector, intended to allow preloading of buried registers, has been clarified. The previous definition was ambiguous. In addition, register numbering guidelines have been added, and a "read-and-retain" test condition has been added to allow selective preloading of registers, without disturbing other registers.

In Standard 3-A, the QP field was defined as containing the number of pins in the test vectors. The new standard defines the QP field as containing the number of pins on the device package. Conditions under which the QP field is mandatory are listed in 5.4

1.5 Addition of a register observation vector

A T vector has been added to allow the observation of all registers in a device. This allows equal access to output, input, and internal registers. Its format parallels that of the B preload vector.

1.6 Additions to JESD3-B which implement JESD3-C

Electrical Fuse Data (E): The E field has been added to allow for the handling of special feature fuses that do not affect the function of the device. An example of this type of fuse is the power miser fuses available in some types of PLD devices. With this addition, the integrity of the L field will remain intact such that one common JEDEC file can be utilized to program any device with the same logical functionality.

User Data (U): The U field has been added to allow for the handling of special feature fuses that do not affect the logic function of the device. An example of this type of fuse is the User Data Signature (information only) available in some types of PLD devices. With this addition, the integrity of the L field will remain intact such that one common JEDEC file can be utilized to program any device with the same logical functionality.

Device Identification (J): The purpose of the J field is to provide a means of verifying that a particular JEDEC file is appropriate for the device which has been selected for programming. Each PLD architecture is assigned a unique code, and each pinout variation of an architecture is assigned a unique code. Uniqueness is defined more thoroughly below. The intent is for the device programmer to check the device and JEDEC file before programming and then issue an error message if the device code in the JEDEC file does not match the device type that has been selected for programming. This field does not provide for verification of the actual device in the socket.

2 SPECIAL NOTATIONS AND DEFINITIONS

2.1 Notation Conventions

In addition to descriptions and examples this document uses the Backus-Naur Form (BNF) to define the syntax of the data transfer format. BNF is a shorthand notation that follows these rules:

- o "::<=" denotes "is defined as".
- o Characters enclosed by single quotes are literals (required).
- o Angle brackets enclose identifiers.
- o Square brackets enclose optional items.
- o Braces (curly brackets) enclose a repeated item. The item may appear zero or more times.
- o Vertical bars indicate a choice between items.
- o Repeat counts are given by a :n suffix. For example, a six digit number would be defined as "<number> ::= <digit>:6."

For example, in words, the definition of a person's name reads:

The full name consists of an optional title followed by a first name, a middle name, and a last name. The person may not have a middle name or may have several middle names. The titles consist of: Mr., Mrs., Ms., Miss, and Dr.

BNF Syntax:

**<full name> ::= [<title>] <f. name> {<m. name>} <l. name>
<title> ::= 'Mr.' | 'Mrs.' | 'Ms.' | 'Miss' | 'Dr.'**

Examples

Miss Mary Ann Smith

Mr. John Jacob Joseph Jones

Tom Anderson

2.2 BNF Rules and Definitions

The following standard definitions are used throughout the rest of this document:

**<digit> ::= '0' | '1' | '2' | '3' | '4'
 | '5' | '6' | '7' | '8' | '9'**

**<hex-digit> ::= <digit>
 | 'A' | 'B' | 'C' | 'D' | 'E' | 'F'**

<binary-digit> ::= '0' | '1'

<number> ::= <digit> {<digit>}

<del> ::= <space> | <carriage return>

<delimiter> ::= <del> {<del>}

<printable character> ::= <ASCII 20 hex ... 7E hex>

**<control character> ::= <ASCII 00 hex ... 1F hex>
 | <ASCII 7F hex>**

<STX> ::= <ASCII 02 hex>

<ETX> ::= <ASCII 03 hex>

<carriage return> ::= <ASCII 0D hex>

<line feed> ::= <ASCII 0A hex>

<space> ::= <ASCII 20 hex> | ''

**<valid character> ::= <printable character>
 | <carriage return> | <line feed>**

**<field character> ::= <ASCII 20 hex ... 29 hex>
 | <ASCII 2B hex ... 7E hex>
 | <carriage return> | <line feed>**

2.3 PLD Register numbering

The B and T vectors require that PLD manufacturers provide a register grouping and numbering sequence to the PLD programmer vendors. This numbering sequence will be used in the B and T vectors.

The registers are numbered from 1 to N, where N is the maximum number of registers in the device. The PLD manufacturer is responsible for assigning and documenting the register numbers.

There are three types of registers: output, internal, and input. Output registers are connected directly to a device pin used as an output. Internal registers do not have a direct connection (or any connection) to a device pin. Input registers are connected to device pins used as dedicated inputs. If a register can be used as an input or output register, then this register is classified as an output register for register numbering purposes.

Registers must be sequenced in the following order:

1. Output and I/O registers
2. Internal registers
3. Input Registers

Within a group, all registers should be numbered in ascending sequence, in the order of the lowest-number programmable element that can be associated with each register. For example, if the lowest numbered element used by internal register A is 31 and the lowest-numbered element used by internal register B is 61, then A is numbered first in the register sequence. However, even if the lowest-numbered element used by output register C is 91, C is still numbered ahead of A and B, since output registers are numbered before internal registers.

3 TRANSMISSION PROTOCOL

3.1 Protocol Syntax

This simple STX-ETX protocol is based on traditional PROM formats that allow a device programmer to share a serial computer port with a terminal. The transmission consists of a start-of-text (STX) character, various fields, an end-of-text (ETX) character, and a transmission checksum. The character set consists of the printable ASCII characters and four control characters (STX, ETX, CR, LF). Other control characters should not be used because they can produce undesirable side-effects in the receiving equipment.

Syntax of the Transmission Protocol:

`<format> ::= <STX> {<field>} <ETX> <omit checksum>`

3.2 Computing the Transmission Checksum

The transmission checksum is the 16-bit sum (i.e., modulo 65,535) of all ASCII characters transmitted between and including the STX and ETX (see figure 2). The parity bit is excluded in the calculation.

Syntax of the Transmission Checksum:

```
<xmit checksum> ::= <hex-digit>:4

random text <return><line feed>           = 0000
<STX>TEST*<return><line feed> 02+54+45+53+54+2A+0D+0A = 0183
QF0384*<return><line feed> 51+46+30+33+38+34+2A+0D+0A = 01A7
F0* <return><line feed> 46+30+2A+20+20+0D+0A = 00F7
L10 101*<return><line feed> 4C+31+30+20+31+30+31+2A+0D+0A = 01A0
<ETX>05C4 <return> random text           03 = 0003
```

—
05C4

Figure 2. Computing the Transmission Checksum.

3.3 Disabling the Transmission Checksum

Some computer operating systems do not allow the user to control what characters are sent, especially at the end of a line. The receiving equipment should always accept a dummy value of "0000" as a valid checksum. This dummy checksum is a method of disabling the transmission checksum.

4 DATA FIELDS

4.1 General Field Syntax

In general, each field in the format starts with an identifier, followed by the information, and terminated with an asterisk. For example, "C1234*" specifies that the checksum of the fuse data is 1234. The design specification header does not have an identifier and must be the first field in the transmission, immediately following the STX.

Syntax of Fields:

```
<field> ::= [<delimiter>] <field identifier>
           {<field character>} '*'

<field identifier> ::= 'A' | 'C' | 'D' | 'E' | 'F' | 'G' | 'J'
                    | 'L' | 'N' | 'P' | 'Q' | 'R'
                    | 'S' | 'T' | 'U' | 'V' | 'X'

<reserved identifier> ::= 'B' | 'H' | 'I' | 'K'
                       | 'M' | 'O' | 'W' | 'Y' | 'Z'
```

4.2 Field Identifiers

Each field begins with a single character identifier that identifies the field type. Multiple character identifiers can be used to create subfields (i.e., "A1", "A3", or "AB3"). The field is terminated with an asterisk. Therefore, asterisks cannot be embedded within the field. While not required, carriage returns and line feeds should be used to improve the readability of the format. Reserved identifiers currently have no function and are reserved for future use. Receiving equipment should ignore fields starting with reserved identifiers. The meanings of the field identifiers are given in table 1.

A - Access Time	N - Note
B - *	O - *
C - Checksum	P - Pin sequence
D - Device type	Q - Value
E - Electrical Fuse Data	R - Resulting vector
F - Default fuse state	S - Starting vector
G - Security fuse	T - Test cycles
	U - User Data
I - *	V - Test vector
J - Device Identification	W - *
K - *	X - Default test condition
L - Fuse list	Y - *
M - *	Z - *

Table 1. Field Identifiers (* indicates reserved for future use)

5 COMMENT AND DEFINITION FIELDS

5.1 Design Specification

The design specification is the first field in the format, must be included, and does not have an identifier signaling its start. An asterisk terminates the field. The contents of the design specification are not defined but should consist of

1. User's name and company
2. Date, part number, and revision
3. Manufacturer's device number
4. Other information

Syntax of the Design Specification:

<design specification> ::= {<field character>} '*'

Example:

```
File for PLD 12S8
Created on 8-Feb-85 3:05PM
6809 memory decode 123-0017-001
Joe Engineer Advanced Logic Corp *
```