TABLE OF CONTENTS

1 Introduction to ATDF

Since the Standard Test Data Format (STDF) was introduced in 1985, it has become a de facto standard in the semiconductor test industry. For the first time, there was a standard that could accommodate test data from analog, digital, memory, and mixed signal test systems, from multiple vendors. STDF was optimized to increase the speed of output from the tester (to minimize the impact on test times), and to reduce disk space usage. To meet these goals, the trade-off was against ease of implementation.

Advanced test systems, whose executives and test languages allow logging data without modification to the test program, can easily provide users with the benefits of STDF. More primitive tester software, however, can require users to include special statements in the test program to write out datalog information. Owners of such primitive tester software find STDF implementation much more difficult. This is usually because many of these test systems do not provide an easy method of writing binary data, or because their proprietary CPU types use a peculiar data format and are not supported by the STDF specification.

Because of this, each test programmer usually writes data in a different format because the system does not impose any standards. The same programmer will often vary the datalog format slightly from one program to the next. As a result, it is impossible to write a single set of data analysis programs that will work with all the test data from one kind of tester. It also becomes prohibitively expensive to write conversion programs from all these diverse outputs to a single standard such as STDF.

To make the advantages of STDF available to this class of tester owners, we have developed a specification for an ASCII version of the Standard Test Data Format, called ASCII Test Data Format (ATDF). This format incorporates all the records and fields provided by STDF, but makes them available in a way that is simple to implement for a test programmer using any language. By providing a concise, easy-to-understand specification, every test programmer in a given facility (or even across a company) can produce data that conforms to a single standard.

A product is available with this specification that contains a conversion program that translates between STDF and ATDF in both directions. This product makes it easier and less expensive to collect and analyze data from diverse kinds of testers.

☑ **NOTE**

This specification refers to the *Standard Test Data Format (STDF) V4 Specification*. You should have a copy of the *STDF V4 Specification* available for reference. See "Introduction to STDF" on page 1-1 of the *STDF Specification*.

ATDF Design Objectives

The following are the major design objectives of ATDF:

- Provide a simple test data format that can be written from any program on any kind of computer.
- Make sure all STDF information is present in ATDF in the same records.
- Provide a way to edit test data using commonly available software.
- Minimize disk usage (to the extent possible with ASCII data).

General Record Layout and Requirements

This section covers the following rules for ATDF record format and other requirements:

Record Terminator

Each record in an ATDF data file consists of a string of ASCII characters terminated by a carriage return and/or a line feed. Implied carriage returns, for operating systems like VMS and RSX that support them, are also valid.

Record Header

Each record begins with a three character record header followed by a colon. The record headers are the same record acronyms used in the STDF specification. For example, the record header for a Master Information Record is "MIR:".

Field Separators

The record header is followed by zero or more data fields, depending on the record type. Data fields are separated by a single field separator character. The default field separator character is the vertical bar (|).

To override the default for a given file, replace the first separator in the File Attributes Record (FAR) with the character you want to use as the separator in this file. (This separator is the sixth character in the file.) You must use this separator throughout the file. Obviously certain characters would be bad choices as field separators because they are likely to appear in the data.

Note that no separator appears between the record header (which ends with a colon) and the first data field.

Optional Fields

Optional fields at the end of a record may be omitted. Optional fields within a record must be present, but may be empty. Such a missing field is signified by consecutive field separator characters.

Line Continuation

If the computer being used to write the data places a limit on the maximum size of a record, the ATDF record may be continued on the next line. To do this, simply begin the continuation line with a space. The space will tell the program reading the file that this is a continuation of the previous line, rather than a record header for the next record. The space will not be included in the data for that field. It is not necessary to terminate the continued line at the end of a field. For example, an FTR that is continued in the middle of a field might look like:

```
FTR:27|2|1|P||15|5|72|14|3|2|X00800100|X30000000|XFFFFF
  FFF|XCFFFFFFF|XFFFFFFFF|ALL_ONES|Check Driver|FUNC_TESTS|H
```

Required Records and Fields

It is very important to follow the instructions in each record definition about record placement. All required records must be in the file. If a record is included, then all required fields within that record must also be included.

Note that “required” means required in a valid ATDF file. Software that uses or analyzes ATDF files may require some records or fields that are not required in a valid ATDF file. You may need to fill in fields or records in addition to those marked “required” in this specification. Check the documentation for the analysis software to see what ATDF data it requires.

Record Ordering

Rules for ATDF record ordering within the file are identical to those for STDF.

ATDF File Names

The rules for ATDF file names are identical to those for STDF except that the file extension must begin with `.atd` rather than `.std`.

Data Truncation

Since the main purpose of this data format is to provide an easy way to get foreign test data into STDF format, it will be necessary on conversion to truncate ATDF data fields if they are longer than the corresponding STDF fields. All variable length STDF ASCII fields may be up to 255 bytes long. ATDF data that is longer will be truncated to 255 bytes. STDF also has several short fixed-length fields. ATDF fields will be truncated to the size of the STDF field

on conversion. ATDF fields that are shorter than the corresponding STDF fixed-length ASCII field will be left justified and padded with spaces.

Test Numbers

The usual reason for collecting data in any sort of standard format is to allow for subsequent analysis. Therefore, it is absolutely necessary that test numbers be unique in the test data. One test number should represent one and only one test and set of input conditions in the test program. Otherwise it is not possible to perform useful statistical analysis on the data.

ATDF Support and Other Software

Note that although the ATDF specification and converter support all STDF records, this is not necessarily true for other conversion and data analysis programs that may use this data. Please check the documentation to determine which records are used or supported by the software you are planning to use.

Data Representation

This section covers the following rules for ATDF data representation:

Floating Point Data

Floating point data within the ATDF and STDF specifications is accompanied by fields that specify the scaling factor and precision of that data. The purpose of this information is to ensure that data is represented precisely in reports and analyses that use that data and that additional precision is not implied where it does not exist. For a detailed explanation of result and test limit scaling and precision, please refer to the section entitled "Storing and Displaying Parametric Test Data" in the Parametric Test Record (PTR) description in the STDF specification.

Floating point numbers in ATDF may be represented in decimal notation (for example, 93.2) or floating point notation (for example, 3.2E-7).

Scaling

The STDF specification requires that test results and limits be scaled to whole units. The ATDF specification is less restrictive in this area. The ATDF specification supports either scaled or unscaled parametric data, based on a flag set in the first record of the file, the File Attributes Record (FAR). Unscaled data will be automatically scaled when it is run through the converter.

Arrays

Each array is a single field in the ATDF. Each element in the array is separated by a comma. Arrays may be of the following types: hexadecimal numbers, integers, and floating point numbers.

Time and Date Fields

All time/date fields in the ATDF use the actual time and date rather than the UNIX representation as in the STDF. This time and date is of the form:

```
hh:mm:ss DD-MMM-YYYY
```

where:

`hh` is the hour of the day

mm	is the minute of the hour
ss	is the second of the minute
DD	is the day of the month
MMM	is the first three characters of the English name of the month
YYYY	is the full four digit year

Insignificant leading zeroes in all numbers are optional.

Additional Points

- Normally, all data should be written in 7-bit ASCII using 8-bit bytes. The high order bit of the byte will be 0. For special applications, it may be possible to use 8-bit data in the ATDF, but users should be aware that this can be interpreted differently on different computers (for example, as extended ASCII or Kanji).
- The ATDF specification does not support multiple byte characters, such as those used in Kanji character sets. This is because the second byte may contain bit patterns that could be interpreted as a field separator or record terminator. These patterns could also create problems as bogus record terminators when converted to STDF format.
- Numeric fields must be decimal (integer or floating point as appropriate).
- Binary bit patterns, such as the vector data in the FTR, must be represented as hexadecimal. For ease of human readability, hexadecimal fields may optionally be preceded by the letter X, but this is not required by the ATDF specification.
- On conversion to STDF, leading spaces in text fields are retained, but trailing spaces are deleted.
- ATDF cannot support embedded carriage control characters in the data. For example, carriage return, line feed and form feed are all illegal.

2 ATDF Record Types

This section contains the definitions for the ATDF record types. The following information is provided for each record type:

- A statement of function: how the record type is used in the ATDF file.
- A table defining the data fields: first the ATDF record header, then the fields specific to this record type. The information includes the field name, the name of the STDF field it is associated with, a brief description of the field, and an indication if it is a required field (“Req?”).
- Any additional notes on specific fields or use of the record.
- Frequency with which the record appears in the ATDF file: for example, once per lot, once per part, once per test, and so on.
- The location of the record in the ATDF file.

The records are given in a logical order, according to their use in the file. In addition, this section contains a listing of all record types, alphabetically by the three-letter abbreviations, for easy reference. See the “Alphabetical Listing by Abbreviation” on page 2-2.

Alphabetical Listing by Abbreviation

In this section, the ATDF record types appear in the same order that the STDF record types appear in the STDF Specification (that is, in order of ascending record type and record subtype codes). For easier reference, the ATDF record types are listed here in alphabetical order, by the three-letter abbreviations for the record types

Table 2-1. ATDF record types by abbreviation

Abbreviation	Record Type
ATR	“Audit Trail Record (ATR)” on page 2-5
BPS	“Begin Program Section Record (BPS)” on page 2-48
DTR	“Datalog Text Record (DTR)” on page 2-52
EPS	“End Program Section Record (EPS)” on page 2-49
FAR	“File Attributes Record (FAR)” on page 2-4
FTR	“Functional Test Record (FTR)” on page 2-42
GDR	“Generic Data Record (GDR)” on page 2-50
HBR	“Hardware Bin Record (HBR)” on page 2-12
MIR	“Master Information Record (MIR)” on page 2-6
MPR	“Multiple-Result Parametric Record (MPR)” on page 2-38
MRR	“Master Results Record (MRR)” on page 2-10
PCR	“Part Count Record (PCR)” on page 2-11
PGR	“Pin Group Record (PGR)” on page 2-17
PIR	“Part Information Record (PIR)” on page 2-29
PLR	“Pin List Record (PLR)” on page 2-18
PMR	“Pin Map Record (PMR)” on page 2-15
PRR	“Part Results Record (PRR)” on page 2-30
PTR	“Parametric Test Record (PTR)” on page 2-34
RDR	“Retest Data Record (RDR)” on page 2-21
SBR	“Software Bin Record (SBR)” on page 2-14
SDR	“Site Description Record (SDR)” on page 2-22
TSR	“Test Synopsis Record (TSR)” on page 2-32
WCR	“Wafer Configuration Record (WCR)” on page 2-27

Table 2-1. ATDF record types by abbreviation

Abbreviation	Record Type
WIR	“Wafer Information Record (WIR)” on page 2-24
WRR	“Wafer Results Record (WRR)” on page 2-25

File Attributes Record (FAR)

Function: Contains the information necessary to determine how to decode the ATDF data contained in the file.

Table 2-2. File Attributes Record (FAR) fields

ATDF Field	STDF Field	Description	Req?
Header:		FAR:	Yes
Data File Type:	CPU_TYPE	A The letter A indicates that this is an ATDF file. Programs reading this file will be able to determine what type of file it is by looking at the fifth byte of the first record in the file. This is the same location as the CPU_TYPE field in the STDF.	Yes
STDF Version:	STDF_VER	4 This number indicates that this version of ATDF corresponds with version 4 of the STDF.	Yes
ATDF Version:		2 This number indicates the version of the ATDF specification.	Yes
Scaling Flag:		This character indicates whether parametric test results in the PTRs and MPRs are scaled or unscaled. Valid values for this field are: S = Scaled U = Unscaled If the flag is missing, the data is assumed to be scaled.	No

Location: The FAR is required and must be the first record in the ATDF file.

Notes: The sixth character of the record, which is the first separator in the file, specifies the separator that is to be used throughout the rest of the file. The default is the vertical bar. You can use this first separator to set a non-default character. See the description of "Field Separators" on page 1-4.

Sample: `FAR:A|4|2|U`

Audit Trail Record (ATR)

Function: An audit trail record is used to record any operation that alters the contents of the file. The name of the program and all its parameters should be recorded in the ASCII field provided in this record. Typically, this record will be used to track filter programs that have been applied to the data

Table 2-3. Audit Trail Record (ATR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`ATR:`	Yes
Modification Timestamp:	`MOD_TIM`	The time and date when the file was modified.	No
Command Line:	`CMD_LINE`	Command line of the program that modified the file.	No

Frequency: Optional. One for each filter or other data transformation program applied to the STDF data.

Location: If present, ATRs must immediately follow the FAR.

The filter program that writes the altered file must write its ATR immediately after the FAR (and hence before any other ATRs that may be in the file). In this way, multiple ATRs will be in reverse chronological order.

Sample:

```
ATR:0:03:00 3-SEP-1992|bin_filter 7,9-12
```

Master Information Record (MIR)

Function: The MIR and the MRR (Master Results Record) contain all the global information that is to be stored for a tested lot of parts. Each ATDF file must have exactly one MIR, immediately following the FAR and any ATRs, if ATRs are used. This will allow any data reporting, analysis, or filtering programs access to the header information in the shortest possible amount of time.

Table 2-4. Master Information Record (MIR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`MIR:`	Yes
Lot ID:	`LOT_ID`	Customer-specified Lot ID for the lot of devices whose data is contained in the file.	Yes
Part Type:	`PART_TYP`	Part or device type being tested in this lot.	Yes
Job Name:	`JOB_NAM`	Name of the test program or job plan testing the devices.	Yes
Node ID:	`NODE_NAM`	Name or number of the tester or other node that created the data.	Yes
Tester Type:	`TSTR_TYP`	Tester type and/or model number used in testing the parts.	Yes
Setup Time:	`SETUP_T`	Time and date setup began.	Yes
Start Time:	`START_T`	Time and date testing began on the first device.	Yes
Operator Name:	`OPER_NAM`	Operator name or ID (at setup time).	Yes

Table 2-4. Master Information Record (MIR) fields

ATDF Field	STDF Field	Description	Req?
Test Mode:	MODE_COD	Test mode code (such as production, maintenance, and debug) This field is truncated to the first character on conversion to STDF. Currently defined values in the STDF are: A = AEL (Automatic Edge Lock) mode C = Checker mode D = Development / Debug test mode E = Engineering mode (same as development) M = Maintenance test mode P = Production test mode Q = Quality Control Teradyne may define other alphabetic codes in the future. To avoid conflict with Teradyne-defined codes, any user-defined codes should be numeric, rather than alphabetic.	Yes
Station Number:	STAT_NUM	Tester station number used in testing the devices.	Yes
Sublot ID:	SBLOT_ID	Customer-specified Sublot ID for the lot of devices whose data is contained in the file.	No
Test Code:	TEST_COD	Test Conditions code. A user-defined field specifying the phase of device testing or test conditions: for example, “WAFER” for wafer test, “CHAR” for characterization or “HOT” for hot test. Valid characters are 0-9 and A-Z.	No
Retest Code:	RTST_COD	Lot Retest Code. Indicates whether the lot of parts has been previously tested under the same test conditions. Recommended values are: Y = Entire lot has been retested N = Lot has not been previously tested 0 - 9 = Number of times lot has been previously tested. This field is truncated to the first character on conversion to STDF.	No
Job Rev:	JOB_REV	Revision number or code for the test program or job plan.	No

Table 2-4. Master Information Record (MIR) fields

ATDF Field	STDF Field	Description	Req?
Executive Type:	EXEC_TYP	Tester executive software type.	No
Exec Version:	EXEC_VER	Version number of tester executive.	No
Protect Code:	PROT_COD	Data protection code indicates the protection desired for the test data being stored. Valid values are the characters 0-9 and A-Z. Note that if multiple characters are placed in this field, it will be truncated to the first character on conversion to STDF.	No
Command Mode:	CMOD_COD	Command mode of the tester during testing of the parts. The user or tester executive software defines the command mode values. Valid values are the characters 0-9 and A-Z. Note that if multiple characters are placed in this field, it will be truncated to the first character on conversion to STDF.	No
Burn-in Time:	BURN_TIM	Burn-in time in minutes.	No
Test Temp:	TST_TEMP	Test temperature. This can be stored as degrees Celsius, Fahrenheit, Kelvin, or whatever. It can also be expressed in terms like “HOT”, “ROOM”, or “COLD” if that is preferred.	No
User Text:	USER_TXT	Generic user text field.	No
Auxiliary File:	AUX_FILE	Name of a file containing auxiliary data related to the data in this file.	No
Package Type:	PKG_TYP	Package type.	No
Family ID:	FAMLY_ID	Product family ID.	No
Date Code:	DATE_COD	Date code.	No
Facility ID:	FACIL_ID	Test facility ID.	No
Floor ID:	FLOOR_ID	Test floor ID.	No
Process ID:	PROC_ID	Fabrication process ID.	No
Operation Freq:	OPER_FRQ	Operation frequency or step.	No
Spec Name:	SPEC_NAM	Test specification name.	No

Table 2-4. Master Information Record (MIR) fields

ATDF Field	STDF Field	Description	Req?
Spec Version:	SPEC_VER	Test specification version number.	No
Flow ID:	FLOW_ID	Test flow ID.	No
Setup ID:	SETUP_ID	Test setup ID.	No
Design Rev:	DSGN_REV	Device design revision.	No
Eng. Lot ID:	ENG_ID	Engineering Lot ID.	No
ROM Code ID:	ROM_COD	ROM code ID.	No
Serial number:	SERL_NUM	Tester serial number.	No
Super Name:	SUPR_NAM	Name of the responsible supervisor at the time testing began.	No

Frequency: Always required. One per data stream.

Location: Must be located immediately after the File Attributes Record (FAR) and the Audit Trail Records (ATR), if present.

Sample:

```
MIR:A3002B|80386|80386HOT|akbar|J971
    |8:14:59 23-JUL-1992|8:23:02 23-JUL-1992|Sandy
    |P|1|2B|HOT|N|3.1.2|IG900|2.4|||300|100||
    386_data.txt|ceramic|386|wk23||MPU2||||||386HOT|
    3S|||A42136S|JOAN_S
```

Master Results Record (MRR)

Function: The Master Results Record is a logical extension of the Master Information Record (MIR). The data can be thought of as belonging with the MIR, but not available when the tester writes the MIR information. Each data stream must have exactly one MRR as the last record of the data stream.

Table 2-5. Master Results Record (MRR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`MRR:`	Yes
Finish Time:	`FINISH_T`	Date and time that the last part was tested.	Yes
Disposition:	`DISP_COD`	Lot disposition code supplied by the user to indicate the disposition of the lot of parts (or of the tester itself in the case of checker or AEL data). The meaning of Disposition: values is user-defined. A valid value is an ASCII alphanumeric character (0-9 or A-Z). Note that in STDF this field is a single character; the data in this field will therefore be truncated to the first character when translating to STDF.	No
User Descrip.:	`USR_DESC`	Lot description supplied by the user. May be any ASCII string.	No
Exec Descrip.:	`EXC_DESC`	Lot description supplied by the tester executive. May be any ASCII string.	No

Frequency: Always required. Exactly one MRR per data stream.

Location: The MRR must be the last record in the data stream.

Sample:

```
MRR:12:17:12 23-JUL-1992|H|Handler problems
    |Yield Alarm
```

Part Count Record (PCR)

Function: The Part Count Record contains the part count totals for one or all test sites. Each data stream must have at least one PCR to show the part count

Table 2-6. Part Count Record (PCR) fields

ATDF Field	STDF Field	Description	Req?
Header:		PCR:	Yes
Head Number:	HEAD_NUM	Test head number. If this PCR contains a summary of the part counts for all test sites, this field must be empty. Otherwise, it is required.	
Site Number:	SITE_NUM	Test site number. If this PCR contains a summary of the part counts for all test sites, this field must be empty. Otherwise, it is required.	
Part Count:	PART_CNT	Number of parts or devices that were tested.	Yes
Retest Count:	RTST_CNT	Number of parts or devices that were retested.	No
Abort Count:	ABRT_CNT	Number of parts or devices that aborted during testing.	No
Good Count:	GOOD_CNT	Number of good (passed) parts or devices tested.	No
Funct. Count:	FUNC_CNT	Number of functional parts or devices tested.	No

Frequency: There must be at least one PCR in the file: either one summary PCR for all test sites (Head Number and Site Number are both empty), or one PCR for each head/site combination, or both.

Location: Anywhere in the data stream following the MIR (and RDR and SDRs, if present) and before the MRR. When data is being recorded in real time, this record will usually appear near the end of the data stream.

Samples:

```
PCR:2|1|497|5|11|212|481        (for Head 2, Site 1)
PCR:||3976|54|76|2311|3809      (for all test sites)
```

Hardware Bin Record (HBR)

Function: The Hardware Bin Record stores a count of parts “physically” placed in a particular bin after testing. (In wafer testing, “physical” binning is not the actual transfer of the chip, but rather is represented by a drop of ink or an entry in a wafer map file.) This bin count can be for a single test site (when parallel testing) or a total for all test sites.

he ATDF specification also supports a Software Bin Record (SBR), for logical binning categories. A part is “physically” placed in a hardware bin after testing. A part can be “logically” associated with a software bin during or after testing.

Table 2-7. Hardware Bin Record (HBR) fields

ATDF Field	STDF Field	Description	Req?
Header:		HBR:	Yes
Head Number:	HEAD_NUM	Test head number. If this HBR contains a summary of the hardware bins for all test sites, this field must be empty. Otherwise, it is required.	
Site Number:	SITE_NUM	Test site number. If this HBR contains a summary of the hardware bins for all test sites, this field must be empty. Otherwise, it is required.	
Bin Number:	HBIN_NUM	Hardware bin number. Legal values are in the range 0 to 32767.	Yes
Bin Count:	HBIN_CNT	Number of parts placed in the hardware bin.	Yes
Pass or Fail:	HBIN_PF	This field indicates whether the hardware bin was a passing or failing bin. Valid values for this field are “P” for pass and “F” for fail. Note that the data in this field will be truncated to the first character when translating to STDF.	No
Bin Name:	HBIN_NAM	Name of the hardware bin.	No

Frequency: One per hardware bin for each site. One per hardware bin for bin totals. Can be included to name unused bins.

Location: Anywhere in the data stream following the MIR (and RDR and SDRs, if present) and before the MRR. When data is recorded in real time, this record usually appears near the end of the data stream.

Samples:

```
HBR:2|1|6|212|F|SHORT
```
(for Head 2, Site 1)

```
HBR:||1|1346|P|PASSED
```
(for all test sites)

Software Bin Record (SBR)

Function: The Software Bin Record stores a count of parts associated with a particular logical bin after testing. This bin count can be for a single test (when parallel testing) or a total for all test sites.

The ATDF specification also supports a Hardware Bin Record (HBR), for actual physical binning. A part is “physically” placed in a hardware bin after testing. A part can be “logically” associated with a software bin during or after testing.

Table 2-8. Software Bin Record (SBR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`SBR:`	Yes
Head Number:	`HEAD_NUM`	Test head number. If this SBR contains a summary of the software bins for all test sites, this field must be empty. Otherwise, it is required.	
Site Number:	`SITE_NUM`	Test site number. If this SBR contains a summary of the software bins for all test sites, this field must be empty. Otherwise, it is required.	
Bin Number:	`SBIN_NUM`	Software bin number. Legal values are in the range 0 to 32767.	Yes
Bin Count:	`SBIN_CNT`	Number of parts placed in the software bin.	Yes
Pass or Fail:	`SBIN_PF`	This field indicates whether the software bin was a passing or failing bin. Valid values for this field are “P” for pass and “F” for fail. Note that the data in this field will be truncated to the first character when translating to STDF.	No
Bin Name:	`SBIN_NAM`	Name of the software bin.	No

Frequency: One per software bin used. Can be included to name unused bins.

Location: Anywhere in the data stream following the MIR (and RDR and SDRs, if present) and before the MRR. When data is recorded in real time, this record usually appears near the end of the data stream.

Samples:

```
SBR:1|2|74|14|F|NOTIFY PRODUCT ENG      (for Head 1, Site 2)
SBR:||1|1346|P|PASSED                   (for all test sites)
```

Pin Map Record (PMR)

Function: The Pin Map Record (PMR) provides indexing of tester channel names, and maps them to physical and logical pin names. Each PMR defines the information for a single channel/pin combination.

Table 2-9. Pin Map Record (PMR) fields

ATDF Field	STDF Field	Description	Req?
Header:		PMR:	Yes
PMR Index:	PMR_INDX	The PMR Index is a number used to associate the channel and pin name information with data in the FTR or MPR. Reporting programs can then look up the PMR index and choose which of the three associated names they will use. The range of legal PMR indexes is 1 - 32767. In the STDF file, the size of the FAIL_PIN and SPIN_MAP arrays in the FTR is directly proportional to the highest PMR index number. Therefore, it is important to start PMR indexes with a low number and use consecutive numbers if possible.	Yes
Channel Type:	CHAN_TYP	The channel type values are tester-specific. Please refer to the tester documentation for a list of the valid tester channel types and codes.	No
Channel Name:	CHAN_NAM	Tester channel name.	No
Pin Name:	PHY_NAM	Physical name of the device pin.	No
Logical Name:	LOG_NAM	Logical name of the device pin.	No
Head Number:	HEAD_NUM	Test head number associated with this pin. If the test system does not support parallel testing and does not have a standard way of identifying its single test head, this field should be empty.	No
Site Number:	SITE_NUM	Test site number associated with this pin. If the test system does not support parallel testing and does not have a standard way of identifying its single test site, this field should be empty.	No

Frequency:	One per channel/pin combination used in the test program. Reuse of a PMR index number is not permitted.
Location:	After the MIR (and the RDR and SDRs, if present) and before the first PGR, PLR, FTR, or MPR that uses this record's PMR Index value.
Sample:	`PMR:2|A|1-7|GND|MAIN GROUND|2|1`

Pin Group Record (PGR)

Function: The Pin Group Record (PGR) associates a name with a group of pins.

Table 2-10. Pin Group Record (PGR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`PGR:`	Yes
Group Index:	`GRP_INDX`	The pin group index is a number used to identify the pin group and to associate information in this record with corresponding data in the Pin List Record (PLR). The pin group index must be unique for each PGR. The range of legal group index numbers is 32768 - 65535.	Yes
Group Name:	`GRP_NAM`	Name of the pin group.	No
Index Array:	`PMR_INDX`	Array of PMR indexes for pins in the group. Indexes will be separated from each other by commas. The order of the PMR indexes should be from most significant to least significant bit in the pin group (regardless of the order of PMR index numbers).	No

Frequency: One per pin group defined in the test program.

Location: After all the PMRs whose PMR index values are listed in the Index Array array of this record; and before the first PLR that uses this record's Group Index value.

Sample:
```
PGR:12|Data Out|5,6,7,8,9,10,11,12
```

Pin List Record (PLR)

Function: The Pin List Record (PLR) defines the current display radix and operating mode for a pin or pin group.

Table 2-11. Pin List Record (PLR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`PLR:`	Yes
Index Array:	`GRP_INDX`	Array of pin or pin group indexes. Each index will be a separated from the next by a comma.	Yes
Mode Array:	`GRP_MODE`	Array of pin group operating modes. Each mode will be a separated from the next by a comma. This array should have the same number of entries as the Index Array. The following hexadecimal values, represented in ASCII, are valid for the pin group mode: 00 = Unknown 10 = Normal 20 = SCIO (Same Cycle I/O) 21 = SCIO Midband 22 = SCIO Valid 23 = SCIO Window Sustain 30 = Dual drive (Two drive bits per cycle) 31 = Dual drive Midband 32 = Dual drive Valid 33 = Dual drive Window Sustain Unused pin group modes in the range of 1 through 32767 are reserved for future use. Pin group modes in 32768 through 65535 are available for customer use.	No

Table 2-11. Pin List Record (PLR) fields

ATDF Field	STDF Field	Description	Req?
Radix Array:	GRP_RADX	Array of pin group display radixes. Each radix will be separated from the next by a comma. This array should have the same number of entries as the Index Array. The following symbols, represented in ASCII, are valid for the pin group display radix: B = Display in Binary O = Display in Octal D = Display in Decimal H = Display in Hexadecimal S = Display as symbolic Leave the field empty to indicate that the program default display radix should be used.	No
Program State:	PGM_CHAL, PGM_CHAR	Programmed state codes are used to display the programmed state in the FTR or MPR record. Use of this field makes it possible to store tester-dependent display representations in a tester-independent format. The programmed state field consists of an array of lists of state codes. One or two characters may be used to represent each entry in a programmed state list (one for each state for which the pin or pin group can be programmed). If one character is used, then on conversion to STDF, it will be stored in the PGM_CHAR field. If two characters are used, then the first will be stored in PGM_CHAL and the second will be stored in PGM_CHAR. If more than two characters are in the item, only the first two will be used. Entries in the programmed state list must be separated by commas. The programmed state array will have one list for each entry of the Index Array. Lists will be separated within the array using the slash “/” character.	No

Table 2-11. Pin List Record (PLR) fields

ATDF Field	STDF Field	Description	Req?
Returned State:	RTN_CHAL, RTN_CHAR	Returned state codes are used to display the returned state in the FTR or MPR record. Use of this array makes it possible to store tester-dependent display representations in a tester-independent format. The returned state field consists of an array of lists of state codes. One or two characters may be used to represent each entry in a returned state list (one for each state for which the pin or pin group can output). If one character is used, then on conversion to STDF, it will be stored in the RTN_CHAR field. If two characters are used, then the first will be stored in RTN_CHAL and the second will be stored in RTN_CHAR. If more than two characters are in the item, only the first two will be used. Entries in the returned state list must be separated by commas. The returned state array will have one list for each entry of the Index Array. Lists will be separated within the array using the slash "/" character.	No

Frequency: One or more whenever the usage of a pin group changes in the test program.

Location: After all the PMRs and PGRs whose PMR index values and pin group index values are listed in the Index Array array of this record; and before the first FTR that references pins or pin groups whose modes are defined in this record.

Sample:

```
PLR:2,3,6|20,20,21|H,H,H|H,L,L/H,H,H/L,L,L
   |1,0,M/1,0,H/M,L,H
```

Retest Data Record (RDR)

Function: This record signals that the data in this ATDF file is for retested parts. The data in this record, combined with the information in the MIR, tells data filtering programs what data to replace when processing retest data.

Table 2-12. Retest Data Record (RDR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`RDR:`	Yes
Retest bins:	`RTST_BIN`	Array of bin numbers being retested. Bin numbers in the array will be separated by commas. If all bins are being retested, this field should be omitted and the record will consist of only the record header. Otherwise the field is required.	

Frequency: Optional. One per data stream.

Location: If present, this record must immediately follow the Master Information Record (MIR).

Sample: `RDR:4,5,7`

Note: The Lot ID, Sublot ID, and Test Code of the current STDF file should match those of the ATDF or STDF file that is being retested in order for the data to be properly merged at a later time.

Site Description Record (SDR)

Function: This record contains the configuration information for one or more test sites, connected to one test head, that compose a site group.

Table 2-13. Site Description Record (SDR) fields

ATDF Field	STDF Field	Description	Req?
Header:		SDR:	Yes
Head Number:	HEAD_NUM	The test head number associated with all the sites described in this record.	Yes
Site Group:	SITE_GRP	The site group number (called a station number in some testers) is unique for the group of sites described by this record. Note that this is different from the station number described in the MIR, which refers to a software station only.	Yes
Site Array:	SITE_NUM	Array of test site numbers. Each test site number will be separated by a comma in the array.	Yes
Handler Type:	HAND_TYP	Type of handler or prober connected to this group of sites.	No
Handler ID:	HAND_ID	ID of the handler or prober connected to this group of sites.	No
Card Type:	CARD_TYP	Type of probe card connected to this group of sites.	No
Card ID:	CARD_ID	ID of the probe card connected to this group of sites.	No
Load Type:	LOAD_TYP	Type of load board connected to this group of sites.	No
Load ID:	LOAD_ID	ID of the load board connected to this group of sites.	No
DIB Type:	DIB_TYP	Type of device interface board connected to this group of sites.	No
DIB ID:	DIB_ID	ID of the device interface board connected to this group of sites.	No
Cable Type:	CABL_TYP	Type of interface cable connected to this group of sites.	No

Table 2-13. Site Description Record (SDR) fields

ATDF Field	STDF Field	Description	Req?
Cable ID:	CABL_ID	ID of the interface cable connected to this group of sites.	No
Contactor Type:	CONT_TYP	Type of handler contactor connected to this group of sites.	No
Contactor ID:	CONT_ID	ID of the handler contactor connected to this group of sites.	No
Laser Type:	LASR_TYP	Type of laser trimmer connected to this group of sites.	No
Laser ID:	LASR_ID	ID of the laser trimmer connected to this group of sites.	No
Extra Type:	EXTR_TYP	Type of any other equipment connected to this group of sites.	No
Extra ID:	EXTR_ID	ID of any other equipment connected to this group of sites.	No

Frequency: One for each site or group of sites that is configured differently.

Location: Immediately following the Master Information Record (MIR) and Retest Data Record (RDR), if present.

Sample:
```
SDR:2|4|5,6,7,8|Delta Flex|D511||B101|17
```

Wafer Information Record (WIR)

Function: The Wafer Information Record acts mainly as a marker to indicate where testing of a particular wafer begins for each wafer tested by the test program. The WIR and the Wafer Results Record (WRR) bracket all the stored information pertaining to one tested wafer. This record is used only when testing at wafer probe.

Table 2-14. Wafer Information Record (WIR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`WIR:`	Yes
Head Number:	`HEAD_NUM`	Test head number.	Yes
Start Time:	`START_T`	Time and date that the first die on the wafer was tested.	Yes
Site Group:	`SITE_GRP`	The site group number of the SDR describing this site. This is a means of relating the wafer information to the configuration of the equipment used to test it.	No
Wafer ID:	`WAFER_ID`	Wafer identification code. This field is included for compatibility with the STDF WIR. The Wafer ID field of the WRR is more generally used. It is recommended that the user omit this field from WIRs to help control the overall size of the ATDF file, and that the Wafer Identification code be provided in the WRR.	No

Frequency: One per wafer tested. Used only when wafer testing.

Location: Anywhere in the data stream following the MIR (and RDR and SDRs, if present) and before the MRR. Sent before testing each wafer.

Sample: `WIR:1|8:23:02 23-JUL-1992|2`

Wafer Results Record (WRR)

Function: The Wafer Results Record contains the results information relating to each wafer tested by the test program. The WRR and the Wafer Information Record (WIR) bracket all the stored information pertaining to one tested wafer. This record is used only when testing at wafer probe.

Table 2-15. Wafer Results Record (WRR) fields

ATDF Field	STDF Field	Description	Req?
Header:		WRR:	Yes
Head Number:	HEAD_NUM	Test head number.	Yes
Finish Time:	FINISH_T	Time and date that the last die on the wafer was tested.	Yes
Part Count:	PART_CNT	Number of parts or devices that were tested.	Yes
Wafer ID:	WAFER_ID	The wafer identification code, although optional, is strongly recommended to make the resultant data files as useful as possible. A wafer ID in the WRR supersedes any wafer ID found in the WIR.	No
Site Group:	SITE_GRP	The site group number of the SDR describing this site. This is a means of relating the wafer information to the configuration of the equipment used to test it. The site group number must match the one in the corresponding WIR.	No
Retest Count:	RTST_CNT	Number of parts or devices that were retested.	No
Abort Count:	ABRT_CNT	Number of parts or devices that aborted during testing.	No
Good Count:	GOOD_CNT	Number of good (passed) parts or devices tested.	No
Funct. Count:	FUNC_CNT	Number of functional parts or devices tested.	No
Fab Wafer ID:	FABWF_ID	Wafer ID used in the fabrication process. This facilitates tracking of wafers and correlation of yield with fabrication variations.	No

Table 2-15. Wafer Results Record (WRR) fields

ATDF Field	STDF Field	Description	Req?
Frame ID:	FRAME_ID	The wafer frame ID facilitates the tracking of wafers once the wafer has been through the saw step and the wafer ID is no longer readable on the wafer itself. This is an important piece of information for implementing an inkless binning scheme.	No
Mask ID:	MASK_ID	Wafer Mask ID. This is useful for correlating yield problems with defective masks.	No
User Descrip.:	USR_DESC	Lot description supplied by the user. May be any ASCII string.	No
Exec Descrip.:	EXC_DESC	Lot description supplied by the tester executive. May be any ASCII string.	No

Frequency: One per wafer tested. Used only for wafer testing.

Location: Anywhere in the data stream after the corresponding WIR and before the MRR. Sent after testing each wafer.

Sample:

```
WRR:1|11:02:42 23-JUL-1992|492|W01|3|102|214|2
   |131|MOS-4|F54|S3-1|Glass buildup on prober
   |Yield alarm on wafer W01
```

Wafer Configuration Record (WCR)

Function: Contains the configuration information for wafers tested by the test program. The WCR provides the dimensions and orientation information for all wafers and dice in the lot. This record is used only when testing at wafer probe time.

Table 2-16. Wafer Configuration Record (WCR) fields

ATDF Field	STDF Field	Description	Req?
Header:		WCR:	Yes
Wafer Flat:	WF_FLAT	Orientation of the wafer flat. These are the legal values: U = Up D = Down L = Left R = Right If more than one character appears in this field, it will be truncated to the first character during conversion to STDF.	No
Positive X:	POS_X	Positive X direction on the wafer. Valid values are: L = Left R = Right If more than one character appears in this field, it will be truncated to the first character during conversion to STDF.	No
Positive Y:	POS_Y	Positive Y direction on the wafer. Legal values are: U = Up D = Down If more than one character appears in this field, it will be truncated to the first character during conversion to STDF.	No
Wafer Size:	WAFR_SIZ	Wafer diameter in units from Wafer Units:.	No
Die Height:	DIE_HT	Height of each die in units from Wafer Units:	No
Die Width:	DIE_WID	Width of each die in units from Wafer Units:	No

Table 2-16. Wafer Configuration Record (WCR) fields

ATDF Field	STDF Field	Description	Req?
Wafer Units:	WF_UNITS	Units used for measuring wafer dimensions: 1 = Units are inches 2 = Units are centimeters 3 = Units are in millimeters 4 = Units are in mils	No
Center X:	CENTER_X	X Coordinate at the center die on the wafer.	No
Center Y:	CENTER_Y	Y Coordinate at the center die on the wafer.	No

Frequency: One per ATDF file. Used only when wafer testing.

Location: Anywhere in the data stream following the MIR (and RDR and SDRs, if present) and before the MRR.

Sample:
```
WCR:D|R|D|5|.3|.25|1|23|19
```

Part Information Record (PIR)

Function: The Part Information Record acts as a marker to indicate where testing of a particular part or device begins for each part tested by the test program. The PIR and the Part Results Record (PRR) bracket all the stored information pertaining to one tested part.

Table 2-17. Part Information Record (PIR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`PIR:`	Yes
Head Number:	`HEAD_NUM`	Test head number. If there is only one test head on the tester, use 1 in this field. When parallel testing, this field and the site number field are used to associate individual datalogged results (PTRs, FTRs, or MPRs) with a PIR/PRR pair.	Yes
Site Number:	`SITE_NUM`	Test site number. If the tester does not support parallel testing, use 1 in this field. When parallel testing, this field and the site number field are used to associate individual datalogged results (PTRs, FTRs, or MPRs) with a PIR/PRR pair.	Yes

Frequency: One per part or device tested.

Location: Anywhere in the data stream after the MIR (and RDR and SDR, if present) and before the corresponding PRR. Sent before testing each part.

Sample: `PIR:2|1`

Part Results Record (PRR)

Function: The Part Results Record contains the results information relating to each part tested by the test program. The PRR and the Part Information Record (PIR) bracket all the stored information pertaining to one tested part.

Table 2-18. Part Results Record (PRR) fields

ATDF Field	STDF Field	Description	Req?
Header:		PRR:	Yes
Head Number:	HEAD_NUM	Test head number. If there is only one test head on the tester, use 1 in this field. It is very important that this number match the head number in the corresponding PIR.	Yes
Site Number:	SITE_NUM	Test site number. If the tester does not support parallel testing, use 1 in this field. It is very important that this number match the site number in the corresponding PIR.	Yes
Part ID:	PART_ID	Part identification number.	
Num. of Tests:	NUM_TEST	Number of tests executed on this device.	Yes
Pass/Fail Code:	PART_FLG bits 3 & 4	Legal values for the pass/fail code are: P = Part passed F = Part failed	
Hardware Bin:	HARD_BIN	Hardware bin number assigned to this device by the test program.	Yes
Software Bin:	SOFT_BIN	Software bin number assigned to this device by the test program.	No
X Coordinate:	X_COORD	X coordinate of the device on the wafer. Highly recommended for wafer probe.	No
Y Coordinate:	Y_COORD	Y coordinate of the device on the wafer. Highly recommended for wafer probe.	No

Table 2-18. Part Results Record (PRR) fields

ATDF Field	STDF Field	Description	Req?
Retest Code:	`PART_FLG bit 0 or 1`	The presence of a value in this field indicates that this is a retested device, and that the data in this record supersedes data for the device with the same identifier. The actual value of this field indicates that the superseded device data is identified by: I = Same Part ID C = Same X/Y Coordinates If not a retest, this field is empty.	No
Abort Code:	`PART_FLG bit 2`	If an abort occurred during testing of this device, the field will contain the character Y.	No
Test Time:	`TEST_T`	Time in milliseconds to test the part.	No
Part Text:	`PART_TXT`	Any descriptive text that will be useful in analyzing the data associated with this part.	No
Part Fix Data:	`PART_FIX`	Repair information as hexadecimal digits represented in ASCII.	No

Frequency: One per part tested.

Location: Anywhere in the data stream after the corresponding PIR and before the MRR. Sent after testing each part.

Sample:

```
PRR:2|1|13|78|F|0|17|-2|7|||644|
   Device at edge of wafer|F13C20
```

Test Synopsis Record (TSR)

Function: Contains the test execution, failure counts, and other statistics for one parametric or functional test in the test program.

Table 2-19. Test Synopsis Record (TSR) fields

ATDF Field	STDF Field	Description	Req?
Header:		TSR:	Yes
Head Number:	HEAD_NUM	Test head number. If this TSR contains a summary of the test counts for all test sites, this field must be empty. Otherwise, it is required.	
Site Number:	SITE_NUM	Test site number. If this TSR contains a summary of the test counts for all test sites, this field must be empty. Otherwise, it is required.	
Test Number:	TEST_NUM	Test number. Test numbers must be unique.	Yes
Test Name:	TEST_NAM	Test name.	No
Test Type:	TEST_TYP	Test type. Valid values are: P = Parametric test F = Functional test M = Multiple parametric test	No
Execut. Count:	EXEC_CNT	Number of test executions. Optional, but strongly recommended.	No
Fail Count:	FAIL_CNT	Number of test failures. Optional, but strongly recommended.	No
Alarm Count:	ALRM_CNT	Number of alarming tests. Optional, but strongly recommended.	No
Sequencer Name:	SEQ_NAME	Sequencer (program segment) name.	No
Test Label:	TEST_LBL	Test label or text.	No
Test Time:	TEST_TIM	Average execution time for this test.	No
Test Min:	TEST_MIN	Minimum test result value recorded. Parametric tests only.	No

Table 2-19. Test Synopsis Record (TSR) fields

ATDF Field	STDF Field	Description	Req?
Test Max:	TEST_MAX	Maximum test result value recorded. Parametric tests only.	No
Test Sums:	TST_SUMS	Sum of all test result values.	No
Test Squares:	TST_SQRS	Sum of squares of test result values.	No

Frequency: One for each test executed in the test program. May optionally be used to identify unexecuted tests.

Location: Anywhere in the data stream following the MIR (and RDR and SDRs, if present) and before the MRR. When test data is being generated in real time, these records will appear after the last PRR.

Sample:
```
TSR:2|2|600|Leakage|P|413|92|3||DC_TESTS|0.005|0.1
    |7.2|1280.3|4329.5
```

Parametric Test Record (PTR)

Function: Contains the results of a single execution of one parametric test in the test program. The first occurrence of a PTR also establishes the default values for all semi-static information about the test, such as limits, units and scaling. The PTR is related to the Test Synopsis Record (TSR) by the test number.

Table 2-20. Parametric Test Record (PTR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`PTR:`	Yes
Test Number:	`TEST_NUM`	Test number. Test numbers must be unique.	Yes
Head Number:	`HEAD_NUM`	Test head number. If there is only one test head on the tester, use 1 in this field.	Yes
Site Number:	`SITE_NUM`	Test site number. If the tester does not support parallel testing, use 1 in this field.	Yes
Test Result:	`RESULT`	Parametric test value. If the Scaling Flag: in the FAR is set to S (scaled data), then the value in this field must be scaled according to the STDF scaling rules (see note below). If the FAR Scaling Flag: is set to U (unscaled data), then the value need not be scaled. In either case, the units field must be set appropriately for the type of data in this field.	No
Pass/Fail Flag:	`TEST_FLG bits 6 & 7 PARM_FLG bit 5`	The Pass/Fail Flag indicates the result of the test. If the flag is empty, the test completed without a pass/fail indication from the test program. Legal values are: A = Test passed alternate limits F = Test failed P = Test passed standard limits empty = Test completed without pass/fail indication	Yes

Table 2-20. Parametric Test Record (PTR) fields

ATDF Field	STDF Field	Description	Req?

Alarm Flags:	`TEST_FLG bits 0, 2, 3, 4, & 5`
	
	`PARM_FLG bits 0, 1, 2, 3, & 4`
		This field may contain zero or more flags. Each flag indicates an alarm or error condition that occurred during the test execution. Legal values are:
		
		A = Alarm detected
		D = Drift error
		H = Measured value higher than test limit
		L = Measured value lower than low limit
		N = Test was not executed
		O = Oscillation detected
		S = Scale error
		T = Time-out occurred
		U = Test result is unreliable
		X = Test aborted
			No

Test Text:	`TEST_TXT`	User-defined text string providing information about the test execution.	No
Alarm ID:	`ALARM_ID`	Name or ID of the alarm or alarms that were triggered. This field should be used only if the alarm flag is also set.	No
Limit Compare:	`PARM_FLG bits 6 & 7`	By default, limits are compared such that the test fails when the low limit is greater than the test result or the high limit is less than the test result. If that is the case, this field should be empty. Otherwise, the field should contain one or two letters indicating: L = Low limit comparison was >= H = High limit comparison was <=	No
Test Units:	`UNITS`	Test units. This field will be truncated to seven characters on conversion to STDF. If the FAR Scaling Flag: is set for scaled test data, this field must represent whole units and a units prefix is not allowed.	No
Low Limit:	`LO_LIMIT`	Low test limit value. If the FAR Scaling Flag: is set for scaled test data, the value in this field should be scaled to whole units according to the value in the Test Units field.	No

Table 2-20. Parametric Test Record (PTR) fields

ATDF Field	STDF Field	Description	Req?
High Limit:	HI_LIMIT	High test limit value. If the FAR Scaling Flag: is set for scaled test data, the value in this field should be scaled to whole units according to the value in the Test Units field.	No
Result Format:	C_RESFMT	ANSI C string used for formatting the test result when printing.	No
Lo Limit Fmt:	C_LLMFMT	ANSI C string used for formatting the low limit when printing.	No
Hi Limit Fmt:	C_HLMFMT	ANSI C string used for formatting the high limit when printing.	No
Lo Spec. Limit:	LO_SPEC	Low specification limit value. Useful in calculating process capability.	No
Hi Spec. Limit:	HI_SPEC	High specification limit value. Useful in calculating process capability.	No
Result scale:	RES_SCAL	Test Results scaling exponent. If this ATDF file contains unscaled data, this field is not used.	No
Lo Limit Scale:	LLM_SCAL	Low limit scaling exponent. If this ATDF file contains unscaled data, this field is not used.	No
Hi Limit Scale:	HLM_SCAL	High limit scaling exponent. If this ATDF file contains unscaled data, this field is not used.	No

Note on Scaling

The converter performs automatic scaling on test results in the PTR if the Scaling Flag: in the FAR is set to U (unscaled data). The scaling process is irreversible and will occur whether the output file is in STDF or ATDF format. The prefix will be stripped from the Test Units: field; the test result, the high and low test limits, and the high and low spec limits will all be scaled; and the three scaling fields will be generated from the Test Units: field. The converter never outputs unscaled data. For more information on the use of Results and Limits Scale, Results and Limits Formats, and Units fields, refer to the section “Storing and Displaying Test Data” in description of the PTR in the STDF specification.

Note on Default Data

All PTR data starting with the Test Units: field has a special function in the ATDF file. The first PTR for each test will have these fields filled in. The values in these fields will be the default values for each subsequent PTR with the same test number. If the field is filled in for subsequent PTRs, that value will override the default. Otherwise the default will be used.

This method replaces use of the PDR in STDF V3. For character strings, it is possible to override the default with a null value by setting the string to a single space.

Unless the default has been overridden, the default data fields should be omitted to save space in the ATDF file.

If the PTR is not associated with a test execution (that is, contains only default information), the Test Not Executed flag in the Alarm Flags field must be set.

Frequency: One per parametric test execution.

Location: Under normal circumstances, the PTR can appear anywhere in the data stream after the corresponding Part Information Record (PIR) and before the corresponding Part Result Record (PRR).

To facilitate conversion from STDF V3, if the first PTR for a test contains default information only (no test results), it may appear anywhere after the MIR (and RDR and SDR, if present), and before the first corresponding PTR, but need not appear between a PIR and PRR.

Sample:

```
PTR:23|2|1|997.3|F|AOH|Check 2nd layer|||
   A|-1.7|45.2| %9.4f|%7.2f|%7.2f|-1.75|45.25|3|3|4
```

Multiple-Result Parametric Record (MPR)

Function: Contains the results of a single execution of a parametric test in the test program where that test returns multiple values. The first occurrence of an MPR also establishes the default values for all semi-static information about the test, such as limits, units and scaling. The MPR is related to the Test Synopsis Record (TSR) by the test number.

Table 2-21. Multiple-Result Parametric Record (MPR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`MPR:`	Yes
Test Number:	`TEST_NUM`	Test number. Test numbers must be unique. The test number does not implicitly increment for successive values in the result array.	Yes
Head Number:	`HEAD_NUM`	Test head number. If there is only one test head on the tester, use 1 in this field.	Yes
Site Number:	`SITE_NUM`	Test site number. If the tester does not support parallel testing, use 1 in this field.	Yes
States Array:	`RTN_STAT`	This is an array of the returned states. It is stored as hexadecimal digits stored in ASCII. Values in this array are optionally separated by commas. The table of valid returned states (expressed as hexadecimal digits) is: 0 = 0 or low 1 = 1 or high 2 = midband 3 = glitch 4 = undetermined 5 = failed low 6 = failed high 7 = failed midband 8 = failed with a glitch 9 = open A = short	No

Table 2-21. Multiple-Result Parametric Record (MPR) fields

ATDF Field	STDF Field	Description	Req?
Results Array:	RTN_RSLT	This is an array of the parametric test results. It is stored as floating point numbers, with each value in the array separated by a comma. If the Scaling Flag: in the FAR is set to S (scaled data), then the values in this array must be scaled according to the STDF scaling rules. (See note below.) If the FAR Scaling Flag: is set to U (unscaled data), then the value need not be scaled. In either case, the units field must be set appropriately for the type of data in this field.	No
Pass/Fail Flag:	TEST_FLG bits 6 & 7 PARM_FLG bit 5	Flag indicates the result of the test. If the pass/fail flag is empty, indicates that the test completed without a pass/fail indication from the test program. Legal values are: A = Test passed alternate limits F = Test failed P = Test passed standard limits empty = Test completed without pass/fail indication	Yes
Alarm Flags:	TEST_FLG bits 0, 2, 3, 4, & 5 PARM_FLG bits 0, 1, 2, 3, & 4	This field may contain zero or more flags. Each flag indicates an alarm or error condition that occurred during the test execution. Legal values are: A = Alarm detected D = Drift error H = Measured value higher than test limit L = Measured value lower than low limit N = Test was not executed O = Oscillation detected S = Scale error T = Time-out occurred U = Test results are unreliable X = Test aborted	No
Test Text:	TEST_TXT	User-defined text string providing information about the test execution.	No
Alarm ID:	ALARM_ID	Name or ID of the alarm or alarms that were triggered. This field should only be used if the alarm flag is also set.	No

Table 2-21. Multiple-Result Parametric Record (MPR) fields

ATDF Field	STDF Field	Description	Req?
Limit Compare:	PARM_FLG bits 6 & 7	By default, limits are compared such that the test fails when the low limit is greater than the test result or the high limit is less than the test result. If that is the case, this field should be empty. Otherwise, the field should contain one or two letters indicating: L = Low limit comparison was >= H = High limit comparison was <=	No
Test Units:	UNITS	Test units. This field will be truncated to seven characters on conversion to STDF. This field must represent whole units and a units prefix is not allowed.	No
Low Limit:	LO_LIMIT	Low test limit value. The value in this field should be scaled to whole units according to the value in the Test Units: field.	No
High Limit:	HI_LIMIT	High test limit value. The value in this field should be scaled to whole units according to the value in the Test Units: field.	No
Starting Value:	START_IN	Starting input value or condition. This is a floating point number.	No
Increment:	INCR_IN	Increment of input value. This is a floating point number and is always unscaled.	No
Input Units:	UNITS_IN	Units of the input condition.	No
Index Array:	RTN_INDX	This is an array of the PMR indexes as defined in the PMR. Entries in the array are separated by commas.	No
Result Format:	C_RESFMT	ANSI C string used for formatting the test results when printing.	No
Lo Limit Fmt:	C_LLMFMT	ANSI C string used for formatting the low limit when printing.	No
Hi Limit Fmt:	C_HLMFMT	ANSI C string used for formatting the high limit when printing.	No
Lo Spec. Limit:	LO_SPEC	Low specification limit value. Useful in calculating process capability.	No
Hi Spec. Limit:	HI_SPEC	High specification limit value. Useful in calculating process capability.	No

Table 2-21. Multiple-Result Parametric Record (MPR) fields

ATDF Field	STDF Field	Description	Req?
Result scale:	RES_SCAL	Test Results scaling exponent. If this ATDF file contains unscaled data, this field is not used.	No
Lo Limit Scale:	LLM_SCAL	Low limit scaling exponent. If this ATDF file contains unscaled data, this field is not used.	No
Hi Limit Scale:	HLM_SCAL	High limit scaling exponent. If this ATDF file contains unscaled data, this field is not used.	No

Note on Scaling

The converter performs automatic scaling on test results in the MPR if the Scaling Flag: in the FAR is set to U (unscaled data). The scaling process is irreversible and will occur whether the output file is in STDF or ATDF format. The prefix will be stripped from the Test Units: field; the test result, the high and low test limits, and the high and low spec limits will all be scaled; and the three scaling fields will be generated from the Test Units: field. The converter never outputs unscaled data. For more information on the use of Results and Limits Scale, Results and Limits Formats, and Units fields, please refer to the section called “Storing and Displaying Test Data” in the PTR portion of the STDF specification.

Note on Default Data

All MPR data starting with the Test Units: field has a special function in the ATDF file. The first MPR for each test will have these fields filled in. The values in these fields will be the default values for each subsequent MPR with the same test number. If the field is filled in for subsequent MPRs, that value will override the default. Otherwise the default will be used. For character strings, it is possible to override the default with a null value by setting the string to a single space.

Unless the default has been overridden, omit the default data fields to save space in the ATDF file.

Frequency: One per multiple-result parametric test execution.

Location: Anywhere in the data stream after the corresponding Part Information Record (PIR) and before the corresponding Part Result Record (PRR).

Sample:

```
MPR:143|2|4||001.3,0009.6,001.5|F|D|||LH|mA|001.0
   |002.0| 4.5|.1|V|3,4,5|%6.1f|%6.1f|%6.1f|0009.75
   |002.25
```

Functional Test Record (FTR)

Function: Contains the results of a single execution of a functional test in the test program. The first occurrence of this record also establishes the default values for all semi-static information about the test. The FTR is related to the TSR by test number.

Table 2-22. Functional Test Record (FTR) fields

ATDF Field	STDF Field	Description	Req?
Header:		FTR:	Yes
Test Number:	TEST_NUM	Test number. Test numbers must be unique.	Yes
Head Number:	HEAD_NUM	Test head number. If there is only one test head on the tester, use 1 in this field.	Yes
Site Number:	SITE_NUM	Test site number. If the tester does not support parallel testing, use 1 in this field.	Yes
Pass/Fail Flag:	TEST_FLG bits 6 & 7	Flag indicates the result of the test. If the pass/fail flag is empty, indicates that the test completed without a pass/fail indication from the test program. Legal values are: F = Test failed P = Test passed	Yes
Alarm Flags:	TEST_FLAG bits 0, 2, 3, 4, & 5	This field may contain zero or more flags. Each flag indicates an alarm or error condition that occurred during the test execution. Legal values are: A = Alarm detected N = Test was not executed T = Time-out occurred U = Test result is unreliable X = Test aborted	No
Vector Name:	VECT_NAM	Vector module pattern name.	No
Timing Set:	TIME_SET	Timing set name.	No
Cycle Count:	CYCL_CNT	Cycle count of vector in decimal.	No
Relative Addr.:	REL_VADR	Relative vector address in hexadecimal.	No
Repeat Count:	REPT_CNT	Repeat count of vector in decimal.	No

Table 2-22. Functional Test Record (FTR) fields

ATDF Field	STDF Field	Description	Req?
Failing Bits:	NUM_FAIL	Number of pins with 1 or more failures.	No
X Fail Addr.:	XFAIL_AD	X component of the logical device address produced by the memory pattern generator, before going through conversion to a physical memory address. This logical address can be different from the physical address presented to the DUT pins.	No
Y Fail Addr.:	YFAIL_AD	Y component of the logical device address produced by the memory pattern generator, before going through conversion to a physical memory address. This logical address can be different from the physical address presented to the DUT pins.	No
Vector Offset:	VECT_OFF	Offset from the vector of interest. This is the integer offset of this vector (in sequence of execution) from the vector of interest (usually the failing vector). For example, if this FTR contains data for the vector before the vector of interest, this field would contain a -1. If this vector contains data for the third vector after the vector of interest, this field would contain a 3. If this FTR is the vector of interest, Vector Offset will contain a 0. It is therefore possible to record an entire sequence of vectors around a failing vector for use with an offline debugger or analysis program.	No
Return Indexes:	RTN_INDX	Array of returned state indexes. For each pin that is read, this array will contain the PMR index number for that pin. The number of values in this array must match the number of values in the Return States: array.	No

Table 2-22. Functional Test Record (FTR) fields

ATDF Field	STDF Field	Description	Req?
Return States:	RTN_STAT	Array of returned states. This array will contain the returned state for each pin listed in the Returned Indexes array above. The table of valid returned state values (expressed as hexadecimal digits) is: 0 = 0 or low 1 = 1 or high 2 = midband 3 = glitch 4 = undetermined 5 = failed low 6 = failed high 7 = failed midband 8 = failed with a glitch 9 = open A = short The characters generated by analysis programs to represent these states are tester-dependent and are specified in the PLR.	No
Prog. Indexes:	PGM_INDX	Array of programmed state indexes. For each pin, this array will contain the PMR index number for that pin. The number of values in this array must match the number of values in the Prog. States: array.	No

Table 2-22. Functional Test Record (FTR) fields

ATDF Field	STDF Field	Description	Req?
Prog. States:	PGM_STAT	Array of programmed states. This array will contain the programmed state for each pin listed in the Prog. Indexes: array above. The table of valid program state values (expressed as hexadecimal digits) is listed below. Note that there are three defined program modes: Normal, Dual Drive (two drive bits per cycle), and SCIO (same cycle I/O). Normal Mode Program States: 0 = Drive Low 1 = Drive High 2 = Expect Low 3 = Expect High 4 = Expect Midband 5 = Expect Valid (not midband) 6 = Don't drive or compare 7 = Keep window open from prior cycle (used to 'stretch' a comparison across cycles) Dual Drive Mode Program States: 0 = Low at D2, Low at D1 times 1 = Low at D2, High at D1 times 2 = Hi at D2, Low at D1 times 3 = Hi at D2, Hi at D1 times 4 = Compare Low 5 = Compare High 6 = Compare Midband 7 = Don't Compare SCIO Mode Program States: 0 = Drive Low, Compare Low 1 = Drive Low, Compare High 2 = Drive Low, Compare Midband 3 = Drive Low, Don't Compare 4 = Drive High, Compare Low 5 = Drive High, Compare High 6 = Drive High, Compare Midband 7 = Drive High, Don't Compare The characters generated by analysis programs to represent these states are tester-dependent and are specified in the PLR.	No

Table 2-22. Functional Test Record (FTR) fields

ATDF Field	STDF Field	Description	Req?
Failing Pins:	FAIL_PIN	Array of PMR index numbers, one for each failing pin. Index numbers are separated by commas.	No
Vector Op Code:	OP_CODE	The OP code for this vector.	No
Test Text:	TEST_TXT	User-defined text string providing information about the test execution.	No
Alarm ID:	ALARM_ID	If the alarm detected flag in the Alarm Flags field is set, this field can optionally contain the name or ID of the alarm or alarms that were triggered. The names of these alarms are tester-dependent.	No
Programed Text:	PROG_TXT	Additional information about the programmed state of this vector.	No
Result Text:	RSLT_TXT	Additional information about the results of this vector.	No
Generator Num:	PATG_NUM	Pattern generator number.	No
Comparators :	SPIN_MAP	List of enabled comparators. This field is an array of the PMR index numbers of the enabled comparators. Each PMR index number is separated from the next by a comma.	No

Note on Default Data

All FTR data starting with the Generator Num: field has a special function in the ATDF file. The first FTR for each test will have these fields filled in. The values in these fields will be the default values for each subsequent FTR with the same test number. If the field is filled in for subsequent FTRs, that value will override the default. Otherwise the default will be used.

Unless the default has been overridden, omit the default data fields to save space in the ATDF file.

Frequency: One per functional test execution.

Location: Anywhere in the data stream between the corresponding PIR and PRR of the device for which the test was executed.

Sample:

```
FTR:27|2|1|P||CHECKERBOARD|A1|5|16|2|3|6|3|0|
    10,2,8,12|0,1,1,4|4,5,6,7|0,0,0,0|8|DRV
    |Check Driver||||2|2,3,4,6
```

Begin Program Section Record (BPS)

Function: Marks the beginning of a new program section (or sequencer) in the test program.

Table 2-23. Begin Program Section Record (BPS) fields

ATDF Field	STDF Field	Description	Req ?
Header:		`BPS:`	Yes
Sequencer Name:	`SEQ_NAME`	Program section (or Sequencer) name.	No

Frequency: Optional on each entry into the program segment.

Location: Anywhere in the data stream between the PIR and the PRR.

Sample: `BPS:DC_TESTS`

End Program Section Record (EPS)

Function: Marks the end of the current program section (or sequencer) in the test program.

Table 2-24. End Program Section Record (EPS) fields

ATDF Field	STDF Field	Description	Req?
Header:		`EPS:`	Yes

Frequency: Optional on each exit from the program segment.

Location: Following the corresponding BPS and before the PRR in the data stream.

Sample:

```
EPS:
```

Generic Data Record (GDR)

Function: Contains information that does not conform to any other record type defined by the ATDF specification. Such records are intended to be written under the control of test programs executing on the tester. This data may be used for any purpose that the user desires.

Table 2-25. Generic Data Record (GDR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`GDR:`	Yes
Generic Data:	`GEN_DATA`	The first character of the field is the format character and will indicate how the ASCII text to follow should be treated. This field may be repeated as many times as desired. Legal values for the first character are: U = U*1 - One byte unsigned integer. M = U*2 - Two byte unsigned integer. B = U*4 - Four byte unsigned integer. I = I*1 - One byte signed integer. S = I*2 - Two byte signed integer. L = I*4 - Four byte signed integer. F = R*4 - Four byte floating point number. D = R*8 - Eight byte floating point number. T = C*n - Variable length ASCII string. X = B*n - Variable length binary data (in hexadecimal). Max length is 255 bytes. Y = D*n - Variable length binary data (in hexadecimal). Max length is 65535 bits. N = N*1 - Unsigned nibble.	No

Note on FLD_CNT

The FLD_CNT field from the STDF is not necessary in the ATDF because the number of fields is implicit in the format of the record.

Note on Pad Bytes

No pad byte is defined because byte alignment is not a problem in the ATDF format. Pad bytes are inserted as necessary on conversion to STDF.

Frequency:	An ATDF file may contain any number of GDRs.
Location:	Anywhere in the data stream following the MIR (and RDR and SDRs, if present) and before the MRR.
Sample:	`GDR:TThis is text|L-435|U255|F645.7110|XFFE0014C`

Datalog Text Record (DTR)

Function: Contains text information that is to be included in the datalog printout. DTRs may be written under the control of a test program: for example, to highlight unexpected test results. They may also be generated by the tester executive software: for example, to indicate that the datalog sampling rate has changed. DTRs are placed as comments in the datalog listing.

Table 2-26. Datalog Text Record (DTR) fields

ATDF Field	STDF Field	Description	Req?
Header:		`DTR:`	Yes
Text Data:	`TEST_DAT`	Any ASCII text string.	No

Frequency: An ATDF file may contain any number of DTRs.

Location: Anywhere in the data stream following the MIR (and RDR and SDRs, if present) and before the MRR.

Sample:
```
DTR:Datalog sampling rate is now 1 in 10
```

3 TABLES

The tables in this section provide the following information, which may be useful for an ATDF file:

Table of ASCII Values

The following is a table of standard ASCII values that are recognized and processed by the ATDF to STDF converter. Other values may be used, but may have varying results on different operating systems.

Table 3-1. ASCII values

HEX	ASCII	HEX	ASCII	HEX	ASCII
0A	LF	3F	?	5F	_
0D	CR	40	@	60	‘
20	space	41	A	61	a
21	!	42	B	62	b
22	“	43	C	63	c
23	#	44	D	64	d
24	$	45	E	65	e
25	%	46	F	66	f
26	&	47	G	67	g
27	‘	48	H	68	h
28	(	49	I	69	i
29	)	4A	J	6A	j
2A	*	4B	K	6B	k
2B	+	4C	Ly	6C	l
2C	,	4D	M	6D	m
2D	-	4E	N	6E	n
2E	.	4F	O	6F	o
2F	/	50	P	70	p
30	0	51	Q	71	q
31	1	52	R	72	r
32	2	53	S	73	s
33	3	54	T	74	t
34	4	55	U	75	u

Table 3-1. ASCII values

HEX	ASCII	HEX	ASCII	HEX	ASCII
35	5	56	V	76	v
36	6	57	W	77	w
37	7	58	X	78	x
38	8	59	Y	79	y
39	9	5A	Z	7A	z
3A	:	5B	[	7B	{
3B	;	5C	\	7C	|
3C	<	5D	]	7D	}
3D	=	5E	^	7E	~
3E	>				

Table of Valid Units Prefixes

The following is a table of Units prefixes that are recognized by the ATDF to STDF converter. These prefixes may used with the units field in the PTR or MPR if the ATDF file contains unscaled test results. Note that if a units prefix is used, you should leave the corresponding scaling fields empty.

Table 3-2. Valid Units Prefixes

Units Prefix	Meaning	Magnitude	SCAL value
f	femto	10**-15	15
p	pico	10**-12	12
n	nano	10**-9	9
u	micro	10**-6	6
m	milli	10**-3	3
%	percent	10**-2	2
no prefix		10**0	0
K	Kilo	10**3	-3
M	Mega	10**6	-6
G	Giga	10**9	-9
T	Tera	10**12	-12

INDEX