INTERNATIONAL STANDARD

ISO 16284

Second edition
2006-03-01

Ophthalmic optics — Information interchange for ophthalmic optical equipment

Optique ophtalmique — Échange d'informations pour l'équipement d'optique ophtalmique

Reference number
ISO 16284:2006(E)

PDF disclaimer

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

Adobe is a trademark of Adobe Systems Incorporated.

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

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

Published in Switzerland

Contents

Page

Foreword

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

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

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

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

ISO 16284 was prepared by Technical Committee ISO/TC 172, *Optics and photonics*, Subcommittee SC 7, *Ophthalmic optics and instruments* and by Technical Committee CEN/TC 170, *Ophthalmic optics* in collaboration.

This second edition cancels and replaces the first edition (ISO 16284:2001), which has been technically revised. Since the publication of the first edition in the year 2001, there have been a number of industry developments. Specifically, surface coater, front surface generator, lens measuring, inspection and lap feeder devices have all been developed. In order to communicate with these devices and to support new features on existing device types, the maintenance committee has proposed a number of new labels and device types. This revised International Standard also proposes a way of dealing with file-based data transfers between devices and hosts. In addition, a number of clarifications has been made to further explain certain requirements of the standard and deprecating several requirements because they have proved difficult to manage in practice.

Introduction

This International Standard is the result of a desire shared by manufacturers of optical laboratory equipment and producers of software used in optical laboratories to simplify the interconnection of their products.

The International Standard defined herein provides:

— a method by which machines and computer systems conduct their exchanges of data;

— a method by which computer systems can initialize such parameters on machines as the manufacturers thereof allow;

— a method by which machines can initialize computer systems with information that the systems can use for various purposes;

— a method by which a machine can inform a computer system as to what information it wants to receive, thus allowing machines to define new interfaces dynamically;

— a standard set of records and device types that are used to communicate agreed upon sets of information.

The last feature listed above requires that this International Standard be amended on a regular basis, as the need for new data elements is inevitable.

Ophthalmic optics — Information interchange for ophthalmic optical equipment

1 Scope

This International Standard establishes a method by which machines and computer software systems used in the fabrication of ophthalmic lenses can exchange information.

2 Normative reference

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO 13666:1998, *Ophthalmic optics — Spectacle lenses — Vocabulary*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO 13666 and the following apply.

3.1 General terms

3.1.1
device
machine or instrument used in the fabrication of ophthalmic lenses that communicates with a computer system to send or receive job information

3.1.2
host
computer system providing information to or receiving information from a device

3.1.3
job
order for prescription ophthalmic lenses or spectacles

3.1.4
download
communication session in which the host system transmits data to the device

3.1.5
upload
communication session in which the device transmits data to the host

3.2 Special characters

3.2.1
code separator
reserved character used to delimit codes in a device record

3.2.2
CRC position character
reserved character marking the location of the end of the data records and the start of the optional CRC record within a packet

3.2.3
end character
reserved character marking the end of a packet

3.2.4
field separator
reserved character delimiting the fields in a record

3.2.5
label separator
reserved character separating the record label from the field(s) within a record

3.2.6
mandatory record flag
reserved character marking certain records as mandatory

3.2.7
start character
reserved character marking the beginning of a packet

3.2.8
record separator
reserved character which delimits records

3.2.9
unknown data indicator
reserved character indicating that data required for a particular field is unknown to the host

3.2.10
ACK character
reserved character indicating successful transmission of a packet

3.2.11
NAK character
reserved character indicating failed transmission of a packet

3.2.12
control character
character having an ASCII value of less than 32

3.3 Data types

3.3.1
limited data
text data limited to a maximum length

3.3.2
literal data
text data limited to a maximum length and specified in this International Standard

3.3.3
numeric data
floating-point and integer numbers

3.3.4
text data
strings of characters that have no pre-defined meaning

3.3.5
integer data
data represented in whole number form

3.3.6
binary data
data presented in a form usable by computer software with little or no translation

NOTE It requires special handling to avoid introduction of control characters.

3.4 Messages

3.4.1
message
structured stream of data transmitted from a host to a device or from a device to a host

3.4.2
confirmation message
message sent by the receiver of a packet and comprised of a single character indicating that the transmission was successful

3.4.3
positive acknowledgement
single character message indicating successful reception of a sender's message

3.4.4
negative acknowledgement
single character message indicating unsuccessful reception of a sender's message

3.4.5
packet
structured message consisting of a start character and a series of records and terminated by an end character

3.4.5.1
data packet
packet sent from a device to a host or a host to a device, and containing requested information

3.4.5.2
request packet
packet sent from a device to a host to initiate a session

3.4.5.3
response packet
packet containing status information

3.5 Records

3.5.1
record
structured stream of characters including a record label, a label separator, zero or more data fields separated by field separators and a terminating record separator

3.5.2
data field
single data element within a record

3.5.3
record label
means of identifying data contained in a record, limited in length to 8 characters and not including spaces or reserved characters defined in this International Standard

NOTE A list of device record labels is in Annex A.

3.5.4
ASCII record
record comprised of ASCII characters and conforming to the structures defined herein

3.5.5
binary record
record comprised of bytes encoded using the binary number system

3.5.6
chiral record
record with two fields, one for a data element for a right lens or eye, and one for a left, arranged in the order right then left

3.5.7
CRC record
record at the end of any packet containing a CCITT[1] CRC-16 cyclical redundancy check value calculated on the characters transmitted

3.5.8
device record
record containing job specific data elements conveyed between devices and hosts

3.5.9
interface record
record supporting the operation of the host-device interface and not containing job-specific data

3.6 Sessions

3.6.1
session
sequence of messages passed between a device and a host that serves to exchange information related to a single order or task

3.6.2
initialization session
specialized session allowing devices to provide hosts with information that would otherwise be included with each request, such as machine model, software version and operator ID

1) Comité Consultatif International Téléphonique et Télégraphique

3.6.2.1
auto-format initialization
initialization session allowing devices to define sets of device records to be requested from hosts

3.6.2.2
preset initialization
initialization session allowing devices to transmit sets of identifying data to hosts

3.6.3
download session
session in which information is passed from a host to a device

3.6.4
upload session
session in which information is passed from a device to a host

3.6.5
INFO session
upload request packet containing job status information used to indicate the completion of a job by a device

3.6.6
MNT session
upload request packet containing vendor specific device information

3.7 Timeout

3.7.1
timeout
numeric value representing that period of time that a host or device shall wait for the arrival of data, after which it assumes that such data will not be forthcoming

3.7.1.1
confirmation timeout
timeout which applies to the reception of the confirmation message

3.7.1.2
intercharacter timeout
timeout which applies to the interval between successive characters in a stream of data

3.7.1.3
packet timeout
timeout which applies to the reception of a packet

4 Overview

The strategy used in this International Standard for the exchange of data between devices and hosts can be expressed as follows.

A machine used in the fabrication of ophthalmic lenses (a device) sends a request to a computer system (a host), indicating a need to do one of the following:

— initialize information to identify the device, software versions, model numbers, etc.;

— upload to the host, information for it to store and/or use in the processing of ophthalmic prescription orders;

— download from the host, information required by the device for it to perform its tasks.

Communication can be initialized in two ways. The device may begin an initialization session or the host can force the device to do so by refusing to accept a normal request and asking for initialization via a special error response. For upload requests, the host acknowledges the request and the device sends its data, the receipt of which the host acknowledges. For download requests, the host responds to the request with the data requested.

The variable-length packets of data that comprise this exchange consist of a series of records, each of which contains data and a label identifying the data. This International Standard defines a set of labels and characterizes the data associated with each. This set of labels shall be expanded as needed in the future.

An exchange of packets related to a single job is called a session. The structure of these sessions and the packets of records of which they are comprised is the subject of this International Standard.

Although this International Standard was conceived as being implemented on point-to-point RS-232 serial links, it could be implemented on other hardware platforms. As this is done, specifications shall be incorporated into this International Standard so as to maximize interconnectability amongst diverse hosts and devices.

In the examples given in this International Standard, in the interests of legibility, the RECORD SEPARATORS may be omitted, the START CHARACTER may be placed on a separate line and CRC RECORDS may be excluded. Remarks have been included as REM records. Comments are enclosed in square brackets (**[…]**) and are not part of the data stream. Ellipses ("…") are used to indicate more data of the same type as precedes and follows the ellipses. SPACES have been inserted around record and field separators for readability; in practice these should not be included in packets as this needlessly decreases the efficiency of expression. In the descriptions of Clause 5, REQUEST, RESPONSE and DATA refer to packets. Since encoded data are always expressed using the point as the decimal sign, to avoid confusion the decimal sign is used throughout this document although the ISO decimal sign is the comma. Furthermore, the space representing the thousandths separator is not given in the encoded data.

5 Requirements

5.1 Records

5.1.1 Interface records

This International Standard defines a set of interface records. These records contain information which the host and device use to communicate. They do not contain job-specific data. These records are enumerated in A.2.

5.1.2 Device records

This International Standard defines a set of device records which identify the data elements that might be required by any of the devices that might be required for the fabrication of a job. These records are enumerated in A.1.

5.1.3 Preset device types

This International Standard further identifies subsets of device records that are deemed to be appropriate for specific types of devices. These preset records are enumerated in A.3.

5.1.4 Records with unknown values

If the host is requested to send any record for which it has no information or partial information, it shall send the record with a question mark "?" in all the unknown data fields in order to indicate that the information is not available. Such records shall be properly formatted according to the rules for chiral records.

5.1.5 Ignored records

Whenever a host or device receives a record with a label it does not recognize, it shall ignore the record.

5.1.6 Experimental records

When a machine vendor wishes to test new records prior to submitting them for inclusion in this International Standard, such records should use labels that begin with an underscore character (ASCII "_", decimal 95). Record labels are limited in length to 8 characters and may not include spaces or reserved characters defined in this International Standard.

5.1.7 Reserved characters

5.1.7.1 Control characters and the additional characters specified may not appear in transmitted data streams except as specified. The set of reserved characters is specified in Table 1.

5.1.7.2 Reserved characters shall appear in ASCII records only to provide the functionality that they are assigned, as in the case of record and field separators. Reserved characters which conform to the definition of text data may also appear in text fields.

5.1.7.3 When a reserved character with a decimal value less than 32 appears in a binary record, it shall be "escaped" in the following manner. In place of such a character, two characters shall be sent. The first character shall be an ESC character followed by the original character with its high bit set, i.e. the character is OR'd with decimal 128, hex 0×80. The receiver, on receipt of an ESC character, shall discard the ESC character and clear the high bit of the following character. The CRC value, if present, shall be determined after such reserved characters are escaped, so that a receiver need not process packets prior to validating a received packet's CRC.

NOTE In other words, the transmitter encodes control characters before calculating the CRC, and the receiver calculates the CRC before decoding them.

EXAMPLE A stream of bytes (a short tracing record in absolute binary form) before and after having been "escaped" as described above.

```
Before:
   R=175  9  23  10  45  10 223  9  90  9 205  8  89  8 252   7 183   7 143   7
     130  7 147   7 197   7  24  8 136  8  18  9 167  9  39  10  85  10  19  10
     213  9 146   9  75   9  14  9 199  8 120  8  38  8 222   7 166   7 131   7
     117  7 122   7 149   7 191  7 241  7  41  8  92  8 152   8 229   8  67   9 <CR/LF>

After:
   R=175  9  23 27 138 45 27 138 223 9 90 9 205 8 89  8 252   7 183   7 143   7
     130  7 147  7  197   7  24  8 136  8  18  9 167  9  39 27 138 85 27 138 27 147 27 138
     213  9 146  9   75   9  14  9 199  8 120  8  38  8 222   7 166   7 131   7
     117  7 122  7  149   7  191 7 241  7  41  8  92  8 152   8 229   8  67   9 <CR/LF>
```

5.1.7.4 Limited data are a string of ASCII characters in the range 32 to 127 decimals excluding 59 (semi-colon). The length of this string is limited to 12 characters.

5.1.7.5 Text data are a string of ASCII characters in the range 32 to 127 decimals excluding 59 (semi-colon) and having no predefined meaning. The length is limited to 80 characters.

5.1.7.6 Literal data is a string of ASCII characters in the range 32 to 127 whose meaning is implied by the record type and specified in this International Standard. Length is limited to 12 characters unless otherwise noted in the record definition. Literal data shall not contain reserved characters defined by the interface (see Table 1). Literal data is case sensitive.

5.1.8 Record length

Non-binary records should not exceed 80 characters in length. Binary data may be longer than 80 characters.

Table 1 — Reserved characters

Character	Hexadecimal value	Decimal value	Control key	Use
FS	0×1C	28	^\	Start of message
GS	0×1D	29	^]	End of message
DC1	0×11	17	^Q	Reserved (XOFF)
DC3	0×13	19	^S	Reserved (XON)
ACK	0×06	06	^F	Positive acknowledgement
NAK	0×15	21	^U	Negative acknowledgement
ESC	0×1B	27	^[	Escape
RS	0×1E	30	^^	CRC separator
SUB	0×1A	26	^Z	DOS end-of-file marker
CR	0×0D	13	^M	Record separator
LF	0×0A	10	^J	Record separator
;	0×3B	59	;	Field separator
=	0×3D	61	=	Label separator
,	0×2C	44	,	Code separator
*	0×2A	42	*	Mandatory record flag
?	0×3F	63	?	Unknown data indicator

5.2 Reference point records

5.2.1 Records are defined to indicate the horizontal and vertical distances between two reference points or to indicate an action that a machine should take relative to a reference point. The following naming scheme will clarify all such reference records included in this International Standard and can easily be extended for future ones.

5.2.2 The first two letters of the record label describe the first reference point (see Table 2), the second two letters of the record label describe the second reference point and the last two letters indicate “IN” (horizontal) or “UP” (vertical) directions (see Table 3). The values indicate the position of the second reference point with respect to the first reference point.

5.2.3 A positive IN value indicates that the second reference point is towards the nasal relative to the first.

5.2.4 A negative IN value indicates that the second reference point is towards the temporal relative to the first.

5.2.5 A positive UP value indicates that the second reference point is above the first.

5.2.6 A negative UP value indicates that the second reference point is below the first.

Table 2 — Reference point identifiers

Identifier	Reference point
BC	Blank centre
FB	Finish block
FC	Frame centre
OC	Optical centre/prism reference point
SB	Surface block
SG	Segment/progressive fitting cross (layout reference point)

Table 3 — Reference point records

Label	Meaning
FBFCIN, FBFCUP	Finish block to frame centre (see Table A.1 for use)
FBSGIN, FBSGUP	Finish block to segment
FBOCIN, FBOCUP	Finish block to prism reference point (O.C.)
SBBCIN, SBBCUP	Surface block to blank centre
BCSGIN, BCSGUP	Blank centre to segment
BCOCIN, BCOCUP	Blank centre to prism reference point (O.C.)
SBSGIN, SBSGUP	Surface block to layout reference point (segment)
SBOCIN, SBOCUP	Surface block to prism reference point (see 5.3.11 for use)
SBFCIN, SBFCUP	Surface block to frame centre
SGOCIN/SGOCUP	Segment to O.C.
FCSGIN/FCSGUP	Frame centre to segment (similar to segment height or drop)
FCOCIN/FCOCUP	Frame centre to O.C. (similar to O.C. height or drop)

5.3 Generator records

5.3.1 The surface generator interface includes a number of records used to indicate adjustments that should be applied to the generator machine settings. Because this International Standard provides for a complete data set (“preset packet”) to be sent to an “unknown” generator, it is necessary to clarify some of the relationships amongst these records, especially as relates to the “compensation” fields.

The position of a lens in a generator can be determined by the RNGH, RNGD, SAGRD and SAGBD fields (ring height, ring diameter, lens sag at ring diameter and lens sag at blank diameter, respectively). Some generators, especially those with exclusively mechanical components, may presume certain values for some of the above records and may be unable to effect the adjustments required by the mismatch in assumptions. The following compensation fields provide the data required to make these adjustments.

5.3.2 BLKCMP represents the change that is required to be made to the generator thickness setting that arises from a mismatch between the curvature of a block that has a curved contact surface with the lens and the curvature of the lens blocked thereon. The BLKB field contains the curvature of that surface of the block that contacts the lens; the BLKD field contains the diameter of the block.

5.3.3 When the blocks used for a job do not have a curved contact surface, the BLKB field is not necessary. If it is sent in such a case, its value should be equivalent to IFRNT.

5.3.4 RNGCMP represents the change that is required to be made to the generator thickness setting that arises from a mismatch between the blocking ring height and/or diameter, known to the host, and that which is presumed by the generator.

5.3.5 FINCMP indicates the amount of thickness that should be added to the generator setting to allow for material removed during fining (smoothing) and polishing.

5.3.6 THKCMP shall be used for such compensations to thickness as may be required, which are not otherwise handled by the records defined in this International Standard (the thickness compensation fields apply equally to GTHK and OTHK).

5.3.7 EECMP indicates a dioptre amount that shall be added to the GCROS and GCROSX values when elliptical error compensation is required.

5.3.8 A host can maintain complete control over the settings on a generator by including all of the compensation values that it believes to be required by a particular machine in the basic setting records and sending zero values in the compensation records; e.g. sum GCROS with EECMP, send the result as GCROS and send EECMP with values equal to zero.

5.3.9 In compensation records, zero values indicate that the host does not want the machine to apply any compensation that it may be capable of being set to do.

5.3.10 The ABSENCE of compensation fields indicates that the generator should apply such compensations as it may be set to do.

5.3.11 For prism versus decentration, the fields GPRVM and GPRVA represent prism magnitude and direction to be generated. These values shall include all possible contributions to prism, including Rx (prescription) prism, thinning prism, and prism for decentration. The fields SBOCIN and SBOCUP express vector distances to which the grinding centre is to be offset from the surface block centre, laterally and vertically. In order that both be expressed in a single packet, as would be desirable in the case of a “generic” GEN request, a set of prism records is defined, RPRVM and RPRVA, which represents prism to be ground exclusive of the decentration expressed in SBOCIN/SBOCUP. In addition, when decentration is expressed directly (via SBOCIN/SBOCUP), the OTHK value should be present. The following rules describe how these fields should be aggregated.

5.3.11.1 When expressing decentration to be generated as prism, the following fields are used together: GPRVM, GPRVA and GTHK.

5.3.11.2 When expressing decentration to be generated directly, the following fields are used together: RPRVM, RPRVA, SBOCIN, SBOCUP and OTHK.

5.3.11.3 Because the method described in 5.3.11.1, in which decentration is expressed in the form of prism, is the more universal one for expressing decentration to be ground, the set of records described therein should be supported by all generators and hosts. The set of records described in 5.3.11.2, in which decentration is expressed as vectors, should be included in addition to the 5.3.11.1 set. Generators not supporting the vector method can safely ignore the vector fields.

5.3.12 For pad thickness**,** the inclusion of the PADTHK record indicates that LAP curves cut by a generator should be compensated for the thickness of the pad. No compensation should be made to the lens curves expressed in GBASE/GCROS or GBASEX/GCROSX. It is the responsibility of the host to determine such compensations as shall be applied to generator curves, the need for which may result from the use of laps not compensated for pad thickness, or the desire to fine lenses from edge to centre.

5.3.13 For curve signs, concave curves shall be expressed as negative numbers and convex curves as positive numbers. One implication of this is that for generators that cut laps, the lens curves and lap curves shall have opposite signs.

5.4 Tracing records

5.4.1 Expression of trace data

Trace data begins with a TRCFMT record that specifies the format in which the tracing is to be expressed, the number of points to be transmitted, whether the radii are equiangular, the orientation of the tracing and an indication of what was traced.

A device indicates its desire to upload or download trace data by including one or more TRCFMT records in either its initialization packet, or, if no initialization is done, its request packet. All supported formats and numbers of points are listed in the order of the device's preference. It is not necessary to include every combination of side specifiers (B, R, and L); because of the requirement specified in 5.4.13, only the device's preferred mode need be specified. Devices that neither upload nor download trace data do not include any TRCFMT records in their initialization or request packets. The same rules apply to sag data and ZFMT.

When trace data is expected by a device, but unavailable from the host, a special form of the TRCFMT record is sent, composed of a single field with the integer value 0. When trace data is available, but sag data is not, a similarly-formed ZFMT record is sent by the host. When TRCFMT=0 is sent, ZFMT=0 is implied and not required. When two sides of radius data are sent, for which sag data are unavailable, a ZFMT=0 record should follow each set of radius data. In the case that two-side data has been negotiated, and only a single side is sent, TRCFMT=0 is not sent for the missing side.

EXAMPLE 1 Trace data requested by device, but unavailable from host:

```
TRCFMT=0
```

EXAMPLE 2 Trace data requested by device and available from host, Z data requested, but unavailable:

```
TRCFMT=1;400;E;R;F<CR/LF>
R=2479;2583;2605;2527;2394;2253;2137;2044;1975;1935<CR/LF>
R=1922;1939;1989;2072;2184;2322;2471;2599;2645;2579<CR/LF>
<etc.>
R=1922;1939;1989;2072;2184;2322;2471;2599;2645;2579<CR/LF>
ZFMT=0
```

5.4.2 Radius data

These are contained in “R” records. Sag data is contained in “Z” records. Angle data for radius data is contained in “A” records and for sag data in ZA records. For the ASCII format, all of the records follow the 80-character line limit rule, therefore there may (in fact, will likely) be multiple R, A, Z and ZA records for each tracing. For BINARY data, the line limit rule will not apply and there will be only one R, A, Z and ZA record as needed. Radius and sag data is expressed in hundredths of a degree with an implied decimal point.

EXAMPLE Radius data:

a) ASCII format

```
TRCFMT=1;400;E;R;F<CR/LF>
R=2479;2583;2605;2527;2394;2253;2137;2044;1975;1935<CR/LF>
R=1922;1939;1989;2072;2184;2322;2471;2599;2645;2579<CR/LF>
etc…
```

b) Binary formats

```
TRCFMT=2;400;E;R;F<CR/LF>
R=<binary data stream><CR/LF>
```

5.4.3 Uneven angles

When these are specified in the TRCFMT record, angle data is required. It shall appear immediately after its corresponding radius data. Angle data is contained in one or more “A” records, and shall be expressed in the same format as the radius data to which it corresponds. Angle data is expressed hundredths of a degree with an implied decimal point and shall be in the range 0 to 35999.

NOTE Angle data can also follow the sag data if uneven angles are specified for the sag data.

5.4.4 ZFMT

The sag data format record, ZFMT, has exactly the same definition as the TRCFMT record. The same rules apply to angle data for sag data as to angle data for radius data.

5.4.5 “R” records

These shall appear immediately after the TRCFMT record. All of the “R” records for a tracing (i.e. a single side, when two are provided) appear together. Each side has its own TRCFMT record.

5.4.6 “Z” records

These shall appear immediately after the ZFMT record. All of the “Z” records for a tracing (i.e. a single side, when two are provided) appear together. Each side has its own ZFMT record.

5.4.7 “A” or ZA records

These, when present, shall appear immediately after their corresponding set of “R” or “Z” records. ZFMT and “Z” records shall appear immediately after their corresponding TRCFMT and “R” records.

EXAMPLE A two-eye tracing data set (abbreviated) showing the required sequence of records:

```
TRCFMT=1;400;U;R;F<CR/LF>
R=2479;2583;2605;2527;2394;2253;2137;2044;1975;1935<CR/LF>
R=1922;1939;1989;2072;2184;2322;2471;2599;2645;2579<CR/LF>
…
R=1909;1914;1941;1983;2033;2089;2140;2200;2277;2371<CR/LF>
A=0;90;180;270;360;450;540;630;720;810<CR/LF>
A=900;990;1080;1170;1260;1350;1440;1530;1620;1710<CR/LF>
…
A=35100;35190;35280;35370;35460;35550;35640;35730;35820;35910<CR/LF>
ZFMT=1;100;U;R;F<CR/LF>
Z=322;331;342;328;314;308;300;295;288;280<CR/LF>
…
Z=316;318;324;328;333;343;349;352;357;362<CR/LF>
ZA=0;360;720;1080;1440;1800;2160;2520;2880;3240<CR/LF>
…
ZA=32400;32760;33120;33480;33840;34200;34560;34920;35280;35640<CR/LF>
TRCFMT=1;400;U;L;F<CR/LF>
R=2517;2450;2379;2318;2247;2168;2086;2014;1958;1923<CR/LF>
R=1909;1914;1941;1983;2033;2089;2140;2200;2277;2371<CR/LF>
…
R=1922;1939;1989;2072;2184;2322;2471;2599;2645;2579<CR/LF>
A=0;90;180;270;360;450;540;630;720;810<CR/LF>
A=900;990;1080;1170;1260;1350;1440;1530;1620;1710<CR/LF>
…
A=35100;35190;35280;35370;35460;35550;35640;35730;35820;35910<CR/LF>
ZFMT=1;100;U;R;F<CR/LF>
Z=322;331;342;328;314;308;300;295;288;280<CR/LF>
```

```
…
Z=316;318;324;328;333;343;349;352;357;362<CR/LF>
ZA=0;360;720;1080;1440;1800;2160;2520;2880;3240<CR/LF>
…
ZA=32400;32760;33120;33480;33840;34200;34560;34920;35280;35640<CR/LF>
```

5.4.8 Frame tracings

These shall be centred geometrically when presented to the host. The host may decentre the shape in order to produce a certain effect on a device.

5.4.9 Frame sizing

5.4.9.1 BSIZ and CSIZ records indicate that devices shall modify the dimensions of the received trace by the amount specified.

5.4.9.2 Non-zero values shall not be supplied for both BSIZ and CSIZ.

5.4.9.3 HBOX, VBOX and CIRC records shall reflect the dimensions of the shape sent in R records.

5.4.9.4 In the particular case where BSIZ and CSIZ are absent or unknown and a single side tracing and two CIRC values which differ are received, the CIRC for the eye sent should reflect the shape. CSIZ is implied to be the difference between the two circumferences and the shape for the eye not sent shall be modified according to the implied CSIZ.

5.4.10 Tracing data

These shall be expressed so that the first radius is at zero degrees (3 o'clock on standard polar scale) and shall proceed anti-clockwise. When angle data are provided, the starting radius shall be the one at the first available meridian greater than or equal to zero.

5.4.11 Eye orientation

5.4.11.1 Eye orientation, right or left, shall be viewed as a refractionist views a spectacle wearer; right-eye oriented data therefore start at the nasal side while left-eye data start at the temporal.

5.4.11.2 Eye orientation may be established during initialization. If it is not, it is specified in the devices' request packet.

5.4.11.3 When eye R (right) is specified, the device wants to send or receive a single set of trace data with right-eye orientation.

5.4.11.4 When eye L (left) is specified, the device wants to send or receive a single set of trace data with left-eye orientation.

5.4.11.5 When eye B (both) is specified, the device wants to send or receive both right and left sets of tracing data, each in the appropriate orientation.

5.4.12 Eye orientation during tracing transmission

5.4.12.1 When eye R (right) is specified in the TRCFMT or ZFMT records, the tracing will have RIGHT eye orientation.

5.4.12.2 When eye L (left) is specified in the TRCFMT or ZFMT records, the tracing will have LEFT eye orientation.

5.4.12.3 Eye B shall not be specified in data packets, only in request or initialization packets. In data packets in which both sides are included, there shall be two TRCFMT records, one for each side, labelled appropriately.

5.4.13 Trace orientation acceptance

Hosts and devices shall handle the reception of either orientation of tracing data without generating an error condition. Hosts should make an effort to provide data in the quantity and orientation requested by the device when possible.

5.4.14 Sag data

In the expression of sag data, the smallest number in the data set shall be positive and shall represent the point located furthest toward the front of the frame or lens. The distances from this point to other points in the data set shall have positive values in increments of 0.01 mm.

5.5 Tracing formats

5.5.1 Example data

It is strongly recommended that devices be consistent in the representations of data. While it is not expressly forbidden, representing radius data in one orientation and sag data in another, might violate host systems' programmers unwarranted assumptions and fail.

The following small sample tracing, comprised of 40 radii, is used for the examples below. The sample is not formatted in any particular way; it is just a list of the radius values used in the examples given in 5.5.2 to 5.5.6.

24.79, 25.83, 26.05, 25.27, 23.94, 22.53, 21.37, 20.44, 19.75, 19.35,

19.22, 19.39, 19.89, 20.72, 21.84, 23.22, 24.71, 25.99, 26.45, 25.79,

25.17, 24.50, 23.79, 23.18, 22.47, 21.68, 20.86, 20.14, 19.58, 19.23,

19.09, 19.14, 19.41, 19.83, 20.33, 20.89, 21.40, 22.00, 22.77, 23.71

5.5.2 ASCII absolute format

In this format, each radius is presented as a 4-digit decimal number in hundredths of a millimetre with an implied decimal point. Each value is separated by a field separator (semi-colon). Data shall follow the 80-character line limit rule, therefore multiple “R” records are required for a single radius data set.

EXAMPLE A tracing expressed in format #1, ASCII absolute, requires 196 bytes of radius data (including the semi-colons).

```
TRCFMT = 1;40;E;R;F<CR/LF>
R=2479;2583;2605;2527;2394;2253;2137;2044;1975;1935<CR/LF>
R=1922;1939;1989;2072;2184;2322;2471;2599;2645;2579<CR/LF>
R=2517;2450;2379;2318;2247;2168;2086;2014;1958;1923<CR/LF>
R=1909;1914;1941;1983;2033;2089;2140;2200;2277;2371<CR/LF>
```

5.5.3 Binary absolute format

In this format, each radius is expressed as a 16-bit integer with the radius in hundredths of a millimetre. The entire data set is contained within a single radius data record.

EXAMPLE Each binary 8-bit byte in the following example is represented by one or more decimal digits separated from the next byte by a space. Semi-colons are used to separate the individual radii. *The semi-colons and spaces are not part of the actual binary record.*

A tracing expressed in format #2, binary absolute, requires 86 bytes including the escape characters. The first example shows the data prior to the escape process (described in 5.1.7.3) having been applied, the second, afterwards. The escape sequences are shown in **bold**.

```
TRCFMT = 2;40;E;R;F
R=175   9; 23  10; 45  10;223   9; 90   9;205   8; 89   8;
  252   7;183   7;143   7;130   7;147   7;197   7; 24   8;
  136   8; 18   9;167   9; 39  10; 85  10; 19  10;213   9;
  146   9; 75   9; 14   9;199   8;120   8; 38   8;222   7;
  166   7; 131  7;117   7;122   7;149   7;191   7;241   7;
   41   8;  92  8;152   8;229   8; 67   9<CR/LF>

TRCFMT = 2;40;E;R;F
R=175   9;23 27 138;45 27 138;223   9; 90   9;205   8 ; 89   8;
  252   7;183   7;143   7;130   7;147   7;197   7; 24   8;
  136   8; 18   9;167   9; 39 27 138; 85 27 138;27 147 27 138;213  9;
  146   9; 75   9; 14   9;199   8;120   8; 38   8;222   7;
  166   7;131   7 117   7;122   7;149   7;191   7;241   7;
  41    8; 92   8;152   8;229   8; 67   9<CR/LF>
```

5.5.4 Binary differential format

In this format, the starting radius is represented as a two-byte integer, as in format 2. Each radius thereafter is represented in a single signed byte as the difference between the current radius and the previous one. The data value expressed is therefore equal to the previous radius subtracted from the current. If the differential cannot be represented by signed byte, whose value shall be between −127 and +127, a special value (hexadecimal 0×80, − 128 decimal) is used to indicate that the next radius is expressed in absolute (i.e. 16-bit) form. The radius immediately following the 16-bit value is then represented in differential form.

EXAMPLE A tracing expressed in format #3, binary differential, requires 54 bytes including the escape characters. As above, the first example shows the data prior to the escape process (described in 5.1.7.3) having been applied, the second, afterwards. In the first example, the absolute radii flags are shown in **bold**. In the second, the escape sequences are shown in **bold**.

```
TRCFMT = 3;40;E;R;F
R=175   9; 104; 22; -78;-128  90  9;-128 205 8;-116; -93;
  -69; -40; -13;  17;  50;  83; 112;-128 18 9;-128 167 9;
 -128  39  10;  46; -66; -62; -67; -71; -61; -71; -79; -82;
  -72; -56; -35; -14;  5; 27; 42; 50; 56; 51; 60; 77; 94<CR/LF>

TRCFMT = 3;40;E;R;F
R=175   9; 104; 22; -78 ;-128 90  9;-128 205 8;-116; -93;
  -69; -40; -13; 27 145; 50; 83; 112;-128 18 9;-128 167 9;
 -128  39 27 138; 46; -66; -62; -67; -71; -61; -71; -79; -82;
  -72; -56; -35; -14;  5; 27 155; 42; 50; 56; 51; 60; 77; 94<CR/LF>
```

5.5.5 Packed binary format

This is the most efficient, and most complex, method for the expression of shape information. To facilitate implementation of this format, a complete example is included in Annex B.

Three data types are defined (the decimal sign is a point):

— absolute, in which sixteen-bit “words” are used to express values in the range −327.67 mm to +327.67 mm;

— differential, in which eight-bit “bytes” are used to express values in the range −1.26 mm to +1.27 mm;

— incremental, in which four-bits “nibbles” are used to express values in the range −0.07 mm to +0.07 mm.

The first radius in a tracing is always expressed using the absolute data type. Radii subsequent to the first may be expressed as any of the three forms. Special values are reserved for each of the three types, which are inserted into the data stream to indicate that subsequent radii will be expressed in a different data type. These are shown in Table 4.

Table 4 — Packed binary type-shifting flag characters

Data type	Hexadecimal value	Decimal value	Action	Label
Word	0×8000	− 32768	Shift from absolute to differential form	AD
Byte	0×80	− 128	Shift from differential to incremental form	DI
Byte	0×81	− 127	Shift from differential to absolute form	DA
Nibble	0×8	− 8	Shift from incremental to differential form	ID

Trace radii can always be expressed by the absolute data type, in which case the magnitude of the radius is simply encoded in a sixteen-bit word value in units of 0,01 mm, thus requiring sixteen bits per radius. Were this done for an entire record, the format would be indistinguishable from the binary absolute format.

NOTE The following examples use the sequences of radii: 25.40, 25.62, 25.97, 27.20, 28.00, 28.30, 28.35, 28.40, 28.43.

EXAMPLE Radii expressed in absolute form.

Radius	**Encoding**	**Value**	**Type**	**Size**
First radius	(25.40 * 100)	2540	Absolute	word
Second radius	(25.62 * 100)	2562	Absolute	word
Third radius	(25.97 * 100)	2597	Absolute	word
Fourth radius	(27.20 * 100)	2720	Absolute	word
Fifth radius	(28.00 * 100)	2800	Absolute	word
Sixth radius	(28.30 * 100)	2830	Absolute	word
Seventh radius	(28.35 * 100)	2835	Absolute	word
Eighth radius	(28.40 * 100)	2840	Absolute	word
Ninth radius	(28.43 * 100)	2843	Absolute	word

If the difference in the magnitudes of two sequential radii is in the range which can be expressed by the differential data type, the magnitude of the second can be expressed in a byte (eight bits). An AD flag is inserted in the data stream indicating that subsequent radii are expressed in differential form unless and until another type-shifting flag appears in the stream. Each subsequent differential value is added to the decoded radius of its predecessor.

EXAMPLE Radii expressed in differential form.

Radius	**Encoding**	**Value**	**Type**	**Size**
First radius	(25.40 * 100)	2540	Absolute	word
AD Flag		−32768	Absolute	word
Second radius	(25.62-25.40 * 100)	22	Differential	byte
Third radius	(25.97-25.62 * 100)	35	Differential	byte
DA Flag		−127	Differential	byte
Fourth radius	(27.20 * 100)	2720	Absolute	word
AD Flag		-32768	Absolute	word

Fifth radius	(28.00-27.20 * 100)	80	Differential	byte
Sixth radius	(28.30-28.00 * 100)	30	Differential	byte
Seventh radius	(28.35-28.30 * 100)	5	Differential	byte
Eighth radius	(28.40-28.35 * 100)	5	Differential	byte
Ninth radius	(28.43-28.40 * 100)	3	Differential	byte

In the above example, it is necessary to revert to the absolute form to express the fourth radius because the difference between the fourth and third (27.20 − 25.62 or 1.58) exceeds the range which can be expressed using the differential form.

If the difference in the magnitude of the differential values of two sequential radii is in the range which can be expressed by the incremental data type, the second radius can be expressed in a nibble (four bits). A DI flag is inserted in the data stream indicating that subsequent radii are expressed in incremental form unless and until an ID flag appears in the stream. Each subsequent incremental value is added to the current differential value, which is then added to the decoded radius of its predecessor.

EXAMPLE Radii expressed in incremental form.

Radius	**Encoding**	**Value**	**Type**	**Size**
First radius	(25.40 * 100)	2540	Absolute	word
AD Flag		−32768	Absolute	word
Second radius	(25.62-25.40 * 100)	22	Differential	byte
Third radius	(25.97-25.62 * 100)	35	Differential	byte
DA Flag		−127	Differential	byte
Fourth radius	(27.20 * 100)	2720	Absolute	word
AD Flag		−32768	Absolute	word
Fifth radius	(28.00-27.20 * 100)	80	Differential	byte
Sixth radius	(28.30-28.00 * 100)	30	Differential	byte
Seventh radius	(28.35-28.30 * 100)	5	Differential	byte
DI Flag		−128	Differential	byte
Eighth radius	((28.35-28.30)-(28.40-28.35) * 100) 0		Incremental	nibble
Ninth radius	((28.40-28.35)-(28.43-28.40) * 100) −2		Incremental	nibble

Because frame shapes are usually comprised of gentle arcs and most tracings are expressed using a large number of radii (four hundred or more), most can be expressed using the incremental form for the entire packet, except for the starting radius.

EXAMPLE A tracing expressed in format #4, packed binary, requires 59 bytes including the escape characters. As above, the first example shows the data prior to the escape process (described in 5.1.7.3) having been applied, the second, afterwards. In the first example, the switch flags are shown in **bold**. In the second, the escape sequences are shown in **bold**.

```
TRCFMT = 4;40;E;R;F
R=af 09 00 80 68 16 b2 81 5a 09 cd
  08 00 80 8c a3 bb d8 f3 11 32 53
  70 81 12 09 a7 09 27 0a 00 80 2e
```

```
  be 80 4b c8 c3 b9 b1 80 d8 b8 c8
  dd f2 05 1b 2a 32 80 6b 83 c4 d5 e0<CR/LF>

TRCFMT = 4;40;E;R;F
R=af 09 00 80 68 16 b2 81 5a 09 cd
  08 00 80 8c a3 bb d8 f3 1b 91 32 53
  70 81 12 09 a7 09 27 1b 8a 00 80 2e
  be 80 4b c8 c3 b9 b1 80 d8 b8 c8
  dd f2 05 1b 9b 2a 32 80 6b 83 c4 d5 e0 <CR/LF>
```

Refer to the code example in Annex B for further details.

5.5.6 Special considerations for binary formats

5.5.6.1 Sixteen-bit numbers in binary data streams are represented in Intel 80×86 byte order in which the first byte is the low order or least significant byte and the second byte is the high order or most significant byte. Some computer systems (e.g. systems running on Motorola 680xx processors) will have to swap bytes internally in order to get the data to process correctly. An example of this is provided in the source code example in Annex B.

5.5.6.2 Care shall be taken to treat the data as signed or unsigned as called for by the data formats described herein.

5.5.6.3 It is important to observe the rules for encoding reserved characters described in 5.1.7.3.

5.5.6.4 Binary data shall begin immediately after the label separator and end immediately prior to the record separator; unlike for ASCII data, superfluous spaces would be catastrophic.

5.6 Packets

5.6.1 Order of records in packets

In all packets, the REQ or ANS record shall appear first. The JOB record shall immediately follow the REQ or ANS record in non-initialization packets. Trace records shall be in the order TRCFMT, R, [A], [ZFMT], [Z], [ZA] where brackets [ ] indicate optional records. See 5.4.7 for a complete example. The CRC record shall be last following the <RS> character. See 5.5.2 for more information. For other records, order is not important.

5.6.2 CRC record

5.6.2.1 Hosts and devices may include, as the last record in any packet, a record used to validate the contents of the packet, as described in 5.6.2.2 to 5.6.2.5. Hosts and devices should include a CRC record in response to any packet that includes a CRC record. However, a device or host that cannot calculate a CRC may simply ignore the CRC record and is not required to send a CRC. A device or host that receives a packet that does not contain a CRC record value shall accept the packet without returning an error message because of such absence.

5.6.2.2 The CRC record will consist of a record label (CRC), a label separator, an unsigned integer data field and a record separator.

5.6.2.3 The CRC field value is calculated according to the description given in Annex C.

5.6.2.4 When the CRC record is to be included in a packet, it appears immediately after the <RS> character. The <RS> character shall occur immediately after the record separator for the prior record, which would otherwise be the last record in the packet, and immediately before the record label for the CRC record. When no CRC record is included, the <RS> character shall appear immediately prior to the <GS> character.

5.6.2.5 The CRC value will be calculated on all of the characters in the message after (but not including) the start character, up to and including the <RS> character immediately prior to the record label of the CRC record.

EXAMPLE The part of a packet used in the CRC calculation. The underlined data are used to calculate the CRC.

```
<FS>REQ = INI<CR/LF>
DEV = GEN<CR/LF>
VEN = IGC<CR/LF>
MODEL = BLASTER1<CR/LF>
MID = IGC12345<CR/LF>
<RS>
CRC = 51242<CR/LF>
<GS>
```

5.6.3 Confirmation and timeouts

5.6.3.1 Confirmation message

The confirmation message consists of a single character, either an ASCII ACK (decimal 06) or an ASCII NAK (decimal 21). Whenever a packet is received, the receiver transmits an ACK to the sender to indicate its packet was received correctly (a positive acknowledgement). If the receiver believes that the packet was not received correctly, it should transmit an NAK to the sender (a negative acknowledgement). The sender should retry its transmission up to three times (a total of four transmissions answered by NAK) before giving up and posting an appropriate error message to the equipment operator.

EXAMPLE A failed session. When a receiver cannot recognize a message, or finds an invalid CRC in a packet, it sends a negative acknowledgement to the sender, which retries up to three times.

DEVICE	HOST
<FS>REQ=12<CR/LF> JOB=1234<CR/LF> <RS> CRC=Optional<CR/LF> <GS>	
	<NAK>
<FS>REQ=12<CR/LF> JOB=1234<CR/LF> <RS> CRC=Optional<CR/LF> <GS>	
	<NAK>
<FS>REQ=12<CR/LF> JOB=1234<CR/LF> <RS> CRC=Optional<CR/LF> <GS>	
	<NAK>
<FS>REQ=12<CR/LF> JOB=1234<CR/LF> <RS> CRC=Optional<CR/LF> <GS>	
	<NAK>

After the fourth negative acknowledgement, the device gives up and displays an error message. If the host's response is other than ACK or NAK, the device might help lead a troubleshooter in the direction of port parameters by displaying a message such as “Unrecognized confirmation”.

5.6.3.2 Confirmation timeout

A sender of a packet will wait up to 6 s for a confirmation message. If 6 s elapse after transmission of a packet without receipt of confirmation, the session is aborted, and the sender posts an appropriate error message to the equipment operator and does not retry its transmission.

5.6.3.3 Packet timeout

When a sender transmits a packet, receives a positive acknowledgement and further expects a packet from the receiver in return, the sender shall wait up to 12 s after its receipt of the positive acknowledgement for the receiver's packet. Should the receiver's packet fail to begin to arrive within that period, the session is aborted. The sender posts an appropriate error message to the equipment operator and does not retry its transmission.

5.6.3.4 Intercharacter timeout

When receipt of a packet pauses for more than 5 s prior to the receipt of the end character, the session is aborted and the receiver posts an appropriate error message to the equipment operator.

5.6.3.5 Defined timeouts

These may be altered by a host during initialization. Timeouts are expressed in seconds. The range of timeouts shall be 2 s to 255 s.

EXAMPLE A device uploading a frame tracer (TRC) packet showing timeouts.

NOTE Up to 5 s can pass between characters (not shown).

DEVICE	HOST
<FS>REQ=TRC<CR/LF> JOB=1234<CR/LF> [Optional Interface Records] <RS> CRC=Optional<CR/LF> <GS>	
up to 6 s pass…	
	<ACK>
up to 12 s pass…	
	<FS>ANS=TRC<CR/LF> JOB=1234<CR/LF> STATUS=0<CR/LF> [any Optional Interface Records that must be echoed] <RS> CRC=Optional<CR/LF> <GS>
up to 6 s pass…	
<ACK>	

up to 12 s pass…

```
<FS>ANS=TRC<CR/LF>
JOB=1234<CR/LF>
[All Mandatory Records for TRC Packet]
<RS>
CRC=Optional<CR/LF>
<GS>
```

up to 6 s pass…

<ACK>

up to 12 s pass…

```
<FS>ANS=TRC<CR/LF>
JOB=1234<CR/LF>
STATUS=0<CR/LF>
<RS>
CRC=Optional<CR/LF>
<GS>
```

up to 6 s pass…

<ACK>

5.7 Deprecated requirements

5.7.1 Mandatory records

The need for mandatory records has been superseded by the addition of the unknown data indicator. Mandatory records may still be supported for backward compatibility, but should not be used in new implementations. Any records that are contained in a record label list or a pre-set packet should be sent using the unknown data indicator if necessary. The following paragraph contains the old explanation of mandatory records:

Records that are mandatory are so designated in their definitions. Records not so designated may be presumed to be optional.

5.7.2 Quotation marks

5.7.2.1 When using the prior version of this International Standard, quotation marks should be placed around limited and text data. Parsers should strip the quotes before using the enclosed data. Starting with this version, the quotation marks are no longer necessary, but implementations may include them for backward compatibility.

5.7.2.2 The presence or absence of quotation marks should not cause an implementation to break.

5.7.2.3 When using the prior version of this International Standard, when the unknown data indicator (?) is used to replace a limited or text data value, it should be quoted following the rules for the data value it replaces.

5.7.2.4 Quotation marks do not count toward text length limits since they simply enclose data values. They do count against the 80-character line limit.

6 Sessions

6.1 General

6.1.1 Sessions consist of an exchange of messages between a device and a host.

6.1.2 Devices initiate all sessions by transmitting a request packet to a host.

6.1.3 Once begun, a session shall be concluded before another session can be initiated between a host and a given device. Sessions are never interleaved.

6.1.4 This International Standard establishes three categories of session:

— initialization,

— upload, and

— download.

The detailed structure of these types of session is described in 6.2 to 6.4. A file-based information transfer variant using the FIL device type is described in 6.5.

6.1.5 When a host receives a packet while not engaged in a session with a device, it expects a request packet. If it receives a packet which has the overall form of an ISO Standard packet, but which lacks a request type (REQ) record, the host shall use the literal data “ERR” in its answer type record, in the form “ANS=ERR”, along with a status record with a status code of 18.

6.1.6 The job ID record is mandatory in all packets outside of initialization. A device shall verify that the job ID received matches that are specified in the request packet. Hosts shall return the job ID specified in the same format it was received.

6.2 Initialization sessions

6.2.1 General

There are two types of initialization:

— auto-format,

— preset.

The purpose of initialization is to:

a) allow devices to provide information to hosts that remains constant between sessions;

b) provide a means by which hosts can identify devices and tailor their responses to them;

c) allow devices to negotiate with the host on various options within the interface, i.e. tracing formats and timeout values.

These features serve to minimize the amount of data that shall be exchanged during subsequent sessions.

Auto-format initialization allows devices to define a set of records, from the device records defined in this International Standard, that they wish to receive in data packets from the host in subsequent sessions. Upload devices should not use auto-format initialization unless they wish to receive data from the host in addition to uploading information to it.

Preset initialization allows devices to identify themselves as one of a set of preset device types defined in this International Standard. Both types of initialization provide for other identifying information to be conveyed which further allows hosts to tailor the information sent.

Devices that do not support initialization will use the preset device types in the REQ record (i.e. GEN, EDG, TRC, etc.).

Devices capable of supporting initialization should attempt initialization when powered on. They should use the request ID received from the host in subsequent REQ records. If a host receives a numeric request which it does not understand, it should send a response packet containing a status record with a status code of **5** (need initialization). Any device receiving a STATUS=5 from a host should attempt initialization. A device not capable of initialization should never receive a STATUS=5 from a host because it should never make a numeric request.

6.2.2 Composition of initialization sessions

Initialization sessions consist of the following exchange of messages.

a) A device transmits a request packet including a request type record with data INI to a host.

b) The host transmits a confirmation message to the device.

c) The host transmits a response packet to the device.

d) The device transmits a confirmation message to the host. If the host has sent a STATUS=15 (initialization not supported) record in its response packet [in step c)], the session ends at this point.

e) The device transmits a data packet to the host.

f) The host transmits a confirmation message to the device.

g) The host transmits a data packet to the device.

h) The device transmits a confirmation message to the host.

Table 5 — Summary of initialization session

DEVICE transmission	HOST transmission
Request packet, REQ=<INI>	
	Confirmation <ACK>
	Response packet, ANS=<INI>
Confirmation <ACK>	
Data packet, ANS=<INI>	
	Confirmation <ACK>
	Data packet, ANS=<INI>
Confirmation<ACK>	

6.2.3 Elements common to auto-format and preset initialization

6.2.3.1 A device may effect auto-format initialization or preset initialization if the host's response packet contains a status code of 0. A device may only effect preset initialization if the host's response packet contains a status code of 14. The set of interface records that the device includes in its data packet may be the same in either case. These records contain data that serve to identify the device to the host. In the case of preset initialization, this comprises the entirety of the device's data packet.

6.2.3.2 A device may override any of the parameters specified during initialization by including them in any subsequent request or data packet. The host should re-set the parameter to that specified during initialization for all sessions subsequent to the one in which the parameter was overridden. Should it be necessary for a device to change permanently a parameter set in initialization, it should issue a new initialization request.

6.2.3.3 Hosts that support auto-format or preset initialization shall manage the assignment of request type identifiers so that the identifiers do not repeat except over long periods. This is required to avoid a situation that could arise when a host is restarted: it could assign a request ID which, prior to its being re-started, had been used by a different machine than the one to which it is assigned after the re-start.

6.2.3.4 Both types of initialization are initiated with the same request packet.

EXAMPLE 1 An initialization request packet.

```
<FS>
REQ = INI
<RS>
<GS>
```

EXAMPLE 2 A response packet in which the host indicates its initialization capability.

```
<FS>
ANS = INI
STATUS = *initialization status
<RS>
<GS>
```

* Initialization status can be 0, 14 or 15

EXAMPLE 3 A portion of a device's initialization data packet containing interface records.

```
<FS>
ANS = INI
DEV = GEN
VEN = IGC
MODEL = B16
MNAME = Intergalactic Generator Co. Model 16
MID = QB485M
OPERID = JACK
<RS>
<GS>
```

6.2.4 Auto-format initialization

6.2.4.1 Upon receipt of a host's response packet with a status code of 0, a device may effect auto-format initialization by including a request definition in its data packet. The request definition consists of a record labelled DEF that contains a text field (called a device tag) that identifies the definition, followed by a variable number of record label list records, labelled “D”, each of which contains a series of record labels. The maximum number of characters in a single record label list, between the label separator and the record separator, is eighty (80) characters. The minimum number of fields in a record label list is one; however, the preferred method of expression is to place the maximum number of fields in each record label list record without exceeding the 80-character maximum. This minimizes the number of record label list records required. The request definition ends with an ENDDEF record that is labelled with the device tag found in the DEF record.

6.2.4.2 R, A, Z, and ZA labels should not appear in request definitions. The use of these labels is determined by TRCFMT and ZFMT that should be present as interface records in the initialization request.

6.2.4.3 In any case where BEVP is included in the record list, BEVM, BEVC, and FCRV shall also be included in the response if they are necessary.

6.2.4.4 A host will assign an integer request ID to each request definition set sent from the device. The request ID is returned to the device in the host's data packet. The device can then use this request ID when initiating sessions.

EXAMPLE 1 A device's initialization data packet in which the device (in this case, a generator) effects auto-format initialization. The device tag is “MYREQUEST”.

```
<FS>
ANS = INI
DEV = GEN
VEN = IGC
MODEL = B16
MNAME = Intergalactic Generator Co. Model 16
MID = QB485M
OPERID = JACK
DEF = MYREQUEST
D = FRNT;BACK;GBASE;GCROS;GAX;GBASEX;GCROSX;SAGRD
D = SAGBD;SAGCD;TIND;RNGH;RNGD;GTHK;LMATID;LAPM
D = DIA;BCTHK;BETHK;CRIB;GPRVA;GPRVM;KPRVA;KPRVM;PIND
ENDDEF = MYREQUEST
<RS>
<GS>
```

EXAMPLE 2 A host's initialization data packet in which the host assigns a request ID to the data set defined in the device's data packet. The host assigns request ID 1643 to device tag MYREQUEST.

```
<FS>
ANS = INI
STATUS = 0
DEF = MYREQUEST;1643
REM = I have assigned ‘1643' to ‘MYREQUEST'
<RS>
<GS>
```

EXAMPLE 3 A device's request packet for the data set identified above.

```
<FS>
REQ = 1643
JOB = 1234
<RS>
<GS>
```

EXAMPLE 4 A host's data packet transmitted in response to the above request.

```
<FS>
ANS = 1643
JOB = 1234
STATUS = 0
DO = B
FRNT = 6.21 ; 6.21
BACK = -5.00 ; -5.00
GBASE = 6.50 ; 6.75
GCROS = 6.75 ; 7.00
GAX = 135 ; 45
GBASEX = 6.48 ; 6.76
```

```
GCROSX = 6.76 ; 7.01
SAGRD = 3.81 ; 3.81
SAGBD = 4.50 ; 4.50
SAGCD = 4.10 ; 4.10
TIND = 1.53 ; 1.53
DIA = 75.0 ; 75.0
BCTHK = 15.0 ; 15.0
BETHK = 17.0 ; 17.0
CRIB = 70.0 ; 70.0
GPRVA = 140.0 ; 46.5
GPRVM = 2.4 ; 2.17
KPRVA = ? ; ?
KPRVM = ? ; ?
PIND = 1.498 ; 1.498
LMATID = 1 ; 1
LAPM = -1 ; -1
RNGH = 11.80 ; 11.80
RNGD = 60.0 ; 60.0
GTHK = 2.54 ; 2.52
<RS>
<GS>
```

EXAMPLE 5 Complete auto-initialization for an edger

DEVICE	HOST
<FS>REQ=INI<CR/LF> <RS> <GS>	
	<ACK> <FS>ANS=INI<CR/LF> STATUS=0; <RS> <GS>
<ACK>	
<FS>ANS=INI<CR/LF> DEV = EDG<CR/LF> VEN = GC<CR/LF> MODEL = ELITE<CR/LF> TRCFMT = 4;400;E;R<CR/LF> TRCFMT = 1;400;E;R<CR/LF> INFO=1; DEF = FIRSTREQ<CR/LF> D = TRCFMT;*R;HBOX;VBOX;*CIRC;FCRV<CR/LF> ENDDEF = FIRSTREQ<CR/LF> <RS> <GS>	
	<ACK>
	<FS>ANS=INI<CR/LF> STATUS=0<CR/LF> DEF= FIRSTREQ;*number<CR/LF> TRCFMT=4;400;E;R<CR/LF> INFO=0<CR/LF> <RS> <GS>
<ACK>	

*number = number to be used in subsequent requests by device.

6.2.5 Preset initialization

6.2.5.1 A device may effect preset initialization when a host responds to a device's request for initialization with a status code of 14 or if the host responds with a status code of 0, but the device does not support auto-format initialization.

6.2.5.2 The device's data packet should include the following interface records:

— device type,

— vendor,

— model ID.

The host may be able to use this information to tailor its responses to requests based on the descriptive information provided. It is anticipated that manufacturers of devices may publish the set of records required by their various devices, especially in those cases in which the devices do not support auto-format initialization. In the event that the host does not recognize the vendor and model ID specified, it shall send the entire set of records specified for the indicated device type as shown in Annex A. Other interface records, such as operator ID, machine ID and serial number, should be included if available.

NOTE When a device type of “DNL” is specified, and the vendor and model ID are not recognized, the host sends the entire set of device records for the indicated job ID.

6.2.5.3 The device's data packet shall not contain a request definition record (“DEF”). It is the absence of this record that provides an indication to the host that preset, rather than auto-format initialization, is being requested.

6.2.5.4 The host's subsequent data packet will contain a request ID that the device will use for subsequent requests. Because there is no request definition, there is no device tag, however there is a field separator as shown in example below.

EXAMPLE A session in which a host supports only preset initialization.

DEVICE	HOST
<FS>REQ=INI<CR/LF> <RS> <GS> <ACK>	
	<FS>ANS=INI<CR/LF> STATUS=14;PRESET only<CR/LF> <RS> <GS>
<ACK> <FS>ANS=INI<CR/LF> DEV = TRC VEN = GC MODEL = FTX MNAME = Company XYZ's Tracer MID = 24076 OPERID = JILL TRCFMT = 1;400;E;R TRCFMT = 4;400;E;R [any other Optional Interface RECORDS]<RS> <GS> <ACK>	

```
                                        <FS>ANS=INI<CR/LF>
                                        STATUS = 0<CR/LF>
                                        DEF = ;1234
                                        TRCFMT = 4;400;E;R
                                        <RS>
                                        <GS>

<ACK>
```

6.2.6 No initialization

6.2.6.1 When neither type of initialization is supported by a host or device, the parameters that would be transmitted from host to device during initialization would have to be included in the data packets transmitted from host to device during each session. Because this is quite inefficient it is not recommended. It would be preferable for the host and device's parameters to be synchronized manually.

6.2.6.2 The device's request packets should contain device type, vendor and model records to allow the host to better tailor the set of records it returns to the device, and such informational records (machine ID, operator ID) as the device supports. When a host receives a request of this type, and does not recognize the vendor and/or model records, it will send all of the records defined for the device type. Likewise, when a device type of DNL is specified, all records will be downloaded.

EXAMPLE A device's request packet, after no initialization has taken place.

```
<FS>
REQ = GEN
JOB = 1234
VEN = IGC
MODEL = B16
MNAME = Intergalactic Generator Co. Model 16
MID = QB485M
OPERID = JACK
<RS>
<GS>
```

6.2.7 Minimum host support for initialization

A host shall, as a minimum, be able to respond to an initialization request with a response packet containing a **STATUS=15** (initialization not supported) record.

EXAMPLE A session with a host that does not support initialization.

DEVICE	HOST
<FS>REQ=INI<CR/LF> <RS> <GS>	
	<ACK>
	<FS>ANS=INI<CR/LF> STATUS=15; No can do<CR/LF> <RS> <GS> <ACK>

6.2.8 Special initialization requirements for devices using trace data

Hosts and devices need to negotiate a tracing format that will be acceptable to both. Devices shall identify one or more desired formats by sending one or more TRCFMT records in the initialization data packet or, in the absence of initialization, in the request packet. Each TRCFMT record shall specify one of the tracing types the device would like to use in the order it would prefer to use them (typically from most compact to least compact). The host shall send back in its response one of the TRCFMT records to indicate which one it supports and will actually use. (For more information, see also the description of status code 17 in A.4.)

The same method is used to negotiate sag data formats. It is possible for hosts to not support sag data; in such cases, the host shall return “ZFMT=0” to provide the device a clear indication that sag data cannot be accepted or provided.

All hosts and devices conforming to this International Standard shall support the ASCII absolute trace format.

EXAMPLE 1 An initialization session showing trace format negotiation.

DEVICE	HOST
<FS>REQ=INI<CR/LF> <RS> <GS> <ACK>	
	<FS>ANS=INI<CR/LF> STATUS=14;PRESET only<CR/LF> <RS> <GS>
<ACK> <FS>ANS = INI<CR/LF> DEV = TRC<CR/LF> VEN = GC<CR/LF> MODEL = FTX<CR/LF> TRCFMT = 4;512;E;R<CR/LF> TRCFMT = 4;400;E;R<CR/LF> TRCFMT = 1;512;E;R<CR/LF> TRCFMT = 1;400;E;R<CR/LF> <RS> <GS>	
	<ACK> <FS>ANS=INI<CR/LF> STATUS=0<CR/LF> TRCFMT=4;400;E;R<CR/LF> <RS> <GS>
<ACK>	

EXAMPLE 2 An initialization session in which the host does not support any of the device's proposed trace formats.

DEVICE	HOST
<FS>REQ=INI<CR/LF> <RS> CRC=Optional<CR/LF> <GS> <ACK>	
	<FS>ANS=INI<CR/LF> STATUS=14;PRESET only<CR/LF> <RS> <GS>
<ACK> <FS>ANS=INI<CR/LF> DEV=TRC VEN= GC MODEL= FTX TRCFMT=1;400;E;R TRCFMT=4;400;E;R TRCFMT=1;512;E;R TRCFMT=4;512;E;R <RS> CRC=Optional<CR/LF> <GS> <ACK>	
	<FS>ANS=INI<CR/LF> STATUS=529; 400 ? 512 ?<CR/LF> <RS> <GS>
<ACK>	

6.3 Upload sessions

6.3.1 Upload sessions consist of the following exchange of messages:

a) A device transmits a request packet to a host.

b) The host responds with a confirmation message.

c) The host transmits a response packet to the device.

d) The device transmits a confirmation message to the host.

e) The device transmits its data packet to the host.

f) The host transmits a confirmation message to the device.

g) The host transmits a response packet to the device.

h) The device transmits a confirmation message to the host.

Table 6 — Summary of upload session

DEVICE Transmission	HOST Transmission
Request packet, REQ=<Request>	
	Confirmation <ACK>
	Response packet, ANS=<Request>
Confirmation <ACK>	
Data packet, ANS=<Request>	
	Confirmation <ACK>
	Response packet, ANS=<Request>
Confirmation <ACK>	

6.3.2 If any of the confirmation messages is a negative acknowledgement, the sender shall retry its transmission up to three times. If a negative acknowledgement is received after the third retry (i.e. the fourth attempt), the session terminates.

6.3.3 An upload session can also fail if either of the host's response packets contains a status record whose field value is non-zero. In this case, the session ends after the device's subsequent confirmation message.

6.3.4 The content of each upload session is determined by the contents of the request type (REQ) record. At this time, the preset request types that indicate upload sessions are TRC, INF, MNT, INS and the generic UPL.

EXAMPLE An abstract upload session.

DEVICE HOST

```
<FS>REQ=[*Device type]<CR/LF>
JOB=1234<CR/LF>
[Optional interface RECORDS]
<RS>
<GS>
```

```
<ACK>
<FS>ANS=[*Device type]<CR/LF>
JOB=1234<CR/LF>
STATUS=0<CR/LF>
[any RECORDS that must be
echoed]
<RS>
<GS>
```

```
<ACK>
<FS>ANS=[*Device type]<CR/LF>
JOB=1234<CR/LF>
[All Mandatory Records for *Device type Packet]
<RS>
<GS>
```

```
<ACK>
<FS>ANS=[*Device type]<CR/LF>
JOB=1234<CR/LF>
STATUS=0<CR/LF>
<RS>
<GS>
```

```
<ACK>
```

NOTE *Device type = TRC, UPL INF, MNT, INS or request ID sent by host during initialization.

6.3.5 UPL, the generic upload session request type, allows a device to upload to a host any subset of the defined records. This International Standard does not specify what actions hosts shall take upon receipt of such data.

NOTE What a host might do with records uploaded via a UPL request but which are normally downloaded to a device is undefined. Replacing such values as it may currently have would be a reasonable action, but is not explicitly mandated.

6.3.6 INF is sent from devices to hosts to indicate that the process associated with the most recent request for the job indicated has been completed. It is a special, abbreviated variant of upload request, as it contains all of the data to be conveyed in the request packet itself. The INF request type will contain only request type, job, status, and (optionally) CRC and model ID records. The host shall respond to the INF packet with a confirmation message.

EXAMPLE An INF upload session. First, a device makes a request (in this example, a GEN request).

DEVICE	HOST
<FS> REQ=GEN<CR/LF> JOB=1234<CR/LF> <RS> CRC=Optional<CR/LF> <GS>	
	<ACK> <FS> ANS=GEN<CR/LF> JOB=1234<CR/LF> [Generator records…] <RS> CRC=Optional<CR/LF> <GS>
<ACK>	

Then, when the generator has finished its processing, it sends the following indicating a normal completion of its process. The host responds only with a confirmation message.

<FS> REQ=INF<CR/LF> JOB=1234<CR/LF> STATUS=0<CR/LF> <RS> CRC=Optional<CR/LF> <GS>	
	<ACK>

6.3.7 MNT is a request that allows a device to transmit machine status and configuration information to its host. This is a special, abbreviated variant of upload request, as it contains all of the data to be conveyed in the request packet itself. The host shall respond to the INF packet with a confirmation message.

The contents of the MNT packet are, in this version, entirely up to the manufacturers of devices who implement this feature. As such, it is reasonable to expect that the data packet may consist largely of experimental records having record labels with a leading underscore. Hosts can support this feature by saving the records received to a file.

6.4 Download sessions

6.4.1 Download sessions consist of the following exchange of messages:

a) A device transmits a request packet to a host.

b) The host responds with a confirmation message.

c) The host transmits a data packet to the device.

d) The device responds with a confirmation message.

Table 7 — Summary of download session

DEVICE Transmission	HOST Transmission
Request packet, REQ=<Request>	
	Confirmation <ACK>
	Data packet, ANS=<Request>
Confirmation <ACK>	

6.4.2 If any of the confirmation messages is a negative acknowledgement, the sender shall retry its transmission up to three times. If a negative acknowledgement is received after the third retry (i.e. the fourth attempt) the session terminates.

EXAMPLE A download session showing trace format negotiation with no previous initialization. *Device Type* is one of PTG, EDG, FBK, SBK, GEN, AGN, COA, FSG, LMD or DNL.

DEVICE HOST

```
<FS>REQ=[Device Type]<CR/LF>
JOB=1234<CR/LF>
VEN = GC<CR/LF>
MODEL = FTX<CR/LF>
TRCFMT = 4;512;E;R<CR/LF>
TRCFMT = 4;400;E;R<CR/LF>
TRCFMT = 1;512;E;R<CR/LF>
TRCFMT = 1;400;E;R<CR/LF>
<RS>
CRC=Optional<CR/LF>
<GS>
```

```
                                        <ACK>
                                        <FS>ANS=[Device Type]<CR/LF>
                                        JOB=1234<CR/LF>
                                               STATUS=0<CR/LF>
                                        DO=B<CR/LF>
                                        [All Mandatory Records for Device type Packet]
                                               TRCFMT=4;400;E;R;F<CR/LF>
                                        R=2479;2583;2605;2527;2394;2253;2137;
                                               2044;1975;1935<CR/LF>
                                        R=1922;1939;1989;2072;2184;2322;2471;
                                               2599;2645;2579<CR/LF>
                                               etc…
                                        <RS>
                                        CRC=Optional<CR/LF>
                                               <GS>
```

```
<ACK>
```

6.4.3 The content of each download session is determined by the contents of the request type (REQ) record. At this time, the preset request types that indicate download sessions are of PTG, EDG, FBK, SBK, GEN, AGN, COA, FSG, LMD, and the generic DNL.

EXAMPLE An abstract preset packet download. The device type is one of PTG, EDG, FBK, SBK, GEN, AGN, COA, FSG, LMD, DNL, or request ID sent by host during initialization.

DEVICE	HOST
<FS>REQ=[Device Type]<CR/LF> JOB=1234<CR/LF> [Optional Interface Records] <RS> CRC=Optional<CR/LF> <GS>	
	<ACK>
	<FS>ANS=[Device Type]<CR/LF> JOB=1234<CR/LF> STATUS=0<CR/LF> DO=B<CR/LF> [All Records for Device type Packet] <RS> <GS>
<ACK>	

6.5 File-based information transfer

6.5.1 The device type FIL shall be used to identify the contents of an OMA data file.

6.5.2 The file will contain printable ASCII characters, line terminators, and depending on the source platform, may contain an end-of-file marker. The line terminators and end-of-file characters may differ depending on the hardware and/or software platform on which the file is produced.

6.5.3 Lines in the file correspond to records as defined in this International Standard.

6.5.4 The contents of the file will be formatted according to the rules for data packets specified in this International Standard.

6.5.5 The contents of the file will differ from the contents of a data packet in the following ways:

a) the start, stop, and CRC position characters will not be included in the file;

b) the CRC Record will not be included in the file;

c) the STATUS record will not be included in the file;

d) the request type record shall have the value “FIL”, e.g. REQ = FIL.

6.5.6 The file may contain trace data, in which case, the only format allowed is format 1, ASCII.

7 Other requirements

7.1 RS-232 Communications parameters

Default parameters are 9 600 baud, 8 data bits, 1 stop bit and no parity. Hosts and devices shall default to these parameters. Other settings could be implemented on-site by installation personnel. Flow control, if implemented, shall be accomplished using RS-232 control lines. Software flow control (XON/XOFF) is *not* implemented under this International Standard. The XON and XOFF characters are, however, reserved.

7.2 Operator messages

Devices should provide operators with messages that report on the progress of sessions; e.g. a device could display “Sending Request…” when transmitting a request packet and change this to “Awaiting Response…” after receipt of the confirmation message.

7.3 Host requirement

Hosts shall allow multiple initializations or request types on each communications port in use.

Annex A
(normative)

Record labels

A.1 Device records

A.1.1 Table A.1 contains all of the device records defined for use by systems conforming to this International Standard, arranged alphabetically. Under the column labelled “Data type”, the proper characteristics of the data associated with each label is indicated.

A.1.1.1 The field separator character identifies the chirality of the record.

A.1.1.2 The lack of a field separator (semi-colon) indicates that only one value is expected, i.e. the record is not chiral.

EXAMPLE A non-chiral record descriptor.

```
LABEL   type        description
```

A.1.1.3 A trailing semi-colon indicates that chiral data is expected. Data for either eye may be empty, but the field separator shall be present.

EXAMPLE A chiral record descriptor.

```
LABEL   type;       description
```

A.1.1.4 Square brackets around the semi-colon indicate the start of optional left eye data which, if present, follows the specification for chiral data. If the semi-colon and left eye data are absent, the value shall apply to both right and left eyes. For example, if only one circumference is supplied, it is assumed to apply to both right and left eyes.

EXAMPLE An optional chiral record descriptor.

```
LABEL   type[;]     description
```

A.1.1.5 The data type indicates constraints on the kind of data that may appear in the record's fields.

A.1.1.5.1 Numeric fields may or may not contain a decimal point. If a decimal point is present, hosts and devices should be able to correctly parse any degree of precision, including zero. Numeric format should be flexible, but reasonable.

A.1.1.5.2 Integer fields shall not contain decimal points.

A.1.1.5.3 The ± symbol indicates that a number may be positive or negative. The absence of the symbol indicates that a value is expected to be positive.

A.1.1.5.4 Square brackets [ ] around any element other than the semi-colon indicate that the enclosed item is optional.

A.1.1.6 The following units of measure are used in Table A.1:

— millimetres;

— dioptres: the index of refraction to be used will be specified;

— degrees.

Table A.1 — Device records

Record label	Data type	Description
A	integer;integer;…	Angle data for radius data (angular locations of trace radii).
ACCN	text	Account number.
ACOAT	text[;]	Applied coating.
ADD	numeric;	Addition power if multifocal, progressive or executive type lens (dioptres).
ADD2	numeric;	2nd (upper) Addition power (dioptres).
AVAL	numeric;	Value indicating the height of a semi-finished blank prior to generating, measured relative to the machine's reception chuck on the centreline of the chuck.
AX	numeric;	Prescribed cylinder axis. May be 0 to 180 (degrees). Notice that this may be different from GAX which is generator AXIS.
BACK	± numeric;	Blank back curve (dioptres).
BCOCIN BCOCUP	± numeric;	Blank geometrical centre to optical centre In & Down (mm). Useful for uncuts where frame information is not available. +IN means O.C. towards nasal with respect to blank geometrical centre. –IN means O.C. towards temporal with respect to blank geometrical centre. +UP means O.C. is above blank geometrical centre. –UP means O.C. is below blank geometrical centre.
BCSGIN BCSGUP	± numeric; ± numeric;	Blank centre to segment In & Down, i.e. manufacturer's stated segment position relative to geometric centre of the blank. +IN means segment towards nasal with respect to blank centre. –IN means segment towards temporal with respect to blank centre. +UP means segment is above blank centre. –UP means segment is below blank centre.
BCTHK	numeric;	Blank centre thickness (mm).
BETHK	numeric;	Blank edge thickness (mm).
BEVC	numeric;	Bevel curve. A dioptre value (1,530 index) for the bevel curve to follow.
BEVM	numeric;	Bevel modifier. Can be percentage or distance in mm based on value of BEVP. Percentage should be expressed as a number between 0 and 100.
BEVP	integer[;]	Bevel position 0 – manual 1 – follow front (BEVM = mm from front) 2 – percent back (BEVM = % from front) 3 – frame curve (BEVC = curve to follow, BEVM = mm from front) 4 – 50/50 5 – follow back (BEVM = mm from back) 6, 8, 9 – Unused 7 – automatic (use edger settings) 10 – free float

Table A.1 (*continued*)

Record label	Data type	Description
BLKB	± numeric;	Block base curve (dioptres).
BLKCMP	± numeric;	Generator thickness compensation for block/lens curve mismatch.
BLKD	numeric;	Block diameter (mm).
BLKTYP	integer;	Integer block type. A number agreed to by device and host to indicate which of several block tables to use. Can be used to distinguish between different types or materials of blocks.
BPRVA	numeric;	Meridian at which BPRVM is located, 0 to 360 (degrees).
BPRVM	numeric;	Amount of prism in semi-finished blank (degrees or dioptres, see PIND RECORD).
BRGSIZ	integer	Bridge size of frame.
BSIZ	± numeric[;]	Boxed lens size: sizing to be applied to shape by horizontal lens size. A device that accepts this record is expected to be able to size any shape sent to it by the amount indicated.
BVD	integer;	Back vertex distance (mm)
CIRC	numeric[;]	Circumference.
CLAMP	integer;	Edger clamping pressure ranging from 1 to 10 where 1 is low pressure and 10 is high pressure. Machines with smaller ranges should convert the value given to a comparable value for their use.
CLIENT	text	Name of the end user or customer.
COLR	text[;]	Lens colour abbreviation.
CPID	integer	Coating process ID 0 – None Others agreed on by device and host
CPRVA	numeric;	Meridian at which CPRVM is located, 0 to 360 (degrees).
CPRVM	numeric;	Magnitude of compensated prism to grind. Prism is compensated for the effect of tilting the lens off the axis of the cutter. See also record GPRVM (degrees or dioptres, see PIND RECORD).
CRIB	± numeric;	Crib diameter (mm). A value of zero (0) will mean no cribbing. A positive value will mean the crib diameter, a negative value will mean the take off amount (from blank diameter) intended for touching up uncut lenses.
CSIZ	± numeric[;]	Circumference sizing: sizing to be applied to shape by circumference. A device that accepts this record is expected to be able to size any shape sent to it by the amount indicated.
CTHICK	numeric;	Finished centre thickness (mm at distance O.C.).
CYL	± numeric;	Rx cylinder power (dioptres).
DBL	numeric	Distance between lenses (mm).
DIA	numeric;	Blank diameter (mm).

Table A.1 (*continued*)

Record label	Data type	Description
DRILL	literal;± numeric; ± numeric; [± numeric]; [± numeric]; [± numeric]; [± numeric]; [literal]; [± numeric]; [± numeric]	This record has the form DRILL = R|L|B; x-start; y-start;[diameter];[x-end];[y-end]; [depth]; [B|F|A]; [lateral angle]; [vertical angle] The first field is the eye drill data is to be applied to: R = right eye L = left eye B = both eyes. The xy coordinates oriented as right eye data. The second field is the Cartesian x coordinate (mm) of hole start position. The third field is the Cartesian y coordinate (mm) of hole start position. The fourth field is the hole diameter (mm). If this field is empty, the hole will be drilled to diameter of tool. The fifth field is the Cartesian x coordinate (mm) of hole end position. If this field is empty, a normal (round) hole will be drilled. The sixth field is the Cartesian y coordinate (mm) of hole end position. If this field is empty, a normal (round) hole will be drilled. The seventh field is the depth (mm) for drilling a blind hole. If this field is empty, hole will be drilled completely through lens. The eighth field is the lens surface hole should be drilled normal to: B = normal to back lens surface F = normal to front lens surface A = specified drill angle from next field The ninth field is the lateral hole-drilling angle. (degrees) The tenth field is the vertical hole-drilling angle. (degrees) The Cartesian coordinates for the hole start and end positions are referenced viewing the front surface of the lens, with its origin at the Frame Box centre and: +x = right eye towards nasal, left eye towards temple $-x$ = right eye towards temple, left eye towards nasal +y = above frame centre $-y$ = below frame centre See Figure A.1. Fields 8, 9 and 10 are used as follows: The hole is drilled either parallel to a “drill-reference axis” or at a specified angle relative to this axis. The drill-reference axis is the normal to the front or back surface at box centre. The letter F or B in field 8 indicates use of the front-surface drill-reference axis or back-surface drill-reference axis. The letter A in field 8 indicates that the hole is drilled at an angle to the front-surface drill-reference axis. Field 8 may contain only a single letter: F, B or A. The lateral and vertical components of the angle appear in fields 9 and 10, respectively. If the letter A appears in field 8 and both fields 9 and 10 are empty, the hole will be drilled parallel to the front-surface drill-reference axis. If field 8 does not contain a valid letter (F, B or A) the hole will be drilled parallel to the front-surface drill-reference axis. The lateral and vertical angles of the drilled hole are expressed in degrees with zero being parallel to the drill-reference axis. The lateral angle for the right lens has a positive value if the centreline of the hole, when projected forward from the lens front surface, moves toward the nasal side. In the case of the left lens, a positive lateral angle results in the centreline moving toward the temple side. The vertical angle for both right and left lens has a positive value if the centreline of the hole, when projected forward from the lens front surface, moves toward the top of the lens. See Figures A.2 and A.3. Each hole will have a separate DRILL record, so there will typically be multiple occurrences of the DRILL record within an OMA message. If FBFCIN and FBFCUP fields are present, indicating shape is to be decentred, DRILL x and y coordinates shall be altered appropriately.

Table A.1 (*continued*)

Record label	Data type	Description
EECMP	± numeric;	Elliptical error curve compensation (dioptres based on TIND). This compensation is applied to GCROS.
ELLH	numeric;	Cribbing ellipse height (mm). Used with the record CRIB to form an ellipse.
EPRESS	integer	Pressure applied to lens against edger wheels 0 – undefined 1 – very fragile 2 - soft pressure 3 – medium pressure 4 – hard pressure
ERDRIN, ERDRUP	± numeric;	Engraving reference point to distance reference point offsets. +IN means distance reference point is towards nasal relative to the engraving reference point. –IN means distance reference point is towards temporal relative to the engraving reference point. +UP means distance reference point is above the engraving reference point. –UP means distance reference point is below the engraving reference point.
ERNRIN, ERNRUP	± numeric;	Engraving reference point to near reference point offsets. +IN means near reference point is towards nasal relative to the engraving reference point. –IN means near reference point is towards temporal relative to the engraving reference point. +UP means near reference point is above the engraving reference point. –UP means near reference point is below the engraving reference point.
ETYP	± integer	Type of edge to cut on lens: –1 – uncut 1 – bevel 2 – rimless 3 – groove 4 – mini-bevel 5…32 reserved 33…127 defined between host and device.
EYESIZ	integer	Nominal lens size of frame.
FBFCIN, FBFCUP	± numeric;	Finish block to frame centre offsets. Causes decentring of shape on edger by moving frame centre with respect to the finish block. +IN means frame centre is towards nasal relative to the finish block. –IN means frame centre is towards temporal relative to the finish block. +UP means frame centre is above the finish block. –UP means frame centre is below the finish block.

Table A.1 (*continued*)

Record label	Data type	Description
FBOCIN, FBOCUP	± numeric; ± numeric;	Finish block to O.C. offsets (mm). Can be used for single vision and for multifocals. +IN means O.C. is towards nasal relative to the finish block. –IN means O.C. is towards temporal relative to the finish block. +UP means O.C. is above the finish block. –UP means O.C. is below the finish block.
FBSGIN, FBSGUP	± numeric; ± numeric;	Finish block to segment offsets (mm). If sent for single vision, assumed the same as FBOCUP/FBOCIN defined below. +IN means segment is towards nasal relative to the finish block. –IN means segment is towards temporal relative to the finish block. +UP means segment is above the finish block. –UP means segment is below the finish block
FCOAT	text[;]	Factory coating.
FCOCIN, FCOCUP	± numeric; ± numeric;	Frame centre to optical centre offsets, i.e., O.C. inset and drop (mm). +IN means O.C. towards nasal with respect to the frame centre. –IN means O.C. towards temporal with respect to the frame centre. +UP means O.C. is above the frame centre. –UP means O.C. is below the frame centre.
FCOL	text	Colour name of frame.
FCRV	± numeric [;]	Frame curve.
FCSGIN, FCSGUP	± numeric;	Frame centre to segment offsets, i.e. segment inset & drop (mm). +IN means segment towards nasal with respect to the frame centre. –IN means segment towards temporal with respect to the frame centre. +UP means segment is above the frame centre. –UP means segment is below the frame centre.
FINCMP	± numeric;	Fine off allowance compensation (mm). This compensation is applied to GTHK.
FLATA	numeric;	Flattening dimension vertical (for square crib) (mm). Used with record CRIB.
FLATB	numeric;	Flattening dimension horizontal (for square crib) (mm). Used with record CRIB.
FMAT	text	Material of frame (e.g. METL).
FMFR	text	Manufacturer of frame.
FRAM	text	Name of frame.
FRNT	± numeric;	Blank true front curve for power calculations (TIND dioptres).
FPINB	numeric;	Edger front pin bevel width in mm measured along cut part of lens.
FTTHK	numeric;	First touch thickness, i.e. thickness at which generator wheel first touches the lens (at any location).

Table A.1 (*continued*)

Record label	Data type	Description
FTYP	integer	Integer frame type. 0 – Undefined 1 – Plastic 2 – Metal 3 – Rimless 4...127 – reserved
GAX	numeric;	Cylinder axis for surfacing machines, 0 to 180 (degrees).
GBASE	± numeric;	Generator base curve (rounded, TIND dioptres).
GBASEX	± numeric;	Generator base curve (unrounded, TIND dioptres).
GCROS	± numeric;	Generator cross curve (rounded, TIND dioptres).
GCROSX	± numeric;	Generator cross curve (unrounded, TIND dioptres).
GDEPTH	numeric;	Groove depth in mm when ETYP = 3.
GPRVA	numeric;	Meridian at which to generate GPRVM, 0 to 360 (degrees).
GPRVM	numeric;	Amount of prism to generate. See also records KPRVM and CPRVM (degrees or dioptres, see PIND RECORD).
GTHK	numeric;	Generator thickness at centre of surface block (mm).
GWIDTH	numeric;	Groove width in mm when ETYP = 3.
HBOX	numeric [;]	Horizontal boxed lens size of frame (mm).
IFRNT	± numeric;	Blank implied front curve. The SAGRD value converted to a dioptric curve based on TIND.
INSADD	± numeric;	Measured add power (dioptres).
INSAX	± numeric;	Measured axis (degrees).
INSCTHK	± numeric;	Measured centre thickness (mm).
INSCYL	± numeric;	Measured cylinder power (dioptres).
INSPRVA	± numeric;	Measured prism base setting (degrees).
INSPRVM	± numeric;	Measured magnitude of prism (degrees or dioptres, see PIND RECORD).
INSSGIN	± numeric;	Measured inset of segment (mm).
INSSGUP	± numeric;	Measured vertical offset of segment (mm).
INSSPH	± numeric;	Measured sphere power (dioptres).
IPD	numeric;	Monocular centration distance (mm).
KPRVA	numeric;	Meridian at which to block KPRVM, 0 to 360 (degrees).
KPRVM	numeric;	Magnitude of BLOCKED prism. (degrees or dioptres, see PIND RECORD).
LAPBAS	± numeric;	Lap base curve (rounded, TIND dioptres).
LAPBASX	± numeric;	Lap base curve (unrounded, TIND dioptres).
LAPBIN	± text;	Lap location.
LAPCRS	± numeric;	Lap cross curve (rounded, TIND dioptres).
LAPCRSX	± numeric;	Lap cross curve (unrounded, TIND dioptres).

Table A.1 (*continued*)

Record label	Data type	Description
LAPM	integer;	Integer lap material number. A number agreed to by device and host to access lap material set-up tables. The value zero is reserved to mean undefined.
LAPPRB	integer;	Lap probing method: 0 – No HOST control 1 – Normal probing 2 – Re-true probing 3 – Probing OFF
LENPRB	integer;	Lens probing method: 0 – No HOST control 1 – Probing OFF 2 – OFF centre probing 3 – ON centre probing 4 – Pre-cut probing
LIB	text;	Library name under which to store trace.
LIND	numeric;	Index of lens material.
LMATID	integer;	Integer material number. A number agreed to by the device and host to access material set-up tables on both. The value zero is reserved to mean undefined.
LMATNAME	text[;]	Name of lens material (e.g. “GLASS”, “PLASTIC”).
LMATTYPE	integer;	Integer basic material type. 0 – Undefined/invalid 1 – Plastic 2 – Polycarbonate 3 – Glass 4 – Pattern 5 – Hi-index 6 – Trivex 7...127 – reserved
LMFR	text[;]	Blank manufacturer.
LNAM	text[;]	Name of lens style “SV”, “VX INFINITY”.
LSIZ	numeric;	Manufacturer's nominal blank diameter (mm).
LTYP	literal[,];	Lens type, constructed using the following 2 letter codes. All codes that apply (up to 4) should be concatenated together and separated by commas; e.g. an aspheric bifocal should be indicated as AS,BI. AS – Aspheric AT – Atoric (contact lens) BI – Bifocal CT – Curve top DS – Double segment EX – E-line multifocal FT – Flat top LT – Lenticular PR – Progressive addition QD – Quadrifocal RD – Round SV – Single vision TR – Trifocal

Table A.1 (*continued*)

Record label	Data type	Description
LTYPE	literal[literal] [literal][literal];	Lens type, constructed using the following 2 letter codes separated by spaces. Exactly one identifier shall be selected from for the “Lens type” set. Depending on the selected “Lens type”, zero or one identifiers are selected from the “Intermediate power” set, and zero, one or two identifiers are selected from the “Segment type” set. In the cases of “SV” and “NV”, no elements from the second or third sets may appear. In the case of Lens type “BI”, one and only segment type appears. In the case of lens type “TR”, one or two segment types may occur; the only allowable use of two segment types is “EX FT” which describes the “E/D” style trifocal. In the case of lens type “PR”, no segment type may appear. In the case of lens types “QD” and “DS”, one or two segment types may appear. When one segment type appears, the two segments are the same; when two segment types appear, the first describes the upper segment and the second, the lower. Any number of identifiers may appear from the “Special characteristics” set in conjunction with any combination of lens type, intermediate power, and segment type. The special characteristics identifiers AS and AT are mutually exclusive; if one appears, the other may not. Special characteristics identifiers, when present, shall appear in the order in which they are defined herein. <u>Set one: Lens type</u> SV – Single vision NV – Near-variable-focus BI – Bifocal TR – Trifocal PR – Progressive addition QD – Quadrifocal DS – Double segment <u>Set two: Intermediate power</u> 40 – 40 % intermediate power 60 – 60 % intermediate power 70 – 70 % intermediate power <u>Set three: Segment type</u> CT – Curve top EX – E-line multifocal FT – Flat top RD – Round <u>Set four: Special characteristics</u> AS – Aspheric AT – Atoric (contact lens) LT – Lenticular

Table A.1 (*continued*)

Record label	Data type	Description
MBASE	± numeric;	Nominal lens box top base curve (e.g. 2, 4, 6, etc.)
MCIRC	numeric	Measure circumference and cut to tolerance. 0 = no measurement. A value larger than zero should be viewed as a tolerance.
MBD	numeric;	Minimum blank diameter (mm).
NOD	numeric;	Near object distance (mm)
NPD	numeric;	Monocular near centration distance (mm).
OCHT	numeric;	Vertical O.C. height measured from the (frame) lower boxed tangent.
OPC	text;	10-digit optical product code (OPC) for lenses to be used.
OPCF	text;	Front wafer 10-digit product code.
OPCB	text;	Back wafer 10-digit product code.
OTHK	numeric;	Generator thickness at the prism reference point (mm), used in conjunction with SBOCIN/SBOCUP.
PADTHK	numeric;	Pad thickness (mm).
PANTO	integer;	Pantoscopic angle (degrees)
PINB	numeric;	Pin bevel width in mm (0 = no pin bevel).
PIND	numeric;	The index of prism value, if prism expressed in dioptres, otherwise the value shall be zero if prism is expressed in degrees.
POLISH	integer	Polishing mode. Currently 0 means no polish, 1 means polish. All other values are reserved for future use.
PREEDGE	integer;	Enable = 1/Disable = 0. Pre-edging of lens by generator.
PRVA	numeric;	Rx Prism base setting, 0 to 360 (degrees). [Prism base setting means direction of the line from apex to base in a principal section of a prism (see 10.7 of ISO 13666:1998).]
PRVM	numeric;	Rx Prism in dioptres, including equithinning prism to inspect at O.C.
R	integer;integer;...	Radius data, see prior definition.
RMT	text;	Information which could be used to link tracings with Rx orders delivered by some means other than that by which the tracing is delivered.
RNGCMP	± numeric;	Thickness compensation for prism or blocking ring thickness (mm). This compensation is applied to GTHK.
RNGD	numeric;	Ring diameter (mm).
RNGH	numeric;	Ring height (mm).
RPRVA	numeric;	Meridian at which to generate RPRVM, 0 to 360 (degrees).
RPRVM	numeric;	Amount of prism to generate when using the SBOCIN/SBOCUP records for decentration.
RXNM	text	Rx number
SAGBD	± numeric;	Sag at blank diameter (mm). Indicates the location of the lens front surface at its edge prior to cribbing.
SAGCD	± numeric;	Sag at crib diameter (mm). Indicates the location of the lens front surface at its edge after cribbing.
SAGRD	± numeric;	Sag at ring diameter (mm). Indicates the position of the lens front surface in the machine.

Table A.1 (*continued*)

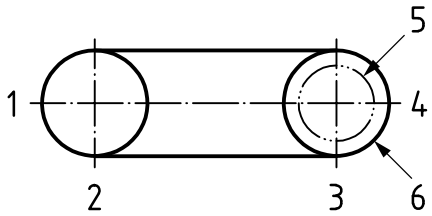
Record label	Data type	Description
SBBCIN, SBBCUP	± numeric; ± numeric;	Surface block to blank centre offsets. +IN means blank centre is towards nasal relative to the surface block. –IN means blank centre is towards temporal relative to the surface block. +UP means blank centre is above the surface block. –UP means blank centre is below the surface block.
SBEV	numeric;	Safety bevel depth to cut (mm).
SBFCIN, SBFCUP	± numeric; ± numeric;	Surface block to frame centre offsets, i.e. indicates the position of the centre of the frame SHAPE relative to the surface blocking position, so that the SHAPE may be shown in its final relative position for determining cut-out. +IN means frame centre is towards nasal relative to the surface block. –IN means frame centre is towards temporal relative to the surface block. +UP means frame centre is above surface the block. –UP means frame centre is below surface the block.
SBOCIN, SBOCUP	± numeric; ± numeric;	Surface block to optical centre offsets. Instructs the generator to grind the optical centre at a position relative to the surface block. +IN means optical centre is towards nasal relative to the surface block. –IN means optical centre is towards temporal relative to the surface block. +UP means optical centre is above the surface block. –UP means optical centre is below the surface block.
SBSGIN, SBSGUP	± numeric; ± numeric;	Surface block to segment offsets. Position of the segment relative to the centre of the surface block. +IN means segment is towards nasal relative to the surface block. –IN means segment is towards temporal relative to the surface block. +UP means segment is above the surface block. –UP means segment is below the surface block.
SDEPTH	numeric;	Segment depth (see Figure 2 of ISO 13666:1998)
SEGHT	numeric;	Vertical SEG height, measured from the (frame) lower boxed tangent.
SGERIN, SGERUP	± numeric; ± numeric;	Layout reference point to engraving reference point offsets. +IN means engraving reference point is towards nasal relative to the layout reference point. –IN means engraving reference point is towards temporal relative to the layout reference point. +UP means engraving reference point is above the layout reference point. –UP means engraving reference point is below the layout reference point.
SGOCIN SGOCUP	± numeric; ± numeric;	Segment to optical centre offsets. Can be used to place the O.C. for multifocals. +IN means O.C. is towards nasal relative to the segment. –IN means O.C. is towards temporal relative to the segment. +UP means O.C. is above the segment. –UP means O.C. is below the segment.
SLBP	numeric;	Slab off prism (dioptres).

Table A.1 (*continued*)

Record label	Data type	Description
SLHT	numeric;	Slab off line height (measured from frame lower edge).
SLDRP	numeric	Slab off drop. Similar to SLHT, but can be used for uncuts (mm).
SPEED	integer;	Integer number agreed to by the device and host to access speed control tables on both. The value zero is reserved to mean undefined.
SPH	± numeric;	Rx Sphere power (dioptres).
SVAL	numeric;	Value indicating the thickness of a lens after generating measured relative to the machine's reception chuck on the centreline of the chuck.
SWIDTH	numeric;	Segment width (mm).
THKA	numeric;	Thick point angle – the angle at which the THKP occurs (degrees).
THKCMP	± numeric;	General purpose thickness compensation (mm). This compensation is applied to GTHK.
THKP	numeric;	Thickest finished edge thickness (mm).
THNA	numeric;	Thin point angle – the angle at which the THNP occurs (degrees).
THNP	numeric;	Thinnest finished edge thickness (mm).
TIND	numeric;	Index of refraction used for all dioptre curves in a generator packet.
TINT	text[;]	Tint abbreviation.
TNORM	integer	? – state of radius data with regard to *z* axis is unknown. 0 – tracing is normalized for tilt (vertical positions of points indicated by *z* data or FCRV). 1 – tracing is tilted laterally as specified in ZTILT. 2 – tracing is “raw”; vertical positions indicated by *z* data. 3 – tracing is flattened to 2-dimensional, planar form, devoid of tilt or curve. FCRV or *z* data, and ZTILT, specify original measurements.
TOLADD	integer;	Tolerance of add power. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLASPEC	integer;	Tolerance of aspect (cosmetics/appearance). 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLAX	integer;	Tolerance of axis. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLCTHK	integer;	Tolerance of centre thickness. 0 – Failed inspection 1 – Passed inspection 9 – Not tested

Table A.1 (*continued*)

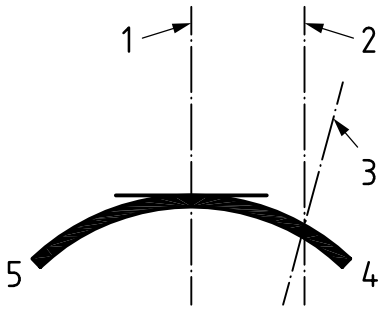
Record label	Data type	Description
TOLCYL	integer;	Tolerance of cylinder power. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLPRVM	integer;	Tolerance of prism magnitude. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLPRVA	integer;	Tolerance of prism base setting. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLSGIN	integer;	Tolerance of segment in horizontal direction. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLSGUP	integer;	Tolerance of segment in vertical direction. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLSHAPE	integer;	Tolerance of shape cut-out. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TOLSPH	integer;	Tolerance of sphere power. 0 – Failed inspection 1 – Passed inspection 9 – Not tested
TPSIZ	integer	Side/temple length of frame (mm).
TPTYP	text	Side/temple type of frame such as cable or straight.
UNI	integer	Patient vision. A value of 1 means the patient is monocular or only has sight in a single eye. A value of 0 indicates binocular vision.
VBOX	numeric[;]	Vertical boxed lens size of frame (mm).
Z	integer;integer;...	Sag data (trace z dimension), see prior definition.
ZA	integer;integer;…	Angle data for sag data.
ZTILT	numeric[;]	Side-to-side tilt of frame as traced.



Key

1	y-start	4	y-end
2	x-start	5	tool diameter
3	x-end	6	specified hole diameter

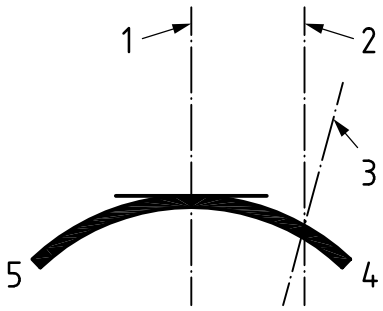
Figure A.1 — Drill information



Key

1	hole reference axis (shown normal to front surface)	4	right lens, temple side
2	hole centreline, no angle specified	5	right lens, nasal side
3	hole centreline, −15° lateral		

Figure A.2 — Lateral drill angle



Key

1	hole reference axis (shown normal to front surface)	4	bottom of lens
2	hole centreline, no angle specified	5	top of lens
3	hole centreline, −15° vertical angle		

Figure A.3 — Vertical drill angle

A.2 Interface records

Table A.2 contains all of the interface records defined for use by systems conforming to this International Standard, arranged alphabetically. The data format is the same as in Table A.1.

Table A.2 — Interface records

Record label	Data type	Description
ANS	literal or integer	Answer type, see record REQ. This is mandatory in all except initial request packets in order to echo the request type specified in the request packet.
CRC	integer	Cyclical-redundancy check. See Annex C for a description of how this is calculated.
D	literal;literal;…	Record label list for auto-initialization. This is used in conjunction with the DEF and ENDDEF records to define the set of records to be associated with an auto-format request. The record has the form D=label;label;label . . .<CR/LF> and can contain one or more record labels. Multiple D records may occur between the DEF and ENDDEF records. This is used to identify to the host such records as are necessary for the device to operate properly.
DEF	limited;integer	Request definition. This is used in the initialization data packet transmitted from the device to the host to indicate the beginning of a list of records that should be assigned a request ID by the host. An ENDDEF record terminates the list. In a device's transmission, the record has the form: DEF = *Device tag*<CR/LF> where <*Device tag*> is a limited string used by the device to identify the request type. In the host's initialization data packet, it has the form DEF = *Device tag* ; request ID <CR/LF> where request ID is an integer that the DEVICE shall use when it wants to make the kind of request it defined for device tag. The device tag may be blank if preset initialization is being performed. The HOST sends a single DEF record per request definition transmitted by the DEVICE. In a device's initialization data packet, the “D” (record label list) record(s) shall immediately follow the request definition record.
DEV	literal	Device type. This record contains one of the literal data choices enumerated in Table A.4, which identifies the kind of device and the corresponding set of device records which should be conveyed. It can also be used in auto-format initialization to indicate the direction of data flow. This appears in initialization packets.
DO	R|L|B|N	This record identifies which lens in a pair is to be processed regardless of presence of other kinds of right/left data. It is mandatory in data packets from the host. R)ight, L)eft, B)oth or N)one: Indicates which eye the device should process.
ENDDEF	limited	End of request definition. This is used to mark the end of a request definition during auto format initialization begun by a request definition (DEF) record. The data field shall contain the matching device tag to the DEF record.
HID	limited	Host identification. The host ID is limited data, which may optionally appear in a request packet. It is echoed in any packets sent by a host to a device during a session begun with a request packet including such a record.
INFO	integer	Information packet control. This is used by hosts to control the transmission of INF requests. When a device indicates INFO=1 in a request or initialization packet, a host can suppress INF requests by specifying INFO = 0 in its data packet.
JOB	limited	Job number. Also called job ID. The JOB ID comprises limited data which is mandatory in all packets outside of initialization. A device shall verify that job IDs in packets received match the job ID originally specified in its request packet; likewise, hosts should take care to return the job IDs specified in request packets in the same format as was received. Job IDs are case sensitive when they include letters. Upload devices that cannot provide job identification shall transmit a field value consisting of the unknown data indicator (“?”).

Table A.2 (*continued*)

Record label	Data type	Description
MESG	text	This is text data conveyed in either direction for any purpose. A device can inform a host of the maximum message length that it can handle by means of the message length (MSL) record.
MID	limited	Machine identification (ID). This comprises limited data that shall uniquely identify a device among all those connected to a particular host, at least within a given set of device types and machine models. It is recommended that devices allow the MACHINE ID to be set in the field. It is optional in initialization or request packets.
MNAME	text	Machine name. This is optional TEXT data that can be used by HOSTS to identify DEVICES using names that will be meaningful to users, such as “ACME Lens Generator Model 101”.
MODEL	limited	Machine model. This comprises literal data that are provided by a machine's manufacturer and incorporated into this International Standard. It is useful in initialization to allow hosts to determine what parameters would be appropriate to send to a machine. It is optional in initialization or request packets to allow a host to refine its handling of preset requests.
MSL	integer	Initialization record to set the maximum message length. This is to inform the host that the device can only display messages of a certain length.
OMAV	MM.mm	Interface version. This is used by hosts to match the structure of its packets to the interface versions of devices. It is optional in request or initialization packets. The structure of the interface version will be “**MM.mm**” where MM is the major version identifier and mm is the minor version identifier. A change to the major identifier shall occur when changes to the interface occur which could affect the ability of hosts and devices to communicate. Changes to the minor version identifier shall occur when additional records are defined and when other changes occur that should not affect the ability of hosts and devices to communicate. The current value for this field is 3.03.
OPERID	limited	Operator identification. This can be used by a host to provide statistical data related to machine operators. These are limited data and may optionally appear in request or initialization packets.
REM	text	Remarks. These are text data that can appear in any packet. Neither hosts nor devices take any action based on remarks. Remarks are *not* echoed back to the sender in any ensuing packets during a session.
REQ	literal or integer	Request type. This is a mandatory record in the request packet used to identify to the host the kind of request being made. Request type contains a value consisting of one of the literal data choices enumerated in Table A.3 or a request ID number returned from a host during initialization. REQ shall be the first record in a request packet.
SN	text	Device serial number. This is text data that shall contain a device's serial number, i.e. an identifier given to the device by its manufacturer. It is expected that this identifier would be useful to the manufacturer for service and/or diagnostic purposes. Its uniqueness is not guaranteed. It is optional in initialization or request packets.
STATUS	Integer[;text]	Status code (described in A.4).
TIMEOUT	integer;integer; integer	This record has the form TIMEOUT = confirmation; Packet; intercharacter. It may be sent from a host to a device during initialization to override the pre-defined timeout values. If a device does not allow such modification, the fact that it does not shall be clearly stated in the device's documentation. A device which does not support altering timeout values shall not return an error status if it receives a TIMEOUT record. Timeout values are in seconds and shall be between 2 s and 255 s.

Table A.2 (*continued*)

Record label	Data type	Description
TRCFMT	integer;integer; U|E;R|L|B; F|P|D	Trace data format. TRCFMT=#;###; E|U; R|L|B;F|P|D a) The first field is the format identifier: 1. Basic ASCII radii format 2. BINARY absolute radii format 3. BINARY differential format 4. PACKED BINARY format 5...100 are reserved for future standard formats. b) The second field is the number of radii in which the tracing shall be expressed; c) The third field is the radius mode identifier: "E" indicates that radii are evenly spaced ("equiangular"); "U" indicates that radii are unevenly spaced, so that an angle data "A" record shall follow the "R" record. d) In initialization or request packets, the fourth field indicates which eye(s) is (are) included: R)ight, L)eft or B)oth. In data packets, it specifies the orientation of the tracing and which eye or eyes were traced. e) The fifth field indicates what has been traced: F)rame, P)attern or D)emo lens. The fifth field is not present when the TRCFMT record is sent during initialization. All five fields shall be sent on subsequent upload and download sessions unless the first field is 0. In this case, only the first field is sent.
VEN	limited	Vendor identification (ID). This contains literal data which is used to identify a device's manufacturer. Vendor IDs will be three character identifiers selected by vendors and incorporated in this International Standard. If the vendor ID is not present in a device's initialization data packet, the host may not be able to transmit set-up parameters to the device.
ZFMT	Integer;integer; U|E;R|L|B; F|P|D	z dimension format for trace data. Identical to TRCFMT.

Table A.3 — Request type (literal data)

Request type	Desired action
INI	Initialization
TRC	Frame trace upload
PTG	Pattern generator download
EDG	Edger download
SBK	Surface blocker download
FBK	Finish blocker download
AGN	Laminator download
GEN	Surface generator download
UPL	Generic upload
DNL	Generic download

Table A.3 (*continued*)

Request type	Desired action
MNT	Maintenance upload
ERR	Error answer response
INF	Information upload
Request ID (integer)	Action as determined by preset or auto-format initialization
COA	Surface coater download
FSG	Front surface generator download
LMD	Lens measuring device download
INS	Inspection device upload
LAP	Lap feeder download

Table A.4 — Device types

Device	Device type (literal data)
Frame tracer	TRC
Pattern generator	PTG
Lens edger	EDG
Surface blocker	SBK
Finish blocker	FBK
Surface generator	GEN
Alternate generating device	AGN
Generic upload device	UPL
Generic download device	DNL
Surface coater	COA
Front surface generator	FSG
Lens measuring device	LMD
File	FIL
Inspection device	INS
Lap feeder	LAP

A.3 Preset records

Tables A.5 to A.18 specify preset packets.

The following formats are used for all the preset definitions.

*LABEL Mandatory, shall always be sent. The asterisk IS NOT SENT in the actual transmission. All labels are mandatory using versions after ISO 16284:2001(E), but the list of previously mandatory labels may be necessary for backwards compatibility.

NOTE For tracing data, the actual tracing records sent depend on the format used.

$LABEL_{3.02}$ Should be sent if the device OMAV is 3.02 or higher.

$LABEL_{3.03}$ Should be sent if the device OMAV is 3.03 or higher.

Descriptions of the record labels used in Tables A.5 to A.11 and Tables A.14 to A.18 are given in Table A.1. Descriptions of the record labels used in Tables A.12 and A.13 are given in Table A.2.

Table A.5 — Preset frame tracer packet

Record label
Trace format should be negotiated as part of initialization before requesting this packet type.
BSIZ
*CIRC
CSIZ
*DBL
FCRV
FTYP
HBOX
$TNORM_{3.02}$
VBOX
ZTILT

Table A.6 — Preset pattern generator packet

Record label
Trace format should be negotiated as part of initialization before requesting this packet type.
$TNORM_{3.02}$

Table A.7 — Preset lens edger packet

Record label
Trace format should be negotiated as part of initialization before requesting this packet type.
BEVM and BEVC shall also be included in the response if they are necessary.
BEVP
BSIZ
*CIRC
$CLAMP_{3.02}$
CSIZ
*DBL
*DIA
$DRILL_{3.02}$
$EPRESS_{3.02}$
$ERDRIN_{3.03}$, $ERDRUP_{3.03}$
$ERNRIN_{3.03}$, $ERNRUP_{3.03}$
*ETYP
*FBFCIN, *FBFCUP
FBSGIN, FBSGUP
FCRV
$FPINB_{3.02}$
*FTYP
$GDEPTH_{3.02}$
$GWIDTH_{3.02}$
*IPD
*LMATTYPE
LMATID
*LTYP
$LTYPE_{3.03}$
$MCIRC_{3.02}$
*NPD
*OCHT
*PINB
*POLISH
*SEGHT
$SGERIN_{3.03}$, $SGERUP_{3.03}$
$TNORM_{3.02}$
ZTILT

Table A.8 — Preset finish blocker packet

Record label
Trace format should be negotiated as part of initialization before requesting this packet type.
*AX
CYL
*DBL
*DIA
$ERDRIN_{3.03}$, $ERDRUP_{3.03}$
$ERNRIN_{3.03}$, $ERNRUP_{3.03}$
*FBFCIN, *FBFCUP
*FBOCIN, *FBOCUP
FBSGIN, FBSGUP
$FCOCIN_{3.02}$, $FCOCUP_{3.02}$
$FCSGIN_{3.02}$, $FCSGUP_{3.02}$
*HBOX
*IPD
*LTYP
$LTYPE_{3.03}$
*NPD
*OCHT
$PRVA_{3.02}$
$PRVM_{3.02}$
SDEPTH
SEGHT
$SGERIN_{3.03}$, $SGERUP_{3.03}$
SGOCIN, SGOCUP
SPH
SWIDTH
*VBOX

Table A.9 — Preset surface blocker packet

Record Label
*BACK
BCSGIN, BCSGUP
*BCTHK
*BETHK
*BPRVA
*BPRVM
*DIA
*FRNT
*GAX
*GPRVA
*GPRVM
*IFRNT
*KPRVA
*KPRVM
*LTYP
$LTYPE_{3.03}$
$OPC_{3.02}$
*SAGRD
*SBBCIN, *SBBCUP
*SBFCIN, *SBFCUP
SBOCIN, SBOCUP
SBSGIN, SBSGUP
SDEPTH
SWIDTH

Table A.10 — Preset surface generator packet

Record label
*AVAL
*BACK
*BCTHK
*BETHK
*BLKB
*BLKCMP
*BLKD
*BLKTYP
$CPRVA_{3.02}$
$CPRVM_{3.02}$
*CRIB
*DIA
EECMP
*ELLH
FINCMP
*FLATA
*FLATB
*FRNT
*FTTHK
*GAX
*GBASE
*GBASEX
*GCROS
*GCROSX
*GPRVA
*GPRVM
*GTHK
*IFRNT
*KPRVA
*KPRVM
*LAPBAS
*LAPBASX
*LAPCRS
*LAPCRSX
LAPM
*LAPPRB
*LENPRB
*LIND
*LMATTYPE

Table A.10 (*continued*)

Record label
*LMATID
*LSIZ
*LTYP
$LTYPE_{3.03}$
*OTHK
PADTHK
*PIND
$PREEDGE_{3.02}$
*RNGD
*RNGH
RNGCMP
RPRVA
RPRVM
*SAGBD
*SAGCD
*SAGRD
*SBBCIN, *SBBCUP
*SBEV
SBOCIN, SBOCUP
*SPEED
*SVAL
THKCMP
*TIND

Table A.11 — Preset alternate generating device (lamination system) packet

Record label
*AX
*ADD
*CYL
*FCOAT
*LIND
*LNAM
*LSIZ
*LTYP
$LTYPE_{3.03}$
*OPCB
*OPCF
*RXNM
*SPH

Table A.12 — Maintenance information packet

Record label
MID
MODEL
OPERID
SN
VEN

Table A.13 — Process status packet

Record label
*JOB
*STATUS

Table A.14 — Preset lens coater packet

Record label
CPID
CRIB
DIA
FRNT
GBASEX
GPRVM
LMATTYPE
LTYP
$LTYPE_{3.03}$
TIND
TINT

Table A.15 — Preset front surface generator packet

Record label
BVD
LNAM
NOD
PANTO
SGOCIN
UNI
ZTILT

Table A.16 — Preset lens measuring device packet

Record label
Trace format should be negotiated as part of initialization before requesting this packet type.
ADD
ADD2
AX
BACK
BCOCIN, BCOCUP
BCSGIN,BCSGUP
CTHICK
CYL
DBL
DIA
FBFCIN, FBFCUP
FBOCIN, FBOCUP
FBSGIN, FBSGUP
FRNT
HBOX
IPD
LIND
LTYP
$LTYPE_{3.03}$
NPD
OCHT
PRVA
PRVM
SEGHT
SGOCIN, SGOCUP
SPH
SWIDTH
VBOX

Table A.17 — Preset inspection device packet

Record label
INSADD
INSAX
INSCTHK
INSCYL
INSPRVA
INSPRVM
INSSGIN
INSSGUP
INSSPH
TOLADD
TOLASPEC
TOLAX
TOLCTHK
TOLCYL
TOLPRVA
TOLPRVM
TOLSGIN
TOLSGUP
TOLSHAPE
TOLSPH

Table A.18 — Preset lap feeder device packet

Record label
LAPBAS
LAPBASX
LAPBIN
LAPCRS
LAPCRSX
LAPM
TIND

A.4 Status codes

Status codes are numbers defined in Table A.19 which are transmitted in response packets to indicate the presence or absence of error conditions. This record has the form STATUS = <code>;<description>. The second field in the record is text, in which the error can be expressed literally. The record is mandatory in response packets. The presence of a non-zero status code in a response to a request will typically cause the current session to abort. The exception is during initialization, where the status code is used to indicate what form of initialization is supported. Status code modifiers, defined in Table A.20, may be added to the status codes to provide more detailed information about the error condition. The current set of modifiers is related to a specific Code, 17. When a host returns a description in the status code record, the device should display it if possible.

Table A.19 — Status codes

Status code	Description
0	No error
1	Job not found
2	Cannot store data; job is protected.
3	General, defined by host or device; the description field describes the error.
4	Cannot process job (data are available in host system, but are not appropriate for the device making the request).
5	Need initialization. This would normally be sent by a host when it has been re-started after an auto-format or preset initialization session has taken place.
6	Invalid job ID.
7	Missing record. The record label for the missing record shall appear in the description field.
8	Host error. Some internal error prevents the host from meeting the request. The error shall be described in the description field.
9	Incompatible device/host.
10	A value in a record is out of range. The offending record label shall appear in the description field.
11	Receiver is busy, temporarily unable to fulfil the request.
12	Out of sync. When a host or device receives an out-of-sync packet, it shall issue this status code and both sides shall abort the session. The receiver of such a packet sends confirmation to the sender.
13	Invalid initialization. Something is wrong with the initialization packet.
14	Auto-format not supported. This is sent by a host which cannot perform auto-format initialization but which can perform preset initialization. This lets the device know that it should proceed with preset format initialization.
15	Initialization not supported. This is sent when a host cannot perform any type of initialization.
16	Invalid request. This is sent when a host cannot identify the type of request the device is making.
17	Unsupported tracing format. This is sent when a device attempts to use a tracing format that is not supported by the host. The status code modifiers (see Table A.20) may be added to this error code to help indicate the exact nature of the error.
18	Format error. This is sent when a device or host receives a packet that contains data that cannot be parsed in accordance with this International Standard.
19	Process failed. This is used only in INF requests, sent from devices to hosts to indicate that the process associated with the most recent request for the job indicated has been completed. This status indicates that the process did not finish successfully.

Table A.20 — Modifiers for status code 17

Modifier	Description
256	None of the proposed tracing formats is acceptable.
512	None of the proposed number of points is acceptable.
1024	None of the proposed radius modes is acceptable.

NOTE The status code modifiers are evaluated as bits in the high byte of a 16-bit word, so that they can be combined without information loss. The 255 possible values in the low byte can represent general error conditions, and the 8 bits in the high byte can be used to refine this data. The current set of modifiers relate only to status code 17. The values could have other meanings in conjunction with other codes.

Annex B
(informative)

Packed binary format example

The following code is a complete implementation of packing and unpacking data to and from the format described in 5.5.5. To pack data, call ***packit(src, dst, nradii)*** where “src” is a pointer to the array of integers to pack, “dst” is a pointer to a buffer into which the packed data will be written, and “nradii” is the number of elements in the data set. To unpack data, call ***unpackit(dst, src, nradii)*** where “src” is a pointer to packed data, “dst” is a pointer to an array of integers, into which the unpacked data will be written, and “nradii” is the number of elements in the data set.

The *pack()* and *unpack()* functions are used internally.

NOTE In the following code segments, integers are defined to be 16-bit quantities.

The following special requirements should be observed:

When packing, one byte value and the nibble counter is not on an even byte boundary, the nibble order output is:

— HIGH NIBBLE of byte goes into the LOW NIBBLE of output byte N;

— LOW NIBBLE of byte goes into the HIGH NIBBLE of output byte N+1.

This makes the byte read correctly when looking at a byte dump.

When packing two byte values and the nibble counter is not on an even byte boundary, the nibble order output is:

— Original data: 0×1234;

— 80×86 storage order is: 34 12. Nibble 3 is output first, then nibbles 4, 1 and 2.

If the nibble counter is on an even boundary, the same nibble order is used. This corresponds to the normal storage order of the 80×86 processor.

If, at the end of the process, an odd number of nibbles results, a 0 nibble is added to the end of the data stream.

```
/* Global variables */
static unsigned char *outftp;
static int ftpn=0;

/*************************************************************************
pack() function

INPUTS:
  i - The value to be packed into the output stream.
  n - The number of nibbles it should take up.
OUTPUTS:
  Deposits the packed data into the output buffer pointed to by outftp and updates
  outftp as needed.
GLOBALS:
  None
STATICS:
  outftp, ftpn
```

```
Packs nibbles into *outftp and increments the nibble counter ftpn by the number of nibbles
packed.

pack()is private to this module, called by packit()

*****************************************************************************/

void pack(i, n)
 unsigned int i;
 int n;

{
#ifdef NOT_80x86
 if (n == 4)
   swab(outftp, outftp, 2);       /* Put into 80x86 order */
#endif
if ( !(ftpn & 1) )            /* on an even nibble boundary? */
{
 if ( n > 1 )
 {
  *outftp++ = (i & 0xff);
   if ( n > 2 )
     *outftp++ = (i & 0xff00)>>8;
 }
 else
  *outftp = i<<4;               /* pack high nibble first */
}
else
{
 if ( n > 1 )
 {
  if ( n == 2 )
  {
     *outftp++ |= (i & 0xf0)>>4;     /* fill in low nibble with high nibble of next byte*/
     *outftp = (i&0x0f)<<4;          /* fill in high nibble with low nibble */
  }
  else
  {
     *outftp++ |= (i & 0x00f0)>>4;              /* fill in nibble 3 */
     *outftp++ = (i & 0x000f)<<4 | (i & 0xf000)>>12;   /* nibbles 4 and 1    */
     *outftp = (i & 0x0f00)>>4;                 /* and nibble 2    */
   }
 }
 else
  *outftp++ |= (i & 0x0f);                  /* fill in low nibble */
}
ftpn+=n;
}

/*****************************************************************************
packit() function

INPUTS:
 src       Pointer to integers to pack.
 dst       Buffer to contain packed data.
 num       Number of integers to pack.
OUTPUTS:
 Deposits packed data in output buffer (dst), and returns number of bytes actually
packed.
GLOBALS:   None
STATICS:   outftp, ftpn

Packs the integer data into the output buffer.
Calls pack()
*****************************************************************************/

int   packit(src, dst, num)
```

```
 int *src;
 unsigned char *dst;
 int num;
{
 int    i;
 int    state=16;
 int    dr, d2r, dr1=0;

 outftp = dst;
 for (ftpn = 0, i = 0; i < num; i++)
 {
  if (!i)
     dr = src[i];
  else
     dr = src[i]-src[i-1];
  d2r = dr-dr1;
  switch(state)
  {
     case 16:
        if (dr<128 && dr>-127)
        {
           state = 8;
           pack(0x8000, 4);
           pack(dr, 2);
        }
        else
        {
           pack(src[i], 4);
        }
        break;
     case 8:
        if (dr>=128 || dr<=-127)
        {
           state = 16;
           pack(0x81, 2);
           pack(src[i], 4);
        }
        else
           if (d2r<8 && d2r>-8)
           {
            state = 4;
            pack(0x80, 2);
            pack(d2r, 1);
            }
           else
           {
            pack(dr, 2);
            }
           break;
     case 4:
        if (d2r>=8 || d2r<=-8)
        {
            pack(0x8, 1);
            if (dr>=128 || dr<=-127)
            {
                state = 16;
                pack(0x81, 2);
                pack(src[i], 4);
            }
            else
            {
                state = 8;
                pack(dr, 2);
            }
            }
           else
           {
            pack(d2r, 1);
```

```
            }
            break;
      default:
         return(-99);
         break;
   }
   dr1 = dr;
  }
  return( (ftpn+1)>>1 );
}


/*************************************************************************
unpack() function

INPUTS:
 i - The number of nibbles to unpack from the packed data stream.
OUTPUTS:
 The 16-bit integer value of the packed data
GLOBALS:
 None
STATICS:
 outftp

Returns the next nibble, byte, or word of packed data based on the size which is passed.
Since byte order for words is different on Z8002, it swaps the order of the bytes before
returning a word value.

unpack() is local to this module, called by unpackit()
*************************************************************************/

int  unpack(i)
 int i;
{
 unsigned int  j;

 if ( !(ftpn & 1) )              /* on an even boundary */
 {
  switch (i)
  {
     case 1:
        j = (*outftp & 0xf0)>>4;     /* pull from high nibble first */
        if ( j > 7 )
           j |= 0xfff0;
        break;
     case 2:
        j = *outftp++;               /* pull a whole byte */
        if ( j > 127 )
           j |= 0xff00;
        break;
     case 4:
        j = *outftp++;           /* pull first low byte */
        j |= ((int)(*outftp++))<<8;  /* or in high byte */
        break;
  }
 }
 else                         /* starting in the middle of the byte */
 {
  j = ((*outftp++) & 0x0f);      /* pull from low nibble first */
  switch (i)
  {
     case 1:
        if ( j > 7 )
           j |= 0xfff0;           /* extend sign */
        break;
     case 2:
        j <<= 4;                     /* shift up a nibble */
        j |= (*outftp & 0xf0)>>4;    /* and grab low nibble from high nibble */
```

```
        if ( j > 127 )
           j |= 0xff00;           /* extend sign */
        break;
      case 4:
        j <<= 4;              /* shift what we have up to be nibble 3 */
        j |= (*outftp & 0xf0)>>4;    /* add in nibble 4 to get low byte complete */
        j |= ((int)(*outftp++ & 0x0f))<<12;     /* add in nibble 1 */
        j |= (*outftp & 0xf0)<<4;        /* add in nibble 2 */
        break;
    }
  }
  ftpn += i;
#ifdef NOT_80x86
  if ( i == 4 )
    swab( (char *) j, (char *) j, 2);         /* put into Z8002 order */
#endif
  return( (int) j );
}

/*****************************************************************************
unpackit() function

INPUTS:
  src    Pointer to an input buffer containing packed data.
  dst    Pointer to an output buffer to fill.
  n      Number of integers to unpack
OUTPUTS:
  Unpacked data in the output buffer (dst). Returns number of unpacked bytes.
GLOBALS:
  None
STATICS:
  outftp

Unpacks the packed data from src into dst. Calls unpackit().

*****************************************************************************/

int   unpackit(dst, src, n)
  int *dst;
  unsigned char *src;
  int n;
{
  int state=16, size=4;
  int i, dr=0, d2r=0, dr1=0, x;

  outftp = src;
  ftpn = 0;
  for (i = 0; i < n; i++)
  {
    AGAIN:
    x=unpack(size);
    switch(state)
    {
      case 16:
        if ( x == 0x8000 )
        {
          state = 8;
          size = 2;
          goto AGAIN;
        }
        dst[i]=x;
        if (i)
        dr1 = x-dst[i-1];
        else
          dr1 = x;
        break;
      case 8:
        if ( (x & 0xff) == 0x80 )
```

```
        {
           state = 4;
           size = 1;
           goto AGAIN;
        }
        if ( (x & 0xff) == 0x81 )
        {
           state = 16;
           size = 4;
           goto AGAIN;
        }
        dr = x;
        if (i)
           dst[i] = dst[i-1]+dr;
        else
           dst[i] = dr;
        dr1 = dr;
        break;
     case 4:
        if ( (x & 0x0f) == 0x8 )
        {
           state = 8;
           size = 2;
           goto AGAIN;
        }
        d2r = x;
        dr = dr1+d2r;
        dst[i] = dst[i-1]+dr;
        dr1 = dr;
        break;
   }
  }
return(  (ftpn+1)>>1 );
}
```

Annex C
(informative)

CRC calculation

The following is a C subroutine in which the algorithm for the CRC calculation is demonstrated.

```
/**********************************************************************

        CRC16 - 16 bit CRC
        CCITT CRC-16 Cyclical Redundancy Check
        polynomial: X^16 + X^12 + X^5 + 1
        (used in XMODEM-CRC communications protocol)

**********************************************************************/

union _crc {
 unsigned char b[2];      /* high byte is b[1], low byte is b[0] */
 unsigned w;              /* word value */
 };

unsigned crc16(len, start_crc, p)
int len;           /* length of p */
unsigned start_crc;           /* starting value, initialize to zero */
unsigned char *p;         /* pointer to memory of which to calculate crc */
{
 union _crc crc;
 int i;
 crc.w = start_crc;           /* set up starting value of CRC */

 while (len-- > 0)
 {
  crc.b[1] ^= *p++;       /* xor value of next byte into HIGH byte of CRC */
                              /* this is for an 80x86 processor */
  for(i=0;i<8;++i)
     if (crc.w & 0x8000)      /* high bit set?? */
     {
        crc.w <<= 1;       /* left shift one */
        crc.w  ^= 0x01021;    /* XOR value 0x1021 */
     }
     else
     {
        crc.w <<= 1;       /* left shift one */
     }
 }
 return (crc.w);
}
```

The following code fragment will print "0cd3" (hexadecimal) as the CRC value to "Hello World!":

```
 char hello[] = „Hello World!„;
 unsigned crc;
 crc = crc16(strlen(hello), 0, hello);
 printf(„%04x\n„,crc);
```

ICS 11.040.70

Price based on 70 pages