Document: EPOSEJ(M310)Protocol Specification
Author: Nikos Maroudas, MICRELEC SA
Publisher: MICRELEC SA
Release: 1.3

Status: Final release

EPOSEJ(M310) Protocol Specification

Table of contents

1. Purpose of this document

The purpose of this document is to provide the necessary specification to software designers interested in communicating with fiscal ECR/POS models. This document assumes that the reader is familiar with basic communication concepts, such as transmittances, receptions, timeouts, etc. Also assumes that the reader is familiar with fiscal POS/ECR functioning and procedures.

2. Goals

The developer will have all necessary information for implementing all protocol layers, thus be able to:

- Keep track of all transaction operations (sales, voids, refunds etc)
- Expand the available local database of items to arbitrary numbers
- Perform the ECR/POS configuration (setup) remotely
- Issue receipts and all reports via protocol commands

3. Design approach and compatibility issues

Developers should take into consideration future additions or expansions to this specification. The goal is that an application designed using an older revision specs will function correctly in newer revision protocol. In order to do so, the developers *must* check responses only for the presence of the known information and 'quietly' discard the information that is unknown. The designers of this protocol guarantee that the extensions of this protocol will not alter the position or the type of the information (unless absolutely unavoidable). Extra fields will always be added to the right of the reply strings. Specifically, these are the rules that deliver the highest compatibility:

a) Check the protocol version number. This information guarantees safety towards new commands. For example (hypothetically):
In protocol revision '01.02' and higher the command '#' is supported, so reading a revision '01.00' indicates that the command '#' will fail.

b) Always assume correct a reply that has more fields than expected.
For example:
Reply expected: "/1/AAAAA/BBBB/CCCC/"
Reply received: "/1/AAAAA/BBBB/CCCC/DDDDD"
(Field 'DDDDD' is unexpected, but should not generate an error because all the expected fields are present. So this field *should* be silently discarded.)

c) Always assume correct a 'FLAGS' field that is longer than expected.
For example:
Reply expected: "/1001001001/"
Reply received: "/1001001001001/"
(Three extra bits in the 'FLAGS' field are unexpected. The application must discard them without generating errors).

d) It is an excellent design approach not to be very strict with numerical ranges or string lengths expected. This guarantees that the application will be compatible with other ECR/POS devices that use this protocol, but having different resources to operate with. For example, an ECR/POS having more memory is probable to support a wider local item

base, reporting higher index numbers. Or, a different printer mechanism may limit, for example, a header line length. Having a flexible design promises maximum compatibility with different hardware requiring very little (or no) changes to application source code.

3.1. Further information

The implementers are encouraged to study and/or use parts of code examples which are part of this document. Also they must keep informed of any changes in this specification due to the status of this document. Suggestions from developers may or may not influence details of the document until it reaches 'final' status.

4. Communications line

The ECR/POS communicates with host computer via an asynchronous serial line of the RS-232C recommended standard. The serial line parameters are:

- Baud rate: 9600 baud
- Parity: none
- Data: 8
- Stop: 1
- Flow control: none

Note that because there is no flow control, only the RX/TX/GND signals are required for the cable configuration. The maximum length of cable is described in the 232C recommended standard for this baud rate. It is highly recommended that the maximum length is not exceeded to prevent drops in communication rate and undesirable retransmittances due to errors, or in worst case a total communication failure. When cable distance is unavoidably long, an extender may be used.

5. Protocol layers discussion

There are two different needs which the ECR/POS satisfies with two separate protocol layers. The first is the need of keeping track of the POS activity and the extension of the local database of items. The second is the need to use the ECR/POS as a terminal device which we can call 'fiscal printer'. The protocol layers for these needs respectively are:

- The 'online' protocol layer (It will be referred as 'online protocol')
- The 'command' protocol layer (It will be referred as 'command protocol')

Note that there is no such case where both layers are active at the same time due to the nature of the needs each layer deals with. To be more clear, the online protocol is required when it is desired to observe the POS device's activity when the operator of the ECR/POS issues receipts or any other document with it. The command protocol is required when is desired to use the device with a host computer application that issues the receipts and reports to the ECR as a fiscal printer.

Although these two layers cannot coexist at the same time of POS operation,

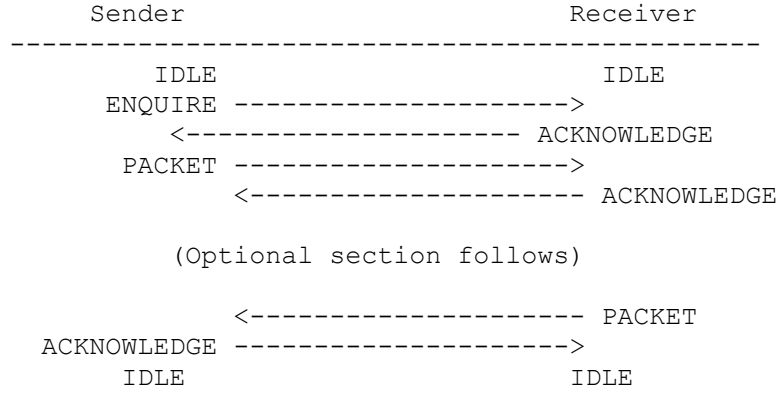
switching between them is allowed anytime. As expected, communication rules and procedures that layers use are the same.

A major difference between the online and command protocol is the origin of the communication. In the online protocol, the communication starts from the POS/ECR in contrast with the command protocol where the communication starts by the host computer.

6. Common rules

6.1. Model of data interchange

Both protocol layers share a common model of interchanging data with the host. The next scheme describes this model:



This scheme although describes the typical flow of data between the two communicating devices (POS and host computer) does not include any other situation such as errors in transmittance, retransmittance etc. Note also that the 'sender' will be the ECR/POS and the 'receiver' will be the host in online protocol. In the command protocol, the 'sender' will be the host and the 'receiver' will be the ECR/POS.

Observe that this model includes two different packet transmittances, one from sender to receiver and one from receiver to sender. In the paragraphs to follow we will call the first packet 'request packet' and the second one 'reply packet' for simplicity. Reply packets are always sent by the ECR/POS when receiving command protocol requests. Also reply packets may be sent in special cases by the host computer at online protocol.

6.2. States of protocol

For a better understanding of the previous paragraph and the communication flow, we can define states which communication 'sides' will enter.

- Idle state
  This is the state before any communication attempt takes place.

- Enquire state
  The sender that wishes to initiate communication sends an inquiry to the receiver. The process of sending this inquiry is the enquire state so only the sender enters this state.

- Acknowledge state
  The receiver will enter this state right after receiving an inquiry or after the verification of a request packet. The sender will enter this state after the verification of a reply packet.

- Verify acknowledge state
  The sender or receiver will enter this state after an enquire state or a packet transmittance state. The process of waiting the other end's positive or negative response is to verify acknowledge state.

- Packet transmittance state
  The sender will enter this state to transmit a request packet and the receiver to transmit a reply packet.

- Packet reception state
  The receiver enters this state after acknowledging the sender's enquire to get the request packet. The sender will enter this state right after verifying a positive acknowledge from the receiver, and only if the specific protocol case requires a reply packet.

Considering the above, the state flow for the sender and the receiver in a typical communication attempt will be:

Sender	Receiver
Idle	Idle
Enquire state / Verify ack. state	Acknowledge state
Packet transmittance state	Packet reception state
Verify acknowledge state	Acknowledge state
Packet reception state	Packet transmittance state
Acknowledge state	Verify acknowledge state
Idle	Idle

6.2.1. States definition -> Enquire state

The enquire state is actually the transmittance of a single ASCII control code ENQ [CC1] by the sender. Doing this, the sender has concluded the enquire state. The purpose of this state is to find out if the receiver is able to reply, without flooding the communication line with too much data. After sending the ENQ code, the sender must wait for a response from the receiver, entering verify acknowledge state (see 6.2.2). It is highly recommended to clear the receiving buffer before entering an enquire state, so discarding any accidental data previously received in the serial communication's receive buffer, especially in cases where serial communication is interrupt driven.

Some synchronization needs may also require that before sending the ENQ code, hosts should send the CAN (cancel) [CC1] control code to cancel any waiting states in the ECR/POS side.

6.2.2. States definition -> Verify acknowledge

The verify acknowledge state is the reception of a response code which indicates that an action from one side has been accepted by the other. For this to work, the ASCII control codes ACK and NAK [CC1] are used to mean positive

or negative acknowledgement respectively. In this state the sender or the receiver enters in the following cases:

- after an enquire state by the sender
- after a request packet transmittance by the sender
- after a reply packet transmittance by the receiver

In any of the above cases, the side which is in the verify acknowledge state must either accept ACK or NAK as valid responses within some specific time window. Any other received control values should be treated as NAK.

On reception of an ACK, the host must leave the verify acknowledge state and proceed to the next state, if any. This means that the previous state was successfully processed by the other side of the communication. On reception of a NAK, the host must leave the verify acknowledge state and repeat once more the previous state. For example, if the verify acknowledge state was for a previous enquire state, the enquire state must be repeated. If the request packet was not acknowledged, the packet must be retransmitted.

To prevent infinite communication loops, each of these cases mentioned are limited to a specific retransmittance count, which, when reached, indicates that the communication attempt causing the retransmittances was unsuccessful and further communication is not possible for some reason. The possible reasons for such a failure may be:

- Disconnection of serial cable
- Host computer or ECR/POS fatal error
- Too noisy communication line

6.2.3. States definition -> Acknowledge state

The acknowledge state is the transmittance of either ACK or NAK control codes after a previous enquire or packet reception. ACK must be transmitted when the enquire is accepted or the packet is verified successfully. This is 'positive acknowledge'. NAK must be transmitted when the enquire must be either delayed or rejected, or if the packet failed checksum verification. This is 'negative acknowledge'. Hosts must not transmit any other codes except ACK, NAK and CAN in this state.

6.2.4. States definition -> Packet transmittance state

This state is the transmittance of either a request or a reply packet by the sender and the receiver respectively. Packets in both cases follow the rules described in a later paragraph [see 6.3]. On completion of the packet transmittance, the sender or receiver advances to the next state, if any. During the packet transmittance state, the sender or receiver may also transmit control codes which will be transparent for the packet data, ie they will not be included in the data section of the packet.

6.2.5. States definition -> Packet reception state

The packet reception state is the process of receiving a request or reply packet. The sender will enter this state when receiving a reply packet and the receiver when receiving a request packet. Packet reception is initiated with the reception of the STX control code [CC1]. Any reception of data before the reception of STX must be silently discarded. Packet reception is terminated with the reception of ETX control code [CC1]. Any data after the termination

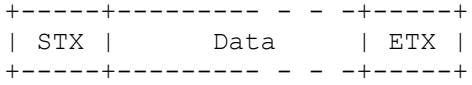
code (ETX) do not belong to this state. See next paragraph for packet handling and structure.

6.3. Packet purpose and structure

The actual communication data in both protocol layers are encapsulated in a 'packet'. As described above, there are request packets and reply packets. In simple words, request packets contain instructions that the sender wishes the receiver to follow or plain information. Reply packets are information which describe how receiver followed the instructions and/or plain information.

Request packets are always sent by the sender. Reply packets are always sent by the receiver. Request and reply packets have the same basic structure in both online and command protocol layers but differ in their contents.

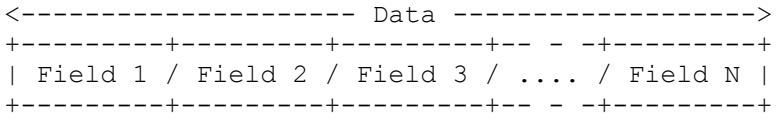
The packet structure is the following:



Notice that the actual data is between STX and ETX fields which are simply the ASCII control codes STX and ETX [CC1]. By ASCII definition, the STX/ETX control codes indicate the start of data transmittance and the end of data transmittance respectively. Any valid octet between the STX and ETX is considered 'data' octet. Valid data octets must be between values '32' and '255' (decimal). Octets lower than '32' are considered 'control' codes [1] and MUST be interpreted specially. Valid data octets are forming the complete data section. Control codes are NOT part of the data and this also applies for the STX/ETX control codes.

The length of the data section is variable, due to it's multifunctioning purpose. ECR/POS is able to accept data up to 250 octets of data in a single packet. Hosts MUST be able to accept at least the same amount of data in a single packet. ECR/POS will discard any further data if this limit is reached producing a negative acknowledge to the host.

Inside the data section of a packet, request or reply, are 'data fields':



Data fields form the total of the data section of a packet. Each field's size may vary. For this reason, a 'special' data character is defined to function as 'field separator'. In both protocol layers, the field separator character is the slash '/' (ASCII character 47 decimal, 057 octal, 2F hexadecimal). ECR/POS interprets this character as 'start of next field'. Host application has to do the same. As a result of this character's special meaning, hosts MUST NOT include this character as part of field data but only as field separator. The reason for this is that the ECR/POS will incorrectly treat it as field separator and count one extra field in the packet, probably also shifting all other fields by one position to the right.

Fields vary in size and content. Various types of fields are described in a later paragraph in detail.

6.3.1. Packet verification - error detection

To ensure that a request or reply packet was received with no errors, both layers use a special field: the checksum. Checksum is always the last field in the packet in all cases of packet transmittances. It also must be separated from the previous field using the slash (/). Checksums are always a 2-digit decimal values and represent the modulo 100 of the 8-bit sum of all data octets in the packet except any control codes or the 2-digits checksum itself but including the field separators. All field separators are calculated in the checksum.

Example checksum calculation function in 'C':

```
+-----------------------------------------------------+
| BYTE CalcChecksum(BYTE *packet)                     |
| {                                                   |
|   BYTE sum = 0;                                     |
|   int  checklength = strlen(packet) - 2;            |
|                                                     |
|   while(checklength--) sum += (BYTE) (*packet++);   |
|   return( (sum % 100) );                            |
| }                                                   |
+-----------------------------------------------------+
```

Example checksum calculation function in pseudo code:

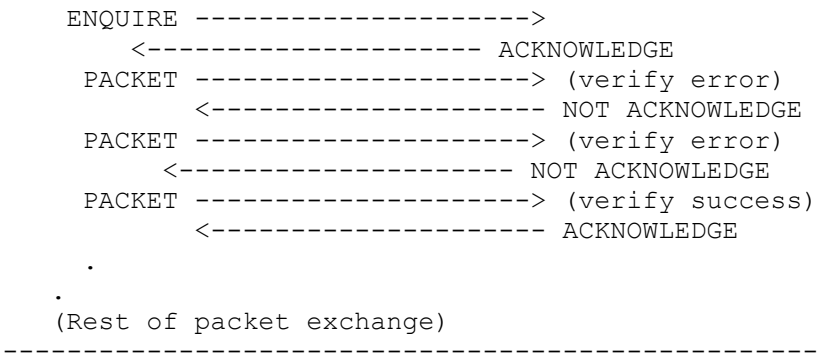
```
+-------------------------------------------------------------------+
| Function Calculate_Checksum( parameter data_packet ) Returns BYTE |
| Begin                                                             |
|   Declare CALCSUM, I as BYTE                                      |
|   CALCSUM = 0                                                     |
|   For I = 0 to stringlength( data_packet ) - 2 Do                 |
|     CALCSUM  =  CALCSUM + ASCII( data_packet[ I ] ) )             |
|   Next  I                                                         |
|   CALCSUM = CALCSUM mod 100                                       |
|   Return CALCSUM                                                  |
| End                                                               |
+-------------------------------------------------------------------+
```

The receiver of the packet must calculate this checksum locally, compare it with the transmitter's checksum and, if found equal, the packet is valid and a positive acknowledgement must be sent. Otherwise the packet was corrupted and a negative acknowledgement must be sent. The checksum will always be a numeric, 2-digit field in range 00-99.

```
<---------------------  Data section ------------------->
<-------------- Layer fields ----------><-- Checksum -->
+---------+---------+---------+---------+---------------+
| Field 1 / Field 2 / Field 3 /  . . .  /       CC      |
+---------+---------+---------+--- - - -+---------------+
```

Remembering the state paragraphs above, negative acknowledgements in packet receptions cause retransmittances of the packet. The scheme that follows describes one such case where the packet failed checksum verification twice and succeeded in the third:

```
     Sender                          Receiver
 ------------------------------------------------
          IDLE                          IDLE
```



6.3.2. Fields - discussion

As already mentioned, fields are the building blocks of a data packet. In this paragraph we will examine all available types of fields and their basic restrictions and requirements.

In both layers, there are only two classes of fields: the string class and the numeric class. Further 'type' labelling was necessary to be defined in order to document each type's ranges and restrictions. Understanding those is essential because when out of 'type' range fields are sent will be rejected by the ECR/POS on further packet processing.

Although fields of certain class and type have a range, the specific packet may REQUIRE a lower range for successful process. Keeping this in mind, applying fields to a packet should be done following this scheme:

- Apply class restrictions checks
- Apply type restrictions and range checks
- Apply packet's specification for fields restrictions and range

6.3.3. Fields - classes

As mentioned, field classes are either string or numeric. These are the attributes of each class.

String class:

- Can contain any character of value 32 to 255 (decimal) except slash ('/')
- Can be of zero to any length that does not exceed the maximum packet size

Numeric class:

- Can contain any numeric character, a decimal point
- Can contain any 'A' to 'F' digit if hexadecimal (*)
- Can contain a minus as a first character
- Can have a total length of zero to 12 characters

(*) Hexadecimal values are only sent at command protocol reply packets for device status map and fiscal status map fields.

6.3.4. Fields - types in detail

Field types are used as a method of generating or recognizing specific or generic fields for a use in a packet. The list that follows defines the ranges and restrictions of the specific types.

- ===== INTEGER type ============================
  Class: Numeric
  Value range: '-999999' to '999999'
  Digit range: 1 to 6 digits
  Notes: Fields of this type must not contain any decimal part or decimal point. This type is usually used as a counter field or an index.

- ====== DATE6 type ============================
  Class: Numeric
  Value range: '010199' to '311240'
  Digit range: When required, must be 6 digits
  When optional, may not be sent at all
  Notes: Specifies a date. Date format is DDMMYY.

- ====== DATE8 type ============================
  Class: Numeric
  Value range: '01011999' to '3112040'
  Digit range: When required, must be 8 digits
  When optional, may not be sent at all
  Notes: Specifies a date. Date format is DDMMYYYY.

- ====== TIME type ============================
  Class: Numeric
  Value range: '000000' to '235959'
  Digit range: When required, must be 6 digits
  When optional, may not be sent at all
  Notes: Specifies a time. Time format is HHMMSS.

- ====== FLAGS type ============================
  Class: Numeric
  Value range: '0' to '1' for each flag in field
  Digit range: When required, must be as long as the packet requires. When optional, may not be sent at all
  Notes: Flags type is used to minimize packet fields where a single "true"/"false" or "yes"/"no" type of information must be passed for various attributes.

- ====== AMOUNT type ============================
  Class: Numeric
  Value range: '-99999999.99' to '99999999.99'
  Digit range: 1 to 12 total
  0 to 8 integer part
  0 to 2 decimal part
  Notes: AMOUNT is usually used to specify prices, discounts, payment values, totals, etc.
  When used to specify payments, this type

```
                      will always be expressed in the active
                      note (ie: drachmas or euro)

- ====== QTY type ==============================
  Class:              Numeric
  Value range:        '-99999.999' to '99999.999'
  Digit range:        1 to 10 total
                      0 to 5 integer part
                      0 to 3 decimal part
  Notes:              QTY is used to specified quantities of
                      any kind.

- ====== RATE type =============================
  Class:              Numeric
  Value range:        '0.000000' to '9999.999999'
  Digit range:        1 to 11 total
                      0 to 4 integer part
                      0 to 6 decimal part
  Notes:              RATE is used to specify currencies of
                      foreign notes or euro to drachmas rate
                      and vice versa

- ====== PERCENT type ==========================
  Class:              Numeric
  Value range:        "0.00" to "100.00"
  Digit range:        1 to 6 total
                      0 to 3 integer part
                      0 to 2 decimal part
  Notes:              PERCENTAGE is used to specify a
                      discount percentage, a markup percentage etc.

- ====== STRING type ===========================
  Class:              String
  Value range:        -
  Character range:    1 to 240 (if not exceeding max packet size)
  Notes:              A normal string
```

7. Online protocol

In this section the online protocol will be explained in detail. The model of communication is the following, initiated by ECR/POS:

```
ECR/POS                          Host computer
=============================================
   IDLE                          IDLE
ENQUIRE ---------------------->
        <---------------------- ACKNOWLEDGE
 PACKET ---------------------->
        <---------------------- ACKNOWLEDGE
   IDLE                          IDLE

or (in case host must send reply packet to ECR)

ECR/POS                          Host computer
=============================================
   IDLE                          IDLE
ENQUIRE ---------------------->
```

```
             <---------------------- ACKNOWLEDGE
      PACKET ---------------------->
             <---------------------- ACKNOWLEDGE
             <---------------------- PACKET
ACKNOWLEDGE ---------------------->
       IDLE                         IDLE
```

7.1. Online protocol packets in detail

Online protocol supports the following packets:

- Item sale packets
- Discount/Markup/Ticket/Coupon packets
- Payment packets
- Closure packets
- Host database: Item search packet
- Host database: Client search packet

Online packets have some leading fields called 'online header' which is common in all cases. So, the request packet becomes:

```
+--------++---------+---------+---------+- - - +---------++----------+
| Header || Field 1 | Field 2 | Field 3 | .... | Field N || Checksum |
+--------++---------+---------+---------+- - - +---------++----------+
```

- Header layout

Header contains 4 fields which are explained later. Notice that the first field after the header we labelled 'Field 1' instead of 'Field 5' which is the actual field number. This is done for simplicity, because later paragraphs which describe the packets will not contain this header or the checksum field.

The header consists of those fields:

```
+------------+--------------+-----------+-----------------+ - -
| ECR number | Clerk number | Euro Flag | Sequence number |
| Integer[2] | Integer[2]   | Integer[1]| Integer[2]      |
+------------+--------------+-----------+-----------------+ - -
```

- ECR number: Is the number of the ECR sent the packet
- Clerk number: The number of clerk which was active during the packet transmittance
- Euro flag: Indicates the 'EURO' status of the ECR Zero means local note (drachmas), non zero means EURO.
- Sequence number: It is a Packet counter. When host receives this packet must check if it equals with the one of the previous packet processed. If yes, then it is retransmittance and must be acknowledged and discarded. Each packet's sequence number equals with previous packet's sequence number plus one. Sequence number will have a range '00' to '99'

Header example: "01/04/0/47/.../45"
This example indicates that ECR#01 sent the packet, clerk #04 was active, mode was drachmas and packet sent had sequence number 47.

- The 'packet descriptor':

In online protocol, the first non header field has a special meaning: it is the packet specifier for the incoming packet. This field is called the 'packet descriptor'. Host software must recognize the packet type by testing this field, which is always of STRING type with fixed length one character.

7.1.1. Online packets in detail -> Item sale packets [SVR]

Item sale packets are sent when in the ECR/POS the clerk does one of the following:

- Sale an ITEM or in department
- Void an ITEM or in department
- Refund an ITEM or in department

```
Packet Descriptors:     'S', 'V', 'R'
Total Field count:      15   (Counting header and checksum fields)
Data Field count:       10    (Without header and checksum fields)
Example:                Fields:   1-2--3----4--5---6-----7-----8-----9----
10----------
                        (header) "S/P/0009/001/0/2.500/123.0/307.5/1000/1010      "
(checksum)


==== Field 1:           Packed descriptors (3 cases)
Type:              STRING
Length:                 Fixed 1 character
Notes:                  'S' for positive sale
                   'V' for void
                   'R' for refund

==== Field 2:           Type of sale identifier
Type:              STRING
Length:                 Fixed 1 character
Notes:                        'P' for PLU
                   'D' for DPT
                   'I' or 'O' for PLU from host

==== Field 3:           Item Index
Type:              INTEGER
Length:                 Default
Notes:                  Index of PLU if field 2 is 'P' (PLU)
                   Index of DPT if field 2 is 'D' (DPT) or 'O' (Host PLU)

==== Field 4:           DPT index of item
Type:              INTEGER
Length:                 Default
Notes:                  Its the DPT number to which the PLU belongs. When field
                   3 is for department, this field equals the previous
                   field.

==== Field 5:           Item VAT category
Type:                   INTEGER
Length:                 Fixed, 1 digit
Notes:                  VAT category (A=0, B=1, ...) of the PLU or DPT.
```

```
==== Field 6:           Sales Quantity
Type:             QUANTITY
Length:                 Default
Notes:                  Quantity of the sale.

==== Field 7:           Item unit price
Type:             AMOUNT
Length:                 Default
Notes:                  The sale item price of the unit.

==== Field 8:           Sales total
Type:             AMOUNT
Length:                 Default
Notes:                  The result of the multiplication (Item Unit Price *
                  Sales quantity)

==== Field 9:           Sales subtotal (*NEW* for this revision)
Type:             AMOUNT
Length:                 Default
Notes:                  The current transaction's subtotal

==== Field 10:          Item search code (*NEW* for this revision)
Type:             STRING
Length:                 Fixed, 16 characters
Notes:                  The item's search code (if no such code exists, spaces
                  are sent).
```

7.1.2. Online packets in detail -> Discount/Markup/Ticket/Coupon packets [DMTU]

This type of packet is sent be the ECR/POS when the clerk does one of the following:

```
  - Discount of PLU or DPT 's price  (Amount)
  - Discount in receipt's subtotal   (Amount)
  - Markup   of PLU or DPT 's price  (Amount)
  - Markup   in receipt's subtotal   (Amount)
  - Discount of PLU or DPT 's price  (Percentage)
  - Discount in receipt's subtotal   (Percentage)
  - Markup   of PLU or DPT 's price  (Percentage)
  - Markup   in receipt's subtotal   (Percentage)
  - Ticket                     (always an amount in subtotal)
  - Coupon   in PLU's or DPT's price (Amount)
  - Coupon   in PLU's or DPT's price (Percentage)
  - Coupon   in receipt's subtotal   (Amount)
  - Coupon   in receipt's subtotal   (Percentage)


Packet descriptors:     'D', 'M'
Total field count:      10 (Counting header and checksum fields)
Data Field count:       5  (Without header and checksum fields)
Example:                Fields:   -1-2---3----4-----5--
                        (header) "/D/P/55.00/1100/5000/" (checksum)


==== Field 1:           Packed descriptor
Type:             STRING
Length:                 Fixed, 1 character
```

```
Notes:                  There are 4 cases in this packet family
                        'D' (Discount, Ticket, Coupon), 'M' (Markup)


==== Field 2:           Type identifier (2 cases)
Type:             STRING
Length:                 Fixed, 1 character
Notes:                  'P' indicates PLU or DPT
                  'S' indicates SUBTOTAL

==== Field 3:           Percentage
Type:             PERCENTAGE
Length:                 Default
Notes:                  The percentage of the modification

==== Field 4:           Amount
Type:             AMOUNT
Length:                 Default
Notes:                  The amount of the modification

==== Field 5:           Subtotal (*NEW* for this revision)
Type:             AMOUNT
Length:                 Default
Notes:                  The current subtotal of the receipt
```

7.1.3. Online packets in detail -> Payment packets [IOPC]

Payment packets are sent by the ECR/POS in one of the following cases:

- Receive on account (cash in)
- Payout   (cash out)
- Payment in receipts
- Change in receipts

```
Packet descriptors:     'I', 'O', 'P', 'C'
Total field count:      10 (Counting header and checksum fields)
Data Field count:       5 (Without header and checksum fields)
Example:                Fields:   -1-2---3-------4-------5-----
                        (header) "/P/0/IAONCOA/10.000/1.000000/" (checksum)

==== Field 1:           Packet descriptor
Type:             STRING
Length:                 Fixed, 1 character
Notes:                  Four cases of payment family packets
                  'O' Indicates Receive on account (Cashin)
                  'I' Indicates Payout (Cashout)
                  'P' Indicates Receipt Payment
                  'C' Indicates Receipt Change

==== Field 2:           Payment Type code
Type:             INTEGER
Length:                 1 - 2 digits,
Notes:                  The Payment's type Code

==== Field 3:           Payment Description
Type:             STRING
Length:                 8
```

```
Notes:                  The programmed payment description for this payment
                  type code

==== Field 4:           Payment Price
Type:             Price
Length:                 Default
Notes:                  Is the price of the Payment type

==== Field 5:           Currency
Type:             RATE
Length:                 Default
Notes:                  Is the programmed exchange rate of this payment type
                  code.
```

7.1.4. Online packets in detail -> Closure packets [012]

Closure packets are sent by the ECR/POS when closing a transaction or when a report has finished. In transaction closure when the packet descriptor is '2' means that the whole receipt has been cancelled. Host application must take care to remove all transaction records for this receipt locally because the ECR/POS is NOT sending any void records to the host.

```
Packet descriptors:     '0', '1', '2'
Total field count:      19 (Counting header and checksum fields)
Data Field count:       14 (Without header and checksum fields)
Example:                Fields:   -1-----2------3------4-----5---6---7----8-----
9-----10--
                        (header)
"/0/13/12/2001/23/34/07/000001/21/4.00/8.00/18.00/"36.00/0.00" (checksum)



==== Field 1:           Packet descriptor
Type:             STRING
Length:                 Fixed, 1 character
Notes:                  There are three cases of closures
                  '0' indicates normal closure of legal receipt
                  '1' indicates closure of a non fiscal report or
                      transaction.
                  '2' indicates the total cancellation of a legal receipt

==== Field 2:           Closure Date (DAY)
Type:             DATE8
Length:                 Default
Notes:                  It is the date printed at the receipt / report

==== Field 3:           Closure Date (MONTH)
Type:             DATE8
Length:                 Default
Notes:                  It is the date printed at the receipt / report

==== Field 4:           Closure Date (YEAR)
Type:             DATE8
Length:                 Default
Notes:                  It is the date printed at the receipt / report
```

==== Field 5: Time (HOUR)
Type: TIME
Length: Default
Notes: It is the time printed at the receipt / report

==== Field 6: Time (MIN)
Type: TIME
Length: Default
Notes: It is the time printed at the receipt / report

==== Field 7: Time (SEC)
Type: TIME
Length: Default
Notes: It is the time printed at the receipt / report

==== Field 8: Counter
Type: INTEGER
Length: Fixed 6 digits
Notes: When the packet descriptor is '0' (normal closure) this field is legal receipt counter. In the other cases ('1' and '2') this field is the illegal receipt counter.

==== Field 9: Closure sub type (*NEW* for this revision)
Type: INTEGER
Length: Fixed, 2 digits
Notes: If the closure came from a report, then this number is the report identifier. If this came from a receipt closure, this number will be zero.

00 = Receipt
01 = X report
02 = Drawer report
03 = PLU sales report
04 = DPT daily sales report
05 = DPT total sales report
06 = Clerk totals
07 = PLU list
08 = DPT list
09 = Parameter list
10 = Payments report
11 = Discount/Markup report
12 = Fiscal report (Z to Z)
13 = Fiscal report (Date to Date)
14 = Cheques report
15 = Fiscal report (totals)
16 = Statistics report
17 =Hourly statistics
18 = Daily statistics
19 = Monthly statistics
20 = Daily accumulators
00 = Z closure report if Field 1 is 1 (illegal transaction)
22 = Ownership change report
23 = Operator List

==== Fields 10 - 14:Transaction totals / Daily Totals (*NEW* for this revision)
Type: AMOUNT
Length: default

```
Notes:                  These fields inform about the closing receipt's vat
                  accumulators when the 'Closure sub-type' is zero (this
                  indicates a receipt closure).
                  These fields inform about the closing day's vat
                  accumulators when the 'Closure sub-type' is '21' (this
                  indicates a Z report).
```

7.1.5. Online packets in detail -> Host database: Item search packet [?]

The item search packet is sent by the ECR/POS when the clerk enters a PLU code using the keyboard or a scanner device. This packet provides a code to the host and requires a transmittance of a reply packet. The reply packet must contain the information: a) a 'found' flag and b) PLU requested data (but only if the item was found in the database).

All fields in the reply packet must be provided (ie: NOT optional)

```
============================== Request packet ================================

Packet descriptor:      '?'
Total field count:      7 (Counting header and checksum fields)
Data Field count:       2 (Without header and checksum fields)
Example request:        Fields:   -1-------2----------
                        (header) "/?/1234567890123456/" (checksum)

==== Field 1:           Packet descriptor
Type:             STRING
Length:                 Fixed, 1 character
Notes:                  Character '?' for host PLU search


==== Field 2:           PLU code for search
Type:             STRING
Length:                 Fixed, 16 characters
Notes:                  The PLU Code that is requested
                  This string is the key for the database lookup.


============================== Reply packet ================================

Total field count:      6 (Counting checksum field)
Data field count:       5 (Without counting checksum field)
Example reply 1:        Fields:   1---------2-----------3----4---5-
                                 "1/ITEM DESCRIPTION/1200.00/0.00/1/" (checksum)
Example reply 2:        Fields:   1-
                                 "0/" (checksum)

==== Field 1:           Found flag
Type:             FLAG
Length:                 1
Notes:                  This flag must be zero when item could not be found
                  in the database, else (if found) it must be set to
                  one. In the 1st case, the rest of the fields must
                  not be sent.

==== Field 2:           Item description
Type:             STRING
Length:                 Variable, 1-20 characters
```

```
Notes:                  The PLU description for the requested code

==== Field 3:           Item unit price
Type:             AMOUNT
Length:                 Default
Notes:                  This is the unit price for the requested code

==== Field 4:           Item discount amount
Type:             AMOUNT
Length:                 Default
Notes:                  This is an auto-discount value for this PLU

==== Field 5:           Department
Type:             INTEGER
Length:                 1
Notes:                  It is the department index which the PLU belongs
```

7.1.6. Online packets in detail -> Host database: Client search packet [K]

The item search packet is sent by the ECR/POS when the clerk enters client code using the keyboard or a scanner device. This packet provides a code to the host and requires a transmittance of a reply packet. The reply packet must contain the information: a) a 'found' flag and b) client requested data (but only if the item found in the database).

All fields in the reply packet must be provided (ie: NOT optional)

```
============================== Request packet ================================

Packet descriptor:      'K'
Total field count:      7 (Counting header and checksum fields)
Data Field count:       2 (Without header and checksum fields)
Example request:        Fields:   -1-------2----------
                        (header) "/?/1234567890123456/" (checksum)

==== Field 1:           Packet descriptor
Type:             STRING
Length:                 Fixed, 1 character
Notes:                  Character 'K' for host Client search

==== Field 2:           Client code for search
Type:             STRING
Length:                 Default
Notes:
                  This string is the key for the database lookup.


============================== Reply packet ================================

Total field count:      7 (Counting checksum field)
Data field count:       6 (Without counting checksum field)
Example reply 1:        Fields:   1-------2------
                                 "1/CLIENT DESCR/2/22/COMMENTS/50.00" (checksum)
Example reply 2:        Fields:   1-
                                 "0/" (checksum)
```

==== Field 1: Found flag
Type: FLAG
Length: 1
Notes: This flag must be zero when client could not be found in the database, else (if found) it must be set to one. In the 1st case, the rest of the fields must not be sent.

==== Field 2: Client description
Type: STRING
Length: Variable, 1-32 characters
Notes: The client description for the requested code.

==== Field 3: Client Bonus Points
Type: INTEGER
Length: Default
Notes: The client's bonus points for the requested code.

==== Field 4: Client's code
Type: INTEGER
Length: Default
Notes: The client's code for the requested code.

==== Field 5: Client's comments
Type: STRING
Length: Default
Notes: The client's comments for the requested code.

==== Field 6: Client's comments
Type: STRING
Length: Default
Notes: The client's comments for the requested code.

==== Field 7: Client's comments
Type: STRING
Length: Default
Notes: The client's comments for the requested code.

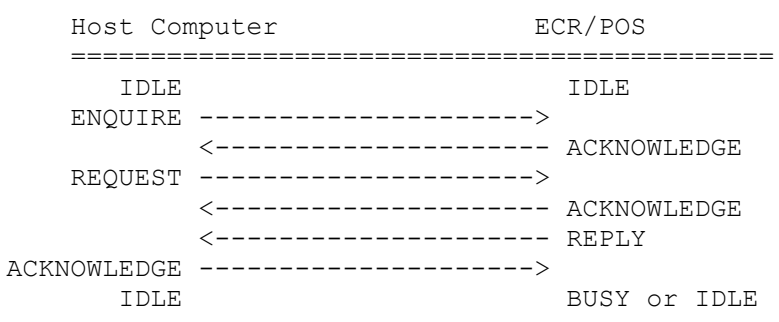
==== Field 8: Client's discount/markup
Type: Amount
Length: Default
Notes: The client's discount/markup amount for the requested code.

8. Command protocol

The command protocol is initiated by the host computer, when the host wants to instruct the ECR/POS to process a specific command. Due to the number of commands this layer supports, they can be grouped as:

- Request information commands
- Setup commands
- Fiscal printer commands
- System commands

The model of the communication the command protocol follows is this:



8.1. Command protocol packets

In the command protocol there are always both packets present in the communication: the request packet and the reply packet. The general form of the request and reply packets follow this model:

```
Request packet:  [Request code] <[Request data]> [checksum]
Reply packet:    [Reply code] / [status fields] / <[Reply data]> [checksum]
```

In request packets, the request data are not always required (notice that 'request data' are inside <>). Additionally in reply packets, the reply data are not always present. All other sections are always present.

8.1.1. A more detailed form of command protocol request packet

```
<---------------------------------- Data ---------------------------------->
                <------------- Optional Section ------------->
+--------------++-------------------------------------------++----------+
| Request code || Field 1 / Field 2 / Field 3 / ... / Field N || Checksum |
+--------------++-------------------------------------------++----------+
```

This defines 3 sections of a request packet:
- The request code section
- The data field section
- The checksum section

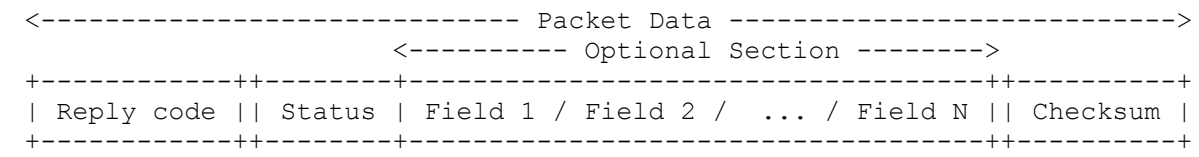
8.1.1.1. Request code

In online protocol packets we dealt with 'packet descriptor' which was a special field for identifying the packet type. In command protocol, the first field is called 'request code' and has the same functionality, although the request code is now sent to the ECR/POS rather than received by it. The request code is always a simple STRING field of one character fixed length.

8.1.1.2. Request packet data fields

Data fields are not always required in all command's request packets. When not a requirement, data fields section is totally omitted, and the checksum section follows directly after the request code.

8.1.2. A more detailed form of command protocol reply packet



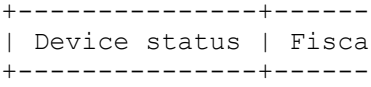
This defines 4 sections of a reply packet:
- The reply code section
- The status section
- The data field section
- The checksum section

8.1.2.1. Reply code section

Reply code is a single numeric field of 2 hexadecimal characters identifying the result of the command execution by the ECR/POS. A zero reply code ('00') indicates that the command executed successfully. A non zero reply code indicates an error in command execution. Error codes returned are explained in detail in a later section. Receiving a non zero reply code means that the command was NOT executed. Receiving a zero reply code means that the command has been or will be successfully executed. Commands that require very little time to execute, such as information retrieve, will be executed before the reply packet is transmitted. This is because the reply packet data fields depend on the command execution itself. Commands that take long time to execute, such as report issuing, will be only checked, a reply packet will be sent, and then will be executed.

8.1.2.2. Status section

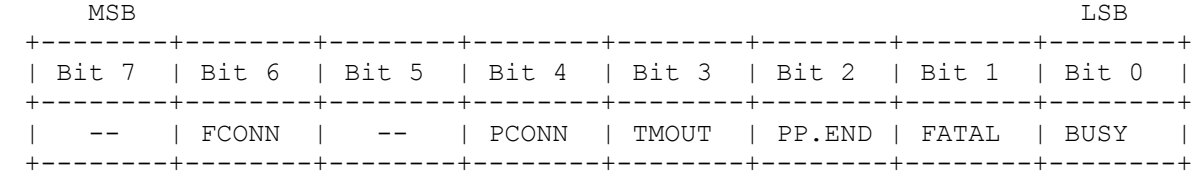
Status is a section consisting of two numeric 2-character hexadecimal fields:



Status section is returned by the ECR/POS to reflect the hardware & fiscal firmware states which must be considered by the host application.

8.1.2.2.1. Device status

Device status informs the host application of some hardware related events of the ECR/POS. The byte that this field forms must be mapped in bits in this way:



Bit 0: Device busy

This bit when set to '1' indicates that the ECR/POS is currently busy executing a previous command or other task. When busy, the ECR/POS may execute some non critical commands and refuse to execute others

replying an error 'Device busy -- Unable to execute' (See error codes). The host must check this bit (requesting a 'status') before issuing any critical commands, or, must keep sending the command until the command is executed (or failed by other reason). BUSY state is a temporary state but, due to very different tasks the ECR/POS may cause the BUSY state, the time which the BUSY flag will be found set is varying from a few milliseconds to few minutes. A host may inform the user after (for example) one minute that the device is busy in other task and ask for a 'retry' or 'cancel' of the requested operation. An example in which a BUSY flag will be set for long time is a fiscal report issuing: When the host (or the ECR/POS user) requests a fiscal report with many records, the report will take long time to finish, thus keeping the BUSY flag set for long. It is highly recommended though that a host should NOT produce a 'device busy' error message to the application user before (at least) twenty (20) seconds. It is also recommended that the host application must allow the user to cancel or retry the operation.

Bit 1: Fatal error

This bit indicates that (when set to one) the ECR/POS detected a fatal hardware related error and cannot process most of the commands. Fatal errors may be a bad fiscal unit, a RAM integrity error or others. From application point of view, this bit means that other critical commands should not be sent, and a service to the ECR/POS is required.

Bit 2: Printer Paper End

This bit indicates (when set to one) that the printer is out of paper, and must be replaced before the previous task has completed its printing duty. Usually, when this flag is set, the 'device busy' flag may be set also, if a previous command that used the printer caused the paper end error. So, it is recommended that the paper end bit MUST be checked before the busy bit. Host application may inform the user of the need to insert a new role of paper to the printing mechanism. After doing so, this bit will be cleared and the command (that detected the paper end) may be retransmitted normally.

Bit 3: Printer timeout

This bit indicates (when set to one) that the printing device is not responding to printing commands. This may be caused by printer's cover which may be open. User must check the cover and close it to continue printing operations. If this is not caused by an open cover and persists after a power off - power on, then the ECR/POS must be serviced.

Bit 4: Printer offline

This bit indicates (when set to one) that the printing device is not responding to printing commands. Recommended action is to power off the printer and on again and retry the command. If the problem persists, the ECR/POS needs to be serviced.

Bit 5: (Reserved, set to zero)

Bit 6: Fiscal Physical unit offline

This bit indicates (when set to one) that a fiscal physical unit is not responding to commands. Because this is a critical error, bit 1 may be also set. Device may need to be serviced.

Bit 7: (Reserved, set to zero)

Example: Assume device status field is '41'. This hexadecimal value, when converted to binary will be '01000001'. The '1's mean that the fiscal unit is offline (bit 6) and the device is busy (bit 0).

8.1.2.2.2. Fiscal status

Fiscal status is a 2-digit numeric hexadecimal field which informs the host about several states of the fiscal firmware inside the ECR/POS. The byte that this field forms must be mapped in bits in this way:

MSB LSB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
FFULL	COUT	CIN	PAYM	--	TROPEN	DAYOPEN	EURO

Bit 0: Euro Mode active

Euro mode flag indicates (when set) that the ECR/POS is operating in EURO. When clear, the ECR is operating in local note (drachmas). This information must be taken into consideration when sending prices or other amounts to the ECR, because in drachmas mode, the 2 decimal digits are truncated.

Bit 1: Day is open

This flag indicates that there is an open day in the ECR/POS. This means that one or more receipts or reports have been issued after a Z clearing report. The day open flag will be zero after the issuing of a Z report and before printing anything else, reports or receipts. A 'day' is defined in the fiscal firmware as the period between two Z closures.

Bit 2: Transaction (Receipt) Open

This flag is indicating that a receipt is currently in 'open' state in the ECR/POS. The flag will be set even if the receipt is in 'payment' state. When this bit is set, information related to an open receipt is valid. An application can prevent errors in commands by detecting this bit. For example, a command 'issue Z report' will fail if this bit is set.

Bit 3: (Reserved)

Bit 4: Transaction in Payment

This flag indicates that ECR/POS has an open receipt in payment state. If it is set, the bit 2 (transaction open) will be also set.

Bit 5: Cash in open

This flag indicates that a cash in receipt is open

Bit 6: Cash out is open

This flag indicates that a cash in receipt is open

Bit 7: Fiscal file full

This flag indicates that the fiscal file used to store daily data after a 'Z' closure report is now full. When this happens, the ECR/POS is unable to issue receipts, reports of any kind except the fiscal periodical report. So, when the host detects this, it must not try to issue receipts or do any other printing.

Example: Assume fiscal status field is '16'. This hexadecimal value, when converted to binary will be '00010110'. The '1's mean that the ECR/POS has a day in open state (bit 1), a receipt is open (bit 2) and the open receipt is in payment state (bit 4).

8.2.1. Command packets in detail -> ECR get identification [a]

This command returns to host information about which ECR/POS unit is communicating with.

```
Request packet:
===============
Request code:           'a'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "a/" (checksum)

==== Field 1:          Request code
Type:                  STRING
Length:        Fixed, 1 character
Notes:                 Must be 'a' for this command


Reply packet:
=============
Total field count:      9 (Counting reply code, status & checksum)
Data field count:       5       (Without reply code, status & checksum)
Example reply:          (reply code)(status) "/99000001/1/?C/EPOSEJ(M310)V1.0R5-
2/EPOSEJ(M310)V1.0R5-2" (checksum)

==== Field 1:  ECR Registration number
Type:                  INTEGER
Length:        DEFAULT
Notes:                 It is the official ECR/POS registration number and it
                       is not programmable. This number is unique to each ECR/POS.
```

```
==== Field 2:  ECR number
Type:                  INTEGER
Length:        1-2 digits
Notes:                 The programmable number of the ECR/POS assigned.

==== Field 3:  Registration owner letters
Type:                  STRING
Length:        Fixed, 2 digits
Notes:                 The ECR's registration characters.

==== Field 4:  ECR MODEL
Type:                  STRING
Length:        Default
Notes:                 The ECR's Model.

==== Field 5:  ECR Firmware Version
Type:                  STRING
Length:        Default
Notes:                 The ECR's Firmware version.
```

8.2.2. Command packets in detail -> PLU get info & stats [p]

This command will return all information about a programmed PLU.

```
Request packet:
===============
Request code:           'p'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "p/125/" (checksum)

==== Field 1:  Request code
Type:                  STRING
Length:        Fixed, 1 character
Notes:                 Must be 'p' for this command

==== Field 2:  PLU number / PLU search code (2 cases)
Type:                  INTEGER-STRING
Length:        DEFAULT
Notes:                 It is the PLU index to read

Reply packet:
=============
Total field count:      15 (Counting reply code, status & checksum)
Data field count:       11  (Without reply code, status & checksum)
Example reply:          (reply code)(status)
"1231231231231231/ABCDEFGHIJKLMNOPQRST
/0001/15/10100111/500/10.00/10.00/3.100/5.810/150.25/"
 (checksum)

==== Field 1:  PLU search code
Type:                  STRING
Length:        Fixed, 13 characters
Notes:                 The search key for this PLU

==== Field 2:  PLU description
```

```
Type:                   STRING
Length:         Fixed, 20 characters
Notes:                  The description for this PLU

==== Field 3:  PLU department
Type:                   INTEGER
Length:         Default
Notes:                  The department number holding this PLU

==== Field 4:  PLU Bonus
Type:                   INTEGER
Length:         Fixed, 2 digits
Notes:                  The bonus value of this PLU

==== Field 5:  PLU settings
Type:                   FLAGS
Length:         Fixed, 8 digits
Notes:                  The flag settings for this PLU as (left to right):
                        1 = Item in package
                        1 = Item can have negative price
                        1 = Item has open price
                        1 = Item is available for sales
                        1 = Item can have a zero price
                        1 = Item will close receipt
                        1 = Print double height
                        1 = Print PLU's department

==== Field 6:  PLU index
Type:                   INTEGER
Length:         Default
Notes:                  The index in internal RAM of the PLU  in Item database.

==== Field 7:  PLU price
Type:                   AMOUNT
Length:         Default
Notes:                  The price for this PLU

==== Field 8:  PLU maximum price
Type:                   AMOUNT
Length:         Default
Notes:                  The maximum valid price for this PLU

==== Field 9:  PLU current stock
Type:                   QTY
Length:         Default
Notes:                  The current stock available for this PLU

==== Field 10: PLU sold quantity
Type:                   QTY
Length:         Default
Notes:                  The current sold quantity of this PLU

==== Field 11: PLU total sales
Type:                   AMOUNT
Length:         Default
Notes:                  The current PLU total sales
```

8.2.3. Command packets in detail ->DPT get info [d]

This command will return all information about a programmed DPT.

```
Request packet:
===============
Request code:           'd'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "d/15/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'd' for this command

==== Field 2:  DPT number
Type:                   INTEGER
Length:         Default
Notes:                  It is the DPT index to read



Reply packet:
=============
Total field count:      15 (Counting reply code, status & checksum)
Data field count:       11  (Without reply code, status & checksum)
Example reply:          (reply code)(status)
"/ABCDEFGHIJKLMNOPQRST/1/101001/10.00/
10.00/0.000/0.00/0.000/0.00/2/5.22"
 (checksum)

==== Field 1:  DPT description
Type:                   STRING
Length:                 Fixed, 20 characters
Notes:                  The description for this DPT

==== Field 2:  DPT vat rate code
Type:                   INTEGER
Length:         Fixed 1 digit, range 1-5
Notes:                  The code of the vat rate of this DPT (1=A, 2=B, ...)

==== Field 3:  DPT settings
Type:                   FLAGS
Length:         Fixed, 7 digits
Notes:                  The flag settings for this DPT as (left to right):
                        1 = Department in package
                        1 = Department can have negative price
                        1 = Department has open price
                        1 = Department is available for sales
                        1 = Department can have a zero price
                        1 = Department will close receipt
                        1 = Print double height

==== Field 4:  DPT price
Type:                   AMOUNT
Length:         Default
Notes:                  The price for this DPT
```

```
==== Field 5:  DPT maximum price
Type:                  AMOUNT
Length:        Default
Notes:                 The maximum valid price for this DPT

==== Field 6:  DPT sold quantity
Type:                  QTY
Length:        Default
Notes:                 The current sold quantity of this DPT

==== Field 7:  DPT total sales
Type:                  AMOUNT
Length:        Default
Notes:                 The current DPT total sales

==== Field 8:  Accumulated DPT sold quantity
Type:                  QTY
Length:        Default
Notes:                 The accumulated sold quantity of this DPT

==== Field 9:  Accumulated DPT total sales
Type:                  AMOUNT
Length:        Default
Notes:                 The accumulated DPT total sales


==== Field 10: Category Code
Type:                  INTEGER
Length:        Default
Notes:                 The Category Code the DPT belongs to


==== Field 11: Second Sales price
Type:                  AMOUNT
Length:        Default
Notes:                 The DPT's second sales price
```

8.2.4. Command packets in detail -> Clerk get info [c]

This command will return all information about a programmed clerk.

```
Request packet:
===============
Request code:           'c'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "c/9/" (checksum)

==== Field 1:  Request code
Type:                  STRING
Length:        Fixed, 1 character
Notes:                 Must be 'c' for this command

==== Field 2:  Clerk number
Type:                  INTEGER
```

```
Length:         1-2 digits
Notes:                  It is the clerk index to read


Reply packet:
=============
Total field count:       19 (Counting reply code, status & checksum)
Data field count:        15 (Without reply code, status & checksum)
Example reply:          (reply code)(status)
                        "OPERATORE 9     /9    /11111111/"
                         (checksum)

==== Field 1:  Clerk description
Type:                   STRING
Length:         Fixed, 16 characters
Notes:                  The clerk's programmed description

==== Field 2:  Clerk access code
Type:                   STRING
Length:         Fixed, 5 characters
Notes:                  The clerk's programmed access code

==== Field 3:  Clerk flags
Type:                   FLAGS
Length:         8 digits
Notes:          The clerk flags as follows (Left to right)

                        1 = Clerk has access to Z reports
                        1 = Clerk has access to ECR/POS programming
                        1 = Clerk has access to X reports
                        1 = Clerk can VOID a sale
                        1 = Clerk can REFUND
                        1 = Clerk can DISCOUNT/MARKUP
                        1 = Clerk can issue CASHIN/OUT tickets
                        1 = Clerk has global access (Manager)

==== Field 4:  Total Receipt's number
Type:                   INTEGER
Length:         DEFAULT
Notes:                  The TOTAL ISSUED RECEIPT's NUMBER

==== Field 5:  Void's Total Amount
Type:                   AMOUNT
Length:         DEFAULT
Notes:                  THE TOTAL VOID's AMOUNT

==== Field 6:  Total Void's number
Type:                   INTEGER
Length:         DEFAULT
Notes:                  The TOTAL VOID's NUMBER

==== Field 7:  Refund's Total Amount
Type:                   AMOUNT
Length:         DEFAULT
Notes:                  THE TOTAL REFUND's AMOUNT

==== Field 8:  Total Refund's number
Type:                   INTEGER
```

```
Length:         DEFAULT
Notes:                  The TOTAL REFUND's NUMBER

==== Field 9:  Cancellation's Total Amount
Type:                   AMOUNT
Length:         DEFAULT
Notes:                  THE TOTAL CANCELLATION's AMOUNT

==== Field 10: Total Cancellation's number
Type:                   INTEGER
Length:         DEFAULT
Notes:                  The TOTAL CANCELLATION's NUMBER

==== Field 11: Cash in's Total Amount
Type:                   AMOUNT
Length:         DEFAULT
Notes:                  THE TOTAL CASH IN's AMOUNT

==== Field 12: Cash out's Total Amount
Type:                   AMOUNT
Length:         DEFAULT
Notes:                  THE TOTAL CASH OUT's AMOUNT

==== Field 13: Cash register's Total Amount
Type:                   AMOUNT
Length:         DEFAULT
Notes:                  THE TOTAL CASH REGISTER's AMOUNT

==== Field 14: Discount's Total Amount
Type:                   AMOUNT
Length:         DEFAULT
Notes:                  THE TOTAL DISCOUNT's AMOUNT

==== Field 15: Markup's Total Amount
Type:                   AMOUNT
Length:         DEFAULT
Notes:                  THE TOTAL MARKUP's AMOUNT
```

8.2.5. Command packets in detail -> Payment get info [y]

This command will return all information about a programmed payment.

```
Request packet:
===============
Request code:           'y'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "y/9/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'y' for this command

==== Field 2:  Payment code
Type:                   INTEGER
Length:         1-2 digits
Notes:                  The payment code
```

```
Reply packet:
=============
Total field count:      14 (Counting reply code, status & checksum)
Data field count:       10 (Without reply code, status & checksum)
Example reply:          (reply code)(status)
                        "/ABCDEFGH/ABC /
10100101/1.000000/10.12/1200.00/1200.00/1/3/0.00"
                        (checksum)

==== Field 1:  Payment description
Type:                  STRING
Length:        Fixed, 8 characters
Notes:                 The payment’s programmed type description

==== Field 2:  Payment shorthand description
Type:                  STRING
Length:        Fixed, 4 characters
Notes:                 A shorthand description, for example 'EUR'

==== Field 3:  Payment type flags
Type:                  FLAGS
Length:        8 digits
Notes:                 Payment flags are mapped as follows (left to right)
                       1 = Payment can give change
                       1 = Must enter amount
                       1 = Must press subtotal
                       1 = Can be used in receive on account (cash in)
                       1 = Can be used in payout (cash out)
                       1 = Will accept payment comments
                       1 = Cannot be negative
                       1 = Decimal

==== Field 4:  Payment rate (currency)
Type:                  RATE
Length:        Default
Notes:                 The payment's programmed rate (currency)

==== Field 5:  Payment daily sum
Type:                  AMOUNT
Length:        Default
Notes:                 It is daily the sum of this payment

==== Field 6:  Payment cash ins
Type:                  AMOUNT
Length:        Default
Notes:                 It is the sum of this payment from cash ins

==== Field 7:  Payment cash outs
Type:                  AMOUNT
Length:        Default
Notes:                 It is the sum of this payment from cash outs

==== Field 8:  Flag for change type
Type:                  INTEGER
Length:        0 - 1 digits
Notes:                 0 change in local currency, 1 change in foreign currency, 2
```

```
                        change in programmed currency

==== Field 9:  The Discount/markup code
Type:                   INTEGER
Length:         0 - 2 digits
Notes:                  The Discount/markup code

==== Field 10: Payment total sum
Type:                   AMOUNT
Length:         Default
Notes:                  It is total the sum of this payment
```

8.2.6. Command packets in detail-> CR parameters read [s]

This command will return various ECR/POS parameters programmed.

```
Request packet:
===============
Request code:            's'
Total field count:       2 (Counting request code & checksum fields)
Data field count:        0 (Without request code & checksum fields)
Example request:         "s" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 's' for this command

Reply packet:
=============
Total field count:       34 (Counting reply code, status & checksum)
Data field count:        30 (Without reply code, status & checksum)
Example reply:          (reply code)(status)
                        "/00110101110000110101110000/999999.99/999.999/9999999.99/99
99.99/100.00/85/99999.99/2/POSID/27/27/27/27/27/27/27/27/27/29/29/29/29/29/29/29/29
/29/250/2"
                        (checksum)

==== Field 1:  Setup Flags
Type:                   FLAGS
Length:         26 digits
Notes:                  Setup flags are mapped as follows (left to right)
                        1 = 0 for normal ,1 for thin
                        1 = 0 no print bitmap ,1 print bitmap
                        1 = 0 normal size ,1 double size print of top bitmap
                        1 = 0 normal size ,1 double size print of bottom bitmap
                        1 = Print departments on Z report
                        1 = Drawer open
                        1 = Drawer open prints receipt
                        1 = Clear PLU stats on Z report
                        1 = Print quantities on receipt
                        1 = Print PLU codes in receipts
                        1 = Check QTY x AMOUNT result for range
                        1 = 0 no message ,1 gives message for negative stock
                        1 = Check PLU codes for special leading codes
                        1 = Activate ONLINE communication
```

1 = Activate online client search
1 = Search local item database if
online search replies 'not found'
1 = 0 no print vat analysis ,1 print vat analysis
1 = 0 no activate clerks ,1 activate clerks, 2 activate
clerks after every sale
1 = Automatically print comments
1 = Print bar-chart or numeric on statistics
1 = Check discount/markup limit
1 = Prints VAT on PLU
1 = Prints logo
1 =Sends Bonus to PC
1 =Prints second quantity
1 =Checks zero sale price from PC

==== Field 2: Maximum sale price
Type: AMOUNT
Length: Default
Notes: The programmed maximum for item price

==== Field 3: Maximum sale quantity
Type: QUANTITY
Length: Default
Notes: The programmed maximum quantity of a sale

==== Field 4: Maximum receipt total
Type: AMOUNT
Length: Default
Notes: The programmed maximum receipt total

==== Field 5: Maximum daily sales total
Type: AMOUNT
Length: Default
Notes: The programmed maximum daily sales total

==== Field 6: Maximum discount/markup amount
Type: AMOUNT
Length: Default
Notes: The programmed maximum discount/markup amount

==== Field 7: Maximum Bonus
Type: AMOUNT
Length: Default
Notes: Maximum Bonus

==== Field 8: Maximum cashier’s amount
Type: AMOUNT
Length: Default
Notes: The programmed maximum cashier’s amount

==== Field 9: ECR/POS number (id)
Type: INTEGER
Length: 1-2 digits
Notes: The programmed ECR/POS number (not the registration number).

==== Field 10: ECR/POS id string
Type: STRING
Length: 1-8 characters
Notes: A programmed string for ECR/POS identification

==== Field 11: Weighting leading code for barcodes (#1)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 12: Weighting leading code for barcodes (#2)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 13: Weighting leading code for barcodes (#3)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 14: Weighting leading code for barcodes (#4)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 15: Weighting leading code for barcodes (#5)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 16: Weighting leading code for barcodes (#6)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 17: Weighting leading code for barcodes (#7)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 18: Weighting leading code for barcodes (#8)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 19: Weighting leading code for barcodes (#9)

Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 20: Pricing leading code for barcodes (#1)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 21: Pricing leading code for barcodes (#2)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 22: Pricing leading code for barcodes (#3)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 23: Pricing leading code for barcodes (#4)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 24: Pricing leading code for barcodes (#5)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 25: Pricing leading code for barcodes (#6)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 26: Pricing leading code for barcodes (#7)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 27: Pricing leading code for barcodes (#8)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 28: Pricing leading code for barcodes (#9)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

```
==== Field 29:  Intensity of printer
Type:                   INTEGER
Length:         0 - 3 digits
Notes:                  Intensity of printer

==== Field 30:  Payment code
Type:                   INTEGER
Length:         0-1 digits
Notes:                  Pay rate code
```

8.2.7. Command packets in detail -> Program PLU [P]

This command is used to program a PLU into the ECR/POS. All fields except the request code (field 1) and the PLU number (field 2) are optional. When not provided, the information in the PLU will not be updated.

```
Request packet:
===============
Request code:            'P'
Total field count:       11 (Counting request code & checksum fields)
Data field count:        9  (Without request code & checksum fields)
Example request:         "P/950/123456789ABC/PLU
DESCRIPTION/1/12/1000.00/1000.00/100.000/10010101/" (checksum)

==== Field 1:   Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'P' for this command

==== Field 2:   PLU number
Type:                   INTEGER
Length:         1-4 digits
Notes:                  The PLU number (index) to program

==== Field 3:   PLU search code
Type:                   STRING
Length:         0-13 characters
Notes:                  The PLU search code (or barcode).

==== Field 4:   PLU description
Type:                   STRING
Length:         1-20 characters
Notes:                  The PLU description printed in receipts

==== Field 5:   PLU department
Type:                   INTEGER
Length:         Default
Notes:                  The department which the PLU belongs

==== Field 6:   PLU item bonus
Type:                   INTEGER
Length:         0-3 digits
Notes:                  This PLU's item bonus


==== Field 7:   PLU item price
```

```
Type:                   AMOUNT
Length:         Default
Notes:                  This PLU's sale price

==== Field 8:  PLU item maximum price
Type:                   AMOUNT
Length:         Default
Notes:                  This PLU's maximum sale price

==== Field 9:  PLU stock
Type:                   QUANTITY
Length:         Default
Notes:                  The stock for this PLU

==== Field 10: PLU flags
Type:                   Flags
Length:         8 digits
Notes:                  The flags for this PLU as follows (left to right)

                        1 = Item in package
                        1 = Item can have negative price
                        1 = Item has open price
                        1 = Item is available for sales
                        1 = Item can have a zero price
                        1 = Item will close receipt
                        1 = Print double height
                        1 = Print Department
Reply packet:
=============
Total field count:       4 (Counting reply code, status & checksum)
Data field count:        0 (Without reply code, status & checksum)
Notes:                   This command's reply packet does not contain
                         additional information; only 1 field reply
                         code, 2 fields status and a checksum.
```

8.2.8. Command packets in detail -> Program DPT (Department) [D]

This command is used to program a DPT into the ECR/POS. All fields except the request code (field 1) and the DPT number (field 2) are optional. When not provided, the information in the DPT will not be updated. Departments cannot be programmed when a day is open in the ECR/POS. Issue a Z report before the department programming to ensure success.

```
Request packet:
===============
Request code:            'D'
Total field count:       10 (Counting request code & checksum fields)
Data field count:        8 (Without request code & checksum fields)
Example request:         "D/12/DEPARTMENT 'A'/1/10.00/1000.00/1001010/12/3.76"
(checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'D' for this command

==== Field 2:  Department number (index)
```

Type: INTEGER
Length: Default
Notes: The department number to program

==== Field 3: Department description
Type: STRING
Length: 1-20 characters
Notes: A description for this department

==== Field 4: Department VAT category
Type: INTEGER
Length: 0-1 digits, range 1-5
Notes: The VAT category this department belongs
(1='A', 2='B', ...)

==== Field 5: Department price
Type: AMOUNT
Length: Default
Notes: The price of this department

==== Field 6: Department maximum price
Type: AMOUNT
Length: Default
Notes: The upper limit price of this department

==== Field 7: Department flags
Type: FLAGS
Length: 0-7 digits
Notes: The department flags mapped as: (left to right)

1 = Department in package
1 = Department can have negative price
1 = Department has open price
1 = Department is available for sales
1 = Department can have a zero price
1 = Department will close receipt
1 = Print double height

==== Field 8: Category Code
Type: INTEGER
Length: Default
Notes: The DPT’s Category Code

==== Field 9: Department second price
Type: AMOUNT
Length: Default
Notes: The department’s second price

Reply packet:
==========
Total field count: 4 (Counting reply code, status & checksum)
Data field count: 0 (Without reply code, status & checksum)
Notes: This command's reply packet does not contain
additional information; only 1 field reply

code, 2 fields status and a checksum.

8.2.9. Command packets in detail -> Program Clerk [C]

This command programs a clerk in the ECR/POS. All fields except the request code and the clerk number are optional. When not provided, the information in the clerk will not be updated.

```
Request packet:
===============
Request code:           'C'
Total field count:      6 (Counting request code & checksum fields)
Data field count:       4 (Without request code & checksum fields)
Example request:        "C/8/CLERK 'A'/12345/01001101/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'C' for this command

==== Field 2:  Clerk number (index)
Type:                   INTEGER
Length:         1-2 digits, range 1-10
Notes:                  The clerk number to program

==== Field 3:  Clerk description
Type:                   STRING
Length:         0 or 16 characters
Notes:                  The clerk string that is printed in receipts

==== Field 4:  Clerk access code
Type:                   STRING
Length:         0 or 5 characters
Notes:                  The clerk access code

==== Field 5:  Clerk flags
Type:                   FLAGS
Length:         8 digits
Notes:                  The clerk's flags as follows (left to right)

                        1 = Clerk has access to Z reports
                        1 = Clerk has access to ECR/POS programming
                        1 = Clerk has access to X reports
                        1 = Clerk can VOID a sale
                        1 = Clerk can REFUND
                        1 = Clerk can DISCOUNT/MARKUP
                        1 = Clerk can issue CASHIN/OUT tickets
                        1 = Clerk has global access (Manager)

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.10. Command packets in detail -> Program payment type [Y]

This command programs a payment type in the ECR/POS. All fields except the request code and the payment number are optional. When not provided, the information in the payment type will not be updated.

```
Request packet:
===============
Request code:           'Y'
Total field count:      12 (Counting request code & checksum fields)
Data field count:       10 (Without request code & checksum fields)
Example request:        "Y/8/Dollars/USD /10011011/352.450000/8/1/1/1" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'Y' for this command

==== Field 2:  Payment type code
Type:                   INTEGER
Length:         1 - 2 digits
Notes:                  The payment type code

==== Field 3:  Payment type description
Type:                   STRING
Length:         1 to 8 characters
Notes:                  The payment type's description

==== Field 4:  Payment type shorthand description
Type:                   STRING
Length:         1 to 4 characters
Notes:          The payment type's shorthand description,
                        for example: "USD" for US dollars.

==== Field 5:  Payment flags
Type:                   Flags
Length:         0 or 8 digits
Notes:                  The payment flags mapped as: (left to right)

                        1 = Payment can give change
                        1 = Must enter amount
                        1 = Must press subtotal
                        1 = Can be used in receive on account (cash in)
                        1 = Can be used in payout (cash out)
                        1 = Will accept payment comments
                        1 = Cannot be negative
                        1 = Decimal

==== Field 6:  Payment rate (currency)
Type:                   RATE
Length:         Default
Notes:                  The payment's rate (currency).

==== Field 7:  The Discount/markup code
Type:                   INTEGER
Length:         0 - 2 digits
```

```
Notes:                  The Discount/markup code

==== Field 8:  Index of change method
Type:                   INTEGER
Length:         0 - 1 digits
Notes:                  Index of change method Range 1-3

==== Field 9:  Flag active
Type:                   INTEGER
Length:         0 - 1 digits
Notes:                  0 for inactive, 1 for active

==== Field 10: CREDIT
Type:                   INTEGER
Length:         0 - 1 digits
Notes:                  0 for inactive, 1 for active


Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.11. Command packets in detail -> Program ECR Parameters [S]

This command programs various 'setup' information in the ECR/POS. All fields are optional so the host can selectively modify specific fields.

```
Request packet:
===============
Request code:           'S'
Total field count:      32 (Counting request code & checksum fields)
Data field count:       30 (Without request code & checksum fields)
Example request:
"S/00110101110000110101110000/999999.99/999.999/9999999.99/9999.99/100.00/85/99999.
99/2/POSID/27/27/27/27/27/27/27/27/27/29/29/29/29/29/29/29/29/29/250/2" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'S' for this command

==== Field 2:  Setup flags
Type:                   FLAGS
Length:         0 or 26 digits
Notes:                  Various flags mapped as: (left to right)

                        1 = 0 for normal ,1 for thin
                        1 = 0 no print bitmap ,1 print bitmap
                        1 = 0 normal size ,1 double size print of top bitmap
                        1 = 0 normal size ,1 double size print of bottom bitmap
                        1 = Print departments on Z report
                        1 = Drawer open
                        1 = Drawer open prints receipt
```

```
                        1 = Clear PLU stats on Z report
                        1 = Print quantities on receipt
                        1 = Print PLU codes in receipts
                        1 = Check QTY x AMOUNT result for range
                        1 = 0 no message ,1 gives message for negative stock
                        1 = Check PLU codes for special leading codes
                        1 = Activate ONLINE communication
                        1 = Activate online client search
                        1 =  Search local item database if
                             online search replies 'not found'
                        1 = 0 no print vat analysis ,1 print vat analysis
                        1 = 0 no activate clerks ,1 activate clerks, 2 activate
                        clerks for every sale
                        1 = Automatically print comments
                        1 = Print bar-chart or numeric on statistics
                        1 = Check discount/markup limit
                        1 = Prints VAT in PLU
                        1 = Print logo
                        1 = Send bonus
                        1 = Print second quantity
                        1 = Check zero sales from PC

==== Field 3:  Maximum item price
Type:                   AMOUNT
Length:         0 or Default
Notes:                  A global maximum limit for item prices

==== Field 4:  Maximum sale quantity
Type:                   QUANTITY
Length:         0 or Default
Notes:                  A global maximum limit for sale quantities

==== Field 5:  Maximum total amount
Type:                   AMOUNT
Length:         0 or Default
Notes:                  A global maximum limit for receipt total

==== Field 6:  Maximum daily sales amount
Type:                   AMOUNT
Length:         Default
Notes:                  A global maximum limit for daily sales total

==== Field 7:  Maximum discount/markup amount
Type:                   AMOUNT
Length:         Default
Notes:                  The programmed maximum discount/markup amount

==== Field 8:  Maximum Bonus
Type:                   AMOUNT
Length:         Default
Notes:                  Maximum Bonus

==== Field 9:  Maximum cashier's amount
Type:                   AMOUNT
Length:         Default
Notes:                  A global maximum limit for cashier's amount

==== Field 10: ECR/POS number (id)
```

Type: INTEGER
Length: 0 - 3 digits
Notes: The ECR/POS number (not the registration number)

==== Field 11: ECR/POS id string
Type: STRING
Length: 0 - 8 chars
Notes: An additional string for ECR/POS identification

==== Field 12: Weighting leading code for barcodes (#1)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 13: Weighting leading code for barcodes (#2)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 14: Weighting leading code for barcodes (#3)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 15: Weighting leading code for barcodes (#4)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 16: Weighting leading code for barcodes (#5)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 17: Weighting leading code for barcodes (#6)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 18: Weighting leading code for barcodes (#7)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 19: Weighting leading code for barcodes (#8)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 20: Weighting leading code for barcodes (#9)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains Weight information.

==== Field 21: Pricing leading code for barcodes (#1)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 22: Pricing leading code for barcodes (#2)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 23: Pricing leading code for barcodes (#3)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 24: Pricing leading code for barcodes (#4)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 25: Pricing leading code for barcodes (#5)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 26: Pricing leading code for barcodes (#6)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 27: Pricing leading code for barcodes (#7)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 28: Pricing leading code for barcodes (#8)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode contains price information.

==== Field 29: Pricing leading code for barcodes (#9)
Type: INTEGER
Length: 1-2 characters
Notes: A leading code for barcodes to indicate that the barcode

contains price information.

==== Field 30: Intensity of printer
Type: INTEGER
Length: 0 - 3 digits
Notes: Intensity of printer

==== Field 31: Payment code
Type: INTEGER
Length: 0-2 digits
Notes: Pay rate code

Reply packet:
=============
Total field count: 4 (Counting reply code, status & checksum)
Data field count: 0 (Without reply code, status & checksum)
Notes: This command's reply packet does not contain additional information; only 1 field reply code, 2 fields status and a checksum.

8.2.12. Command packets in detail -> Program header [H]

Programs the header in the ECR/POS. The header is stored in the fiscal memory. Lines that will not be passed in the command will not be printed. To program a blank line, the host must pass the line filled with spaces. The lines provided for header will NOT
be centred automatically.

Request packet:
===============
Request code: 'H'
Total field count: 14 (Counting request code & checksum fields)
Data field count: 12 (Without request code & checksum fields)
Example request: "H/0/HEADER LINE 1/0/HEADER LINE 2/0/HEADER LINE 3/0/HEADER LINE 4/0/HEADER LINE 5/2/3" (checksum)

==== Field 1: Request code
Type: STRING
Length: Fixed, 1 character
Notes: Must be 'H' for this command

==Fields 2,4,6,8,10,12: Header line printing types
Type: INTEGER
Length: 0-1Digits
Notes: The printing type for each header line as:

1 = Normal printing
2 = Double height
3 = Double width
4 = Double width and height
When printing double width, only 20 characters of the line are printed.

==Fields 3,5,7,9,11,13: Header lines
Type: STRING
Length: 0-32 characters
Notes: The lines of the header

Reply packet:
=============
Total field count: 4 (Counting reply code, status & checksum)
Data field count: 0 (Without reply code, status & checksum)
Notes: This command's reply packet does not contain additional information; only 1 field reply code, 2 fields status and a checksum.

8.2.13. Command packets in detail -> Program footer [F]

Programs the footer in the ECR/POS. Lines that will not be passed in the command will not be printed. To program a blank line, the host must pass the line filled with spaces. The lines provided for footer will NOT be centred automatically.

Request packet:
===============
Request code: 'F'
Total field count: 8 (Counting request code & checksum fields)
Data field count: 6 (Without request code & checksum fields)
Example request: "F/0/FOOTER LINE 1/0/FOOTER LINE 2/0/FOOTER LINE 3" (checksum)

==== Field 1: Request code
Type: STRING
Length: Fixed, 1 character
Notes: Must be 'F' for this command

==== Fields 2,4,6: Footer line printing types
Type: INTEGER
Length: 0-1
Notes: The printing type for each footer line as:

1 = Normal printing
2 = Double height
3 = Double width
4 = Double width and height
When printing double width, only 20 characters of the line are printed.

==== Fields 3,5,7: Footer lines
Type: STRING
Length: 0-32 characters
Notes: The text lines of the footer

Reply packet:
=============
Total field count: 4 (Counting reply code, status & checksum)
Data field count: 0 (Without reply code, status & checksum)
Notes: This command's reply packet does not contain additional information; only 1 field reply code, 2 fields status and a checksum.

8.2.14. Command packets in detail -> Program scrolling message [M]

Programs a message in the ECR/POS which will be scrolled left to right when the ECR/POS is idle.

```
Request packet:
===============
Request code:           'M'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "M/THIS IS MY SCROLLING MESSAGE/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'M' for this command

==== Field 2:  Message
Type:                   STRING
Length:         0-96 characters
Notes:                  The scrolling message to program

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum
```

8.2.15. Command packets in detail -> Item sale [3]

This command belongs to the fiscal printer commands. It is used to sale an item remotely. If a transaction is not open, the ECR/POS will open it. Not all fields in this command are optional.

```
Request packet:
===============
Request code:           '3'
Total field count:      9 (Counting request code & checksum fields)
Data field count:       7 (Without request code & checksum fields)
Example request:        "3/S/ITEM-1/ADDITIONAL INFO/1.000/100.00/1/5/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be '3' for this command

==== Field 2:  Operation
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  The operation code must be one of the following:
                        'S' for positive sale
                        'V' for void (negative) sale
                        'R' for refund
```

```
==== Field 3:  Item description
Type:                  STRING
Length:        1-20 characters
Notes:                 The description of the item (required)

==== Field 4:  Sale extended description line
Type:                  STRING
Length:        0-30 characters
Notes:                 An extra information line printed
                       below the 'sale' line. Optional

==== Field 5:  Sales quantity
Type:                  QUANTITY
Length:        Default
Notes:                 The item sale quantity

==== Field 6:  Item unit price
Type:                  AMOUNT
Length:        Default
Notes:                 The item's unit price for the sale

==== Field 7:  Department owning item
Type:                  INTEGER
Length:        1-2
Notes:                 The item's department number.
                       (1='A', 2='B'...)

==== Field 8:  Item Vat rate
Type:                  AMOUNT
Length:        Default
Notes:                 The VAT rate that applies to this item. This rate MUST be
                       equal to the rate which the department of this item
                       corresponds to.


Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.16. Command packets in detail -> Discount/Markup /Coupon/Ticket [4]

This command is for issuing discounts, markups, ticket sales, or coupons to the ECR/POS. A transaction must be open.

```
Request packet:
===============
Request code:           '4'
Total field count:      8 (Counting request code & checksum fields)
Data field count:       6 (Without request code & checksum fields)
Example request:        "4/1200.00/MARKUP//4/0/0/1/" (checksum)

==== Field 1:  Request code
Type:                  STRING
Length:        Fixed, 1 character
```

Notes: Must be '4' for this command

==== Field 2: Amount of operation/Percentage of operation
Type: AMOUNT
Length: Default
Notes: The amount or percentage of the discount/markup/coupon/ticket operation.

==== Field 3: The operation description
Type: STRING
Length: 0-16 characters
Notes: Optional string for description of operation. If not passed, the default string (programmed in the ECR/POS's setup) will be used.

==== Field 4: Operation extended description
Type: STRING
Length: 0-30 characters
Notes: Optional string for additional information printing of the operation. Prints one additional line below the operation printing lines.

==== Field 5: Operation code
Type: INTEGER
Length: Fixed, 1 digit, range 0-3
Notes: The operation code must be one of the following:

0 = Discount
1 = Markup
2 = Coupon
3 = Ticket
4 = The discount/markup will be used according to the fields 6 and 7 of the command (temporary)

If field 5 value is not 4, then the discount/markup's fields 6 and 7 will not matter and the discount/markup will be executed as programmed in the ECR

==== Field 6: Operation mode
Type: INTEGER
Length: Fixed, 1 digit, range 0-1
Notes: The operation mode specifies where the operation will take place:

0 = Do the operation in the Last item sold
1 = Do the operation in receipts current subtotal
Ticket operation is always performed on subtotal regardless of the operation mode passed

==== Field 7: Amount/Percentage specifier
Type: INTEGER
Length: Fixed, 1 digit, range 0-1
Notes: This field specifies whether the field 1 is a percentage or the field 1 is an amount.
0 = Percentage, 1 = Amount

```
Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.17. Command packets in detail -> Payments in receipt/cash in/cash out [5]

This command is used in 3 cases:

- In receipts (positive)
- In cash in (positive)
- In cash out (negative)

When a receipt is open, this command will force the ECR/POS firmware state to enter payment mode.

```
Request packet:
===============
Request code:           '5'
Total field count:      7 (Counting request code & checksum fields)
Data field count:       5 (Without request code & checksum fields)
Example request:        "5/1/1000.00/PAYMNT DESCR/1.000000/CREDIT/" (checksum)

==== Field 1:  Request code
Type:                  STRING
Length:        Fixed, 1 character
Notes:                 Must be '5' for this command

==== Field 2:  Payment type (index)
Type:                  INTEGER
Length:        1-2 digits
Notes:                 The payment code

==== Field 3:  Payment Amount
Type:                  AMOUNT
Length:        Default
Notes:                 The amount for the payment (must be zero for cash in/out)

==== Field 4:  Payment description
Type:                  STRING
Length:        0-30 characters
Notes:                 An optional string for extended description of the payment.
                       This line will be printed after the actual payment lines.

==== Field 5:  Payment currency
Type:                  RATE
Length:        Default
Notes:                 The currency (rate) of the payment type.
                       The payment currency is used only when the payment type
                       (field 2) is 20 (a generic payment slot). When not 20, this
                       field is ignored.

==== Field 6:  Payment extended description
Type:                  STRING
Length:        0-8 characters
```

```
Notes:                  The description of the payment.
                        The payment description is used only when the payment type
                        (field 2) is 20 (a generic payment slot). When not 20 this
                        field is ignored.

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.18. Command packets in detail -> Set active clerk [1]

This command is used to set an active clerk for issuing receipts and reports. The clerk cannot be changed if any receipt is open (legal or not).

```
Request packet:
===============
Request code:           '1'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "1/8/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be '1' for this command

==== Field 2:  Clerk number (index)
Type:                   INTEGER
Length:         1-2 digits
Notes:                  The clerk code to be activated

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.19. Command packets in detail -> Set client name [2]

This command is for searching for an online client description.

```
Request packet:
===============
Request code:           '2'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "2/Panagiotou Dimitris/" (checksum)

==== Field 1:  Request code
Type:                   STRING
```

```
Length:         Fixed, 1 character
Notes:                  Must be '2' for this command

==== Field 2:  Client description
Type:                   STRING
Length:         1-32 characters
Notes:                  A description for a client printed in the receipt.

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.20. Command packets in detail -> Cancel receipt [+]

This command is for cancelling an open legal receipt which is not in payment state. The ECR/POS will remove any record keeping for this receipt and will terminate it.

```
Request packet:
===============
Request code:           '+'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "+/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be '+' for this command

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.21. Command packets in detail -> Read transaction totals [9]

This command is used for getting the current transaction totals when a receipt is currently open. If a receipt is not open, the transaction totals will be zero.

```
Request packet:
===============
Request code:           '9'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "9/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
```

```
Notes:                  Must be '9' for this command

Reply packet:
=============
Total field count:      11 (Counting reply code, status & checksum)
Data field count:       7  (Without reply code, status & checksum)
Example reply:          (reply code)(status)
"/100.00/200.00/300.00/400.00/500.00/17/1500.00/"(checksum)

==== Fields 1,2,3,4,5: Receipt Accumulators
Type:                   AMOUNT
Length:         Default
Notes:                  Receipt's sums belonging to each VAT category

==== Field 6:  Receipt number
Type:                   INTEGER
Length:         1-6 digits
Notes:                  The open receipt's number
                        When a receipt is not open, it indicates the last receipt's
                        number

==== Field 7:  Transaction Total
Type:                   AMOUNT
Length:         Default
Notes:                  The amount that requires payment before the transaction can
                        be closed. If the receipt is not in payment state, this
                        amount equals to the sum of all VAT accumulators. When the
                        receipts is in payment state, it shows the amount remain to
                        be paid.
```

8.2.22. Command packets in detail -> Read daily totals [0]

This command is used to read the daily totals accumulated in one day.

```
Request packet:
===============
Request code:           '0'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "0/" (checksum)

==== Field 1:          Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be '0' for this command

Reply packet:
=============
Total field count:      15 (Counting reply code, status & checksum)
Data field count:       11 (Without reply code, status & checksum)
Example reply:          (reply code)(status)
"/10.00/20.00/30.00/40.00/50.00/150.00/121/11/0.00/0.00/0.00/" (checksum)

==== Field 1,2,3,4,5:  Daily VAT
Type:                   AMOUNT
Length:         Default
Notes:                  Daily sums belonging to each VAT category
```

```
==== Field 6:  Daily total
Type:                  AMOUNT
Length:        Default
Notes:                 Daily total sum (the sum of fields 1 to 5)

==== Field 7:  Receipt number
Type:                  INTEGER
Length:        1-6 digits
Notes:                 The open receipt's number
                       When a receipt is not open, it indicates the last receipt's
                       number

==== Field 8:  Illegal receipt number
Type:                  INTEGER
Length:        1-6 digits
Notes:                 The open illegal receipt's number
                       When an illegal receipt is not open, it indicates the last
                       receipt's number

==== Field 9:  Voids total
Type:                  AMOUNT
Length:        Default
Notes:                 The sum of all voids during the day

==== Field 10: Refunds total
Type:                  AMOUNT
Length:        Default
Notes:                 The sum of all refunds during the day

==== Field 11: Cancels total
Type:                  AMOUNT
Length:        Default
Notes:                 The sum of all cancels during the day
```

8.2.23. Command packets in detail -> Open cash in/out transaction [6]

This command is for opening a cash-in or cash-out transaction to the ECR/POS. This must be done prior to sending any payment commands to the ECR. A legal receipt or a cash-in / cash out transaction must not be open. After successful open of either cashin or cashout,
host may issue 'payment' commands. Issuing a payment command with zero value will close the cashin/out transaction.

```
Request packet:
===============
Request code:           '6'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "6/1" (checksum)

==== Field 1:  Request code
Type:                  STRING
Length:        Fixed, 1 character
Notes:                 Must be '6' for this command

==== Field 2:  Cash in / Cash out type
```

```
Type:                   INTEGER
Length:         Fixed, 1 digit
Notes:                  The type can be:
                        0 = Open Cash in transaction
                        1 = Open Cash out transaction

Reply packet:
=============
Total field count:       4 (Counting reply code, status & checksum)
Data field count:        0 (Without reply code, status & checksum)
Notes:                   This command's reply packet does not contain
                         additional information; only 1 field reply
                         code, 2 fields status and a checksum.
```

8.2.24. Command packets in detail -> Issue report [x]

This command is used to issue the standard reports supported by the ECR.

```
Request packet:
===============
Request code:            'x'
Total field count:       4 (Counting request code & checksum fields)
Data field count:        2 (Without request code & checksum fields)
Example request:         "x/1/1/3" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'x' for this command

==== Field 2:  Report type
Type:                   INTEGER
Length:         1 digit, range 0-9
Notes:                  The report type can be:
                        1 = X sales total report
                        2 = Drawer report
                        3 = PLU report
                        4 = Department report type 1
                        5 = Department report type 2
                        6 = Clerk report
                        7 = Z closure report (if fields 3,4 are 0 then a copy of the
                        last Z report is issued)
                        8 = Statistics report(NOT USED)
                        9 = Items List (field 3 shows the
                        starting Item's number and filed
                        number 4 shows the Item's ending
                        number)


==== Field 3:  The Clerk number
Type:                   INTEGER
Length:         0 - 2 digits
Notes:                  The Clerk number to begin the clerk's report Leave blanc for
                        the other options of Field 2

==== Field 4:  The Clerk number
Type:                   INTEGER
Length:         0 - 2 digits
```

```
Notes:                  The Clerk number to end the clerks report Leave blanc for
                        the other options of Field 2, except option number 9


Reply packet:
=============
Total field count:       4 (Counting reply code, status & checksum)
Data field count:        0 (Without reply code, status & checksum)
Notes:                   This command's reply packet does not contain
                         additional information; only 1 field reply
                         code, 2 fields status and a checksum.
```

8.2.25. Command packets in detail -> Clear statistics [8]

This command is used to clear statistic accumulators and counters for various files.

```
Request packet:
===============
Request code:            '8'
Total field count:       4 (Counting request code & checksum fields)
Data field count:        2 (Without request code & checksum fields)
Example request:         "8/1/0" (checksum)

==== Field 1:  Request code
Type:                    STRING
Length:         Fixed, 1 character
Notes:                   Must be '8' for this command

==== Field 2:  File no
Type:                    INTEGER
Length:         Fixed, 1 digit
Notes:                   The file to clear. Can be:
                         0 - for PLU statistics
                         1 - for DPT statistics
                         2 - for generic statistics

==== Field 3:  Generic selector
Type:                    INTEGER
Length:         Fixed, 1 digit
Notes:                   The generic statistics to clear
                         Options 1,2,3 used only if field 2 is set to 2. Can be:

                         0 - for full general stats clear
                         1 - for hourly stats clear
                         2 - for daily stats clear
                         3 - for monthly stats clear

Reply packet:
=============
Total field count:       4 (Counting reply code, status & checksum)
Data field count:        0 (Without reply code, status & checksum)
Notes:                   This command's reply packet does not contain
                         additional information; only 1 field reply
                         code, 2 fields status and a checksum.
```

8.2.26. Command packets in detail -> Device status read [?]

This command is used to retrieve the status of the ECR/POS. Because this status information is always sent in the reply packet, the status command need not any additional information receive or return.

```
Request packet:
===============
Request code:           '?'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "?/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be '?' for this command

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum
```

8.2.27. Command packets in detail -> Real time clock read [t]

This command is used to read the ECR/POS's real time clock.

```
Request packet:
===============
Request code:           't'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "t/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 't' for this command

Reply packet:
=============
Total field count:      6 (Counting reply code, status & checksum)
Data field count:       2 (Without reply code, status & checksum)
Example reply:          (reply code)(status)
                        "200404/160913"
                        (checksum)

==== Field 1:  System date
Type:                   DATE6
Length:         Fixed, 6 digits
Notes:                  The current date in ECR/POS

==== Field 2:  System time
Type:                   TIME
```

```
Length:         Fixed, 6 digits
Notes:                  The current time in ECR/POS
```

8.2.28. Command packets in detail -> Program Real Time Clock [T]

This command is used to program the ECR/POS real time clock (ie: time and date). For this command to succeed, the 'clock' jumper must be short, otherwise the command will fail. Also, the date must not be prior to the last fiscal record's date.

```
Request packet:
===============
Request code:           'T'
Total field count:      4 (Counting request code & checksum fields)
Data field count:       2 (Without request code & checksum fields)
Example request:        "T/200404/160913/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'T' for this command

==== Field 2:  System date
Type:                   DATE6
Length:         Default
Notes:                  The date to set in RTC (Real time clock)

==== Field 3:  System time
Type:                   TIME
Length:         Default
Notes:                  The time to set in RTC

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.29. Command packets in detail -> Device write [7]

This command is used to print user messages to display or the printer. When sending printing lines, after every six lines the firmware will aromatically print the 'ILLEGAL RECEIPT' message.

```
Request packet:
===============
Request code:           '7'
Total field count:      5 (Counting request code & checksum fields)
Data field count:       3 (Without request code & checksum fields)
Example request:        "7/0/1/-- WELCOME --/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be '7' for this command
```

```
==== Field 2:  Device code
Type:                  INTEGER
Length:        Fixed, 1 digit, range 0-2
Notes:                 The device to which the message will be printed

                       0 = Print to Display
                       1 = Print to 'Receipt & Journal' stations
                       2 = Print to Slip station
                       (When printing in slip station, the firmware automatically
                       prints a non-fiscal message every ix lines of printing)

==== Field 3:  Printing type
Type:                  INTEGER
Length:        Fixed, 1 digit, range 1 - 3
Notes:                 The printing style for the device requested.
                       For Display, types can be:

                         1 - Print to 1st line only
                         2 - Print to 2nd line only
                         3 - Print to 1st line and clear 2nd
                       For printer, types can be:

                         1 - Normal print
                         2 - Double height
                         3 - Double width
                         4 - Double width and height

==== Field 4:  Print line
Type:                  STRING
Length:        0-32 characters
Notes:                 The line to send to the selected device

                       For display, the line may be up to 20 characters

                       For printer, the line may be 40 or 80 characters depending
                       on the selected station (Receipt and Journal stations can
                       print up to 40 characters and the Slip can print up to 80
                       characters). When printing in double width, the character
                       width is halved.


Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.30. Command packets in detail -> Drawer open [q]

This command is for remotely forcing the drawer to open.

```
Request packet:
===============
Request code:           'q'
```

```
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "q/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'q' for this command

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.31. Command packets in detail -> Printer feed [w]

This command is for feeding the printer. (*NEW* for this revision)

```
Request packet:
===============
Request code:           'w'
Total field count:      4 (Counting request code & checksum fields)
Data field count:       2 (Without request code & checksum fields)
Example request:        "w/1/2/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'q' for this command

==== Field 2:  Station number
Type:                   INTEGER
Length:         1 digit, range 0-3
Notes:                  0 = Receipt + Journal station
                        1 = Receipt station
                        2 = Journal station
                        3 = Slip station

==== Field 3:  Feed count
Type:                   INTEGER
Length:         1 digit, range 1-9
Notes:                  Number of lines to feed the selected station

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.32. Command packets in detail -> End user printing [m]

This command is used to terminate any user-report printing in slip or other

stations. The command will eject the paper if the printing is on slip station. If the printing was in receipt/journal, the command will cut the paper if a cutter is available

```
Request packet:
===============
Request code:           'm'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "m/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'm' for this command

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.33. Command packets in detail -> Get version/device info [v]

This command will return version information for protocol and firmware of the ECR/POS. Also returns the device capabilities. (*NEW* in this revision)

```
Request packet:
===============
Request code:           'v'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "v/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'v' for this command

Reply packet:
=============
Total field count:      15 (Counting reply code, status & checksum)
Data field count:       11  (Without reply code, status & checksum)
Example reply:          (reply code)(status) "10.01/1.001/ 68/5000/ 61/21/10/10/ 6/
5/30" (checksum)

==== Field 1:  Firmware revision
Type:                   STRING
Length:         5 chars
Notes:                  The ECR/POS firmware version.
```

```
==== Field 2:  Protocol revision
Type:                  STRING
Length:        5 chars
Notes:                 The ECR/POS protocol version.


==== Field 3:  Key's total number
Type:                  INTEGER
Length:        1-5 digits
Notes:                 Maximum keys

==== Field 4:  Total PLU's number
Type:                  INTEGER
Length:        1-5 digit
Notes:                 Maximum plus

==== Field 5:  Total DPT's number
Type:                  INTEGER
Length:        1-5 digits
Notes:                 Maximum departments

==== Field 6:  Total's payment's number
Type:                  INTEGER
Length:        1-5 digits
Notes:                 Maximum payments

==== Field 7:  Total's clerk's number
Type:                  INTEGER
Length:        1-5 digits
Notes:                 Maximum clerks

==== Field 8:  Total Discount/Markups number
Type:                  INTEGER
Length:        1-5 digits
Notes:                 Maximum disc/markups

==== Field 9:  Total Header Lines
Type:                  INTEGER
Length:        1-5 digits
Notes:                 Header lines

==== Field 10: Fixed Number
Type:                  INTEGER
Length:        Fixed
Notes:                 Fixed number '5'

==== Field 11: Total's category number
Type:                  INTEGER
Length:        Default
Notes:                 Maximum Category number
```

8.2.34. Command packets in detail -> ECR's online communication [n]

Set ECR online/offline

```
Request packet:
===============
```

```
Request code:           'n'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "n/1" (checksum)

==== Field 1:
Type:                   STRING
Length:         Fixed 1 character
Notes:                  Must be 'n' for this command

==== Field 2:  Set ONLINE or OFFLINE
Type:                   INTEGER
Length:         Fixed 1 digit
Notes:                  2 for online, 1 for offline

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.35. Command packets in detail -> Scroll-Line message displaying [o]

```
 Scroll message displaying

Request packet:
===============
Request code:           'o'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "o/TEST MESSAGE" (checksum)

==== Field 1:  Request Code
Type:                   STRING
Length:         Fixed 1 character
Notes:                  Must be 'o' for this command

==== Field 2:  Advertising Message
Type:                   STRING
Length:         0 - 96 characters
Notes:                  Scrolling message


Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.36. Command packets in detail -> Program DISCOUNT/ MARKUP [u]

```
  Program a DISCOUNT/ MARKUP
```

Request packet:
===============
Request code: 'u'
Total field count: 12 (Counting request code & checksum fields)
Data field count: 10 (Without request code & checksum fields)
Example request: "u/2/DISCDESCR/1/1/1/1/1/1000.00/2000.00/100.00" (checksum)

==== Field 1: Request code
Type: STRING
Length: Fixed 1 character
Notes: Must be 'u' for this command

==== Field 2: Discount’s Markup Number
Type: INTEGER
Length: 1 - 2 digits
Notes: The Discount/markup number to be programmed

==== Field 3: Description
Type: STRING
Length: 0 - 16 chars
Notes: The Discount/markup description

==== Field 4: Flag discount/markup
Type: INTEGER
Length: 0 - 1 digits
Notes: 0 for discount, 1 for markup

==== Field 5: Flag price/percentage
Type: INTEGER
Length: 0 - 1 digits
Notes: 0 for percentage, 1 for price

==== Field 6: Flag sales/subtotal/both
Type: INTEGER
Length: 0 - 1 digits
Notes: 0 for sales, 1 for subtotal, 2 for both

==== Field 7: Flag active
Type: INTEGER
Length: 0 - 1 digits
Notes: 0 for inactive, 1 for active

==== Field 8: Flag ticket
Type: INTEGER
Length: 0 - 1 digits
Notes: 0 no, 1 yes

==== Field 9: Amount/percentage for discount/markup
Type: AMOUNT
Length: Default
Notes: Amount/percentage for discount/markup

==== Field 10: Maximum price
Type: AMOUNT
Length: Default
Notes: The maximum price

```
==== Field 11: Maximum percentage
Type:                  PERCENTAGE
Length:        0 - 6 digits
Notes:                 Maximum percentage


Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.37. Command packets in detail -> READ DISCOUNT/ MARKUP [V]

```
  READ DISCOUNT/ MARKUP

Request packet:
===============
Request code:           'V'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "V/10" (checksum)

==== Field 1:  Request Code
Type:                  STRING
Length:        Fixed 1 char
Notes:                 Must be 'V' for this command

==== Field 2:  Number
Type:                  INTEGER
Length:        1 - 2 digits
Notes:                 The Discount/markup number



Reply packet:
=============
Total field count:      15 (Counting reply code, status & checksum)
Data field count:       11 (Without reply code, status & checksum)
Example reply:          (reply code)(status)
"/abcdefghijklmnop/0/1/0/1/1/12.75/16/5/135.67/6/3/5/
(checksum)”

==== Field 1:  Discount/Markup description
Type:                  STRING
Length:        16 characters
Notes:                 The discount/Markup description

==== Field 2:  Flag discount/markup
Type:                  INTEGER
Length:        1 digit
Notes:                 0 for discount, 1 for markup

==== Field 3:  Flag price/percentage
```

```
Type:                   INTEGER
Length:         1 digit
Notes:                  0 for percentage, 1 for price

==== Field 4:  Flag sales/subtotal/both
Type:                   INTEGER
Length:         1 digit
Notes:                  0 for sales, 1 for subtotal, 2 for both

==== Field 5:  Flag active
Type:                   INTEGER
Length:         1 digit
Notes:                  0 for inactive, 1 for active

==== Field 6:  Flag ticket
Type:                   INTEGER
Length:         1 digit
Notes:                  0 no, 1 yes

==== Field 7:  Amount/percentage for discount/markup
Type:                   AMOUNT
Length:         Default
Notes:                  Amount/percentage for discount/markup

==== Field 8:  Maximum price
Type:                   AMOUNT
Length:         Default
Notes:                  The maximum price

==== Field 9:  Maximum percentage
Type:                   AMOUNT
Length:         Default
Notes:                  Maximum percentage

==== Field 10: Daily totals of discount/markup
Type:                   AMOUNT
Length:         Default
Notes:                  Daily total of discount/markup

==== Field 11: Total amount for discount/markup
Type:                   AMOUNT
Length:         Default
Notes:                  Total amount for discount/markup

==== Field 12: Daily Times for discount/markup
Type:                   INTEGER
Length:         DEFAULT
Notes:                  Total number of daily discount/markup

==== Field 13: Total Times for  discount/markup
Type:                   INTEGER
Length:         DEFAULT
Notes:                  Total number of daily discount/markup
```

8.2.38. Command packets in detail -> PLU get info/stats by code [h]

This command will return all information about a programmed PLU.

```
Request packet:
===============
Request code:           'h'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "h/1234567890123456/" (checksum)
Field numbers:          -1----------2--------

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'h' for this command

==== Field 2:  PLU number / PLU search code (2 cases)
Type:                   STRING
Length:         DEFAULT
Notes:                  It is the PLU code to search & reply

Reply packet:
=============
Total field count:      15 (Counting reply code, status & checksum)
Data field count:       11  (Without reply code, status & checksum)
Example reply:          (reply code)(status)
"1231231231231231/ABCDEFGHIJKLMNOPQRST
/0001/12/10100111/500/10.00/10.00/3.100/5.810/150.25/"
 (checksum)

==== Field 1:  PLU search code
Type:                   STRING
Length:         DEFAULT
Notes:                  The search key for this PLU

==== Field 2:  PLU description
Type:                   STRING
Length:         Fixed, 20 characters
Notes:                  The description for this PLU

==== Field 3:  PLU department
Type:                   INTEGER
Length:         Fixed, 4 digits
Notes:                  The department number holding this PLU

==== Field 4:  PLU bonus
Type:                   INTEGER
Length:         DEFAULT
Notes:                  The Plus bonus holding this PLU


==== Field 5:  PLU settings
Type:                   FLAGS
Length:         Fixed, 8 digits
Notes:                  The flag settings for this PLU as (left to right):

                        1 = Item in package
                        1 = Item can have negative price
                        1 = Item has open price
                        1 = Item is available for sales
```

```
                        1 = Item can have a zero price
                        1 = Item will close receipt
                        1 = Print double height
                        1 = Print PLU's department

==== Field 6:  PLU index
Type:                   INTEGER
Length:         1-4 digits,
Notes:                  The index in internal RAM of the PLU in Item database.

==== Field 7:  PLU price
Type:                   AMOUNT
Length:         Default
Notes:                  The price for this PLU

==== Field 8:  PLU maximum price
Type:                   AMOUNT
Length:         Default
Notes:                  The maximum valid price for this PLU

==== Field 9:  PLU current stock
Type:                   QTY
Length:         Default
Notes:                  The current stock available for this PLU

==== Field 10: PLU sold quantity
Type:                   QTY
Length:         Default
Notes:                  The current sold quantity of this PLU

==== Field 11: PLU total sales
Type:                   AMOUNT
Length:         Default
Notes:                  The current PLU total sales
```

8.2.39. Command packets in detail -> Program VAT rates [b]

This command is used to program the VAT rates of the ECR/POS. For this command to succeed, a day must not be open.

```
Request packet:
===============
Request code:            'b'
Total field count:       7 (Counting request code & checksum fields)
Data field count:        5 (Without request code & checksum fields)
Example request:         "b/4/8/18/36/0" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'b' for this command

==== Fields 2,3,4,5,6: Vat rates
Type:                   PERCENTAGE
Length:         0 - 5 digits
Notes:                  The VAT rates to program.
```

```
Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.40. Command packets in detail -> Set receipt comment text [j]

This command is used to set an additional 3-line footnote that will be printed at the end of the receipt just before the programmed footer. This footnote will be active only for the current receipt or the next receipt (if none is currently open) if the last commnad filed is 0, or will be active for all the receipts if the last command filed is 1.

```
Request packet:
===============
Request code:           'j'
Total field count:      6 (Counting request code & checksum fields)
Data field count:       4 (Without request code & checksum fields)
Example request:        "j/COMMENT 1/COMMENT 2/COMMENT 3/0/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'j' for this command

==== Fields 2,3,4:     Comment lines
Type:                   STRING
Length:         0-32 chars
Notes:                  The actual text that will be printed. Not all three lines
                        required.

==== Field 5:  Comments Type
Type:                   Integer
Length:         Fixed, 1 digit
Notes:                  0= Comments will be active only for the current receipt or
the next receipt (if                    none is currently open)
                        1= Comments will be active for all the receipts



Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.41. Command packets in detail -> Payment amount transfer [l]

This command is for transferring an amount from one payment type to another. For

successful completion of this command, the source payment type must contain a sum that is higher or equal to the amount that must be transferred.

```
Request packet:
===============
Request code:           'l' (the letter)
Total field count:      5 (Counting request code & checksum fields)
Data field count:       3 (Without request code & checksum fields)
Example request:        "l/1/2/100.00/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'l' for this command

==== Field 2:  'Source' Payment
Type:                   INTEGER
Length:         1-2 Digits
Notes:                  The payment to transfer the amount from. Payment codes are
                        defined in another paragraph.

==== Field 3:  'Destination' Payment
Type:                   INTEGER
Length:         1-2 Digits
Notes:                  The payment to transfer the amount to.
                        Payment codes are defined in another paragraph.

==== Field 4:  Transfer Amount
Type:                   AMOUNT
Length:         Default
Notes:                  The amount to be transferred from source to destination.


Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.42. Command packets in detail -> Read last fiscal data [i]

This command returns a packet with the the accumulated sums of the fiscal memory .

```
Request packet:
===============
Request code:           'i'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "i/" (checksum)

==== Field 1:  Request code
Type:                   STRING
Length:         Fixed, 1 character
Notes:                  Must be 'i' for this command
```

```
Reply packet:
=============
Total field count:      13 (Counting reply code, status & checksum)
Data field count:       9 (Without reply code, status & checksum)
Example reply:          (reply code)(status)
"200404/161802/2/4/20.00/0.00/0.00/0.00/20.00"       (checksum)

==== Field 1:  Last Z date
Type:                   DATE6
Length:         Default
Notes:                  It is the date of the last Z closure stored
                        in fiscal memory

==== Field 2:  Last Z time
Type:                   TIME
Length:         Default
Notes:                  It is the time of the last Z closure stored in fiscal memory

==== Field 3:  Last Z number
Type:                   INTEGER
Length:         Default
Notes:                  It is the number of the last Z closure stored in fiscal
                        memory

==== Field 4:  Total receipts counter
Type:                   INTEGER
Length:         Default
Notes:                  The progressive number of all legal receipts recorded in the
                        Fiscal Memory

==== Field 5,6,7,8:    Fiscal accumulated totals
Type:                   AMOUNT
Length:         Default
Notes:                  Each first 4 Vat's accumulated sums stored in fiscal memory.

==== Field 9:  Grand Total
Type:                   Amount
Length:         Default
Notes:                  The grand total of all the transactions stored inside the
                        fiscal memory
```

8.2.43. Command packets in detail -> Read VAT rates [e]

This command is used to retrieve the current vat rates programmed into the ECR/POS .

```
Request packet:
===============
Request code:           'e'
Total field count:      2 (Counting request code & checksum fields)
Data field count:       0 (Without request code & checksum fields)
Example request:        "e/" (checksum)

==== Field 1:  Request code
Type:                   STRING
```

```
Length:         Fixed, 1 character
Notes:                  Must be 'e' for this command

Reply packet:
=============
Total field count:      9 (Counting reply code, status & checksum)
Data field count:       5 (Without reply code, status & checksum)
Example reply:          (reply code)(status) "/4/8/18/36/0" (checksum)

==== Fields 1,2,3,4,5: Vat rates
Type:                   PERCENTAGE
Length:         DEFAULT, range 0-100
Notes:                  The VAT rates that are programmed.
```

8.2.44. Command packets in detail -> Read keyboard [B]

Reads the keyboard

```
Request packet:
===============
Request code:           'B'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "B/21/" (checksum)

==== Field 1:
Type:                   STRING
Length:                 Fixed 1 char
Notes:                  Must be 'B' for this command

==== Field 2:  Number
Type:                   INTEGER
Length:         1 - 3 digits
Notes:                  The keyboard code number



Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.45. Command packets in detail -> Write keyboard [G]

Writes the keyboard

```
Request packet:
===============
Request code:           'G'
Total field count:      6 (Counting request code & checksum fields)
Data field count:       4 (Without request code & checksum fields)
Example request:        "G/21/2/1/15/" (checksum)
```

```
==== Field 1:  Request Code
Type:                 STRING
Length:        Fixed 1 character
Notes:                Must be 'G' for this command

==== Field 2:  Number
Type:                 INTEGER
Length:        1 - 3 digits
Notes:                The keyboard code number

==== Field 3:  General functions
Type:                 INTEGER
Length:        Fixed 1 digit
Notes:                General functions ,range 1-7

==== Field 4:  The keyboard level
Type:                 INTEGER
Length:        Fixed 1 digit
Notes:                The keyboard level ,range 0-1

==== Field 5:  The function id
Type:                 INTEGER
Length:        1 - 4 digits
Notes:                The function id

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.46. Command packets in detail -> Set Pos Bmp [Z]

```
  Set Pos BMP

Request packet:
===============
Request code:           'Z'
Total field count:      4 (Counting request code & checksum fields)
Data field count:       2 (Without request code & checksum fields)
Example request:        "Z/1/2/" (checksum)

==== Field 1:  Request Code
Type:                 STRING
Length:        Fixed 1 char
Notes:                Must be 'Z' for this command

==== Field 2:  Setup bitmap top
Type:                 INTEGER
Length:        0 - 1 digits
Notes:                Setup bitmap top

==== Field 3:  Setup bitmap bottom
Type:                 INTEGER
Length:        0 - 1 digits
```

```
Notes:                  Setup bitmap bottom


Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.47. Command packets in detail -> Set Bitmap [$]

```
  Set the Bitmap

Request packet:
===============
Request code:           '$'
Total field count:      7 (Counting request code & checksum fields)
Data field count:       5 (Without request code & checksum fields)
Example request:        "$/1/128/128/15/f0a0fe....." (checksum)

==== Field 1:  Request Code
Type:                   STRING
Length:         Fixed 1 char
Notes:                 Must be '$' for this command

==== Field 2:  Number
Type:                   INTEGER
Length:         Fixed 1 digit
Notes:                  The Bmp index number

==== Field 3:  Number
Type:                   INTEGER
Length:         1- 3 digits
Notes:                  The width

==== Field 4:  Number
Type:                   INTEGER
Length:         1- 3 digits
Notes:                  The height

==== Field 5:  Number
Type:                   INTEGER
Length:         1- 3 digits
Notes:                  The line index

==== Field 6:  Description
Type:                   STRING
Length:         0 - 64 characters
Notes:                  The Bmp data

Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
```

```
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.48. Command packets in detail -> Set Category [R]

Set a Category number and description

```
Request packet:
===============
Request code:           'R'
Total field count:      4 (Counting request code & checksum fields)
Data field count:       2 (Without request code & checksum fields)
Example request:        "R/5/CATEGORY 5" (checksum)

==== Field 1:  Request Code
Type:                   STRING
Length:         Fixed 1 char
Notes:                  Must be 'R' for this command

==== Field 2:  Category Number
Type:                   INTEGER
Length:         DEFAULT
Notes:                  The Category's index number

==== Field 3:  Category Description
Type:                   STRING
Length:         0-20 characters
Notes:                  The Category's description



Reply packet:
=============
Total field count:      4 (Counting reply code, status & checksum)
Data field count:       0 (Without reply code, status & checksum)
Notes:                  This command's reply packet does not contain
                        additional information; only 1 field reply
                        code, 2 fields status and a checksum.
```

8.2.49. Command packets in detail -> Read Category [Q]

Read a Category's number and description

```
Request packet:
===============
Request code:           'Q'
Total field count:      3 (Counting request code & checksum fields)
Data field count:       1 (Without request code & checksum fields)
Example request:        "Q/15" (checksum)

==== Field 1:  Request Code
Type:                   STRING
Length:         Fixed 1 char
Notes:                  Must be 'Q' for this command

==== Field 1:  Number
```

Type: INTEGER
Length: DEFAULT
Notes: Must be 'Q' for this command

Reply packet:
=============
Total field count: 5 (Counting reply code, status & checksum)
Data field count: 1 (Without reply code, status & checksum)
Example reply: (reply code)(status)
"15/CATEGORY 15"   (checksum)

==== Field 1: Description
Type: STRING
Length: Default
Notes: It is the description of the requested category

8.3 Command Protocol Error codes (DECIMAL FORM)

ERR 1	The protocol command expects more fields
ERR 2	A protocol command field is longer than expected
ERR 3	A protocol command filed is smaller than expected
ERR 4	Check the protocol command fields
ERR 5	Check the protocol command fields
ERR 6	The protocol command is not supported
ERR 7	The PLU code doesn't exist
ERR 8	The DPT Code doesn't exist
ERR 9	Wrong VAT code
ERR 10	The Clerk's index number doesn't exist
ERR 11	Wrong Clerk's password
ERR 12	The payment code doesn't exist
ERR 13	The requested Fiscal record doesn't exist
ERR 14	The requested Fiscal record type doesn't exist
ERR 15	Printing type error
ERR 16	The day is open, issue a Z- Report first
ERR 17	Disconnect Jumpers first
ERR 18	Wrong TIME, call SERVICE
ERR 19	NOT USED
ERR 20	A transaction is open, close the transaction first
ERR 21	Invalid Payment
ERR 22	CASH IN/OUT transaction in progress
ERR 23	Wrong VAT rate
ERR 24	Price Error
ERR 25	The online communication of the ECR is ON
ERR 26	The ECR is busy, try again later
ERR 27	Invalid sales operation
ERR 28	Invalid Discount/Markup type
ERR 29	No more headers can be programmed
ERR 30	A user's report is open
ERR 31	A user's report is open
ERR 32	The Fiscal Memory has no transactions
ERR 33	Discount/Markup index number error
ERR 34	You can't program any more PLUs
ERR 35	Error in BMP Data
ERR 36	The BMP index number doesn't exist
ERR 37	The category index number doesn't exist
ERR 38	NOT USED
ERR 39	Error printing type
ERR 40	NOT USED
ERR 41	No more sales can be performed
ERR 42	Keyboard error-or keyboard disconnected
ERR 43	Battery error-or battery low
ERR – 100	.Larger Quantity Value than the one allowed
ERR – 101	Larger Sales Value than the one allowed.
ERR – 102	PLU does not exist.
ERR – 103	DPT does not exist.
ERR – 104	There is an open receipt
ERR – 105	There is an open CASH IN.
ERR – 106	There is an open CASH OUT.
ERR – 107	Negative VAT amount.
ERR – 108	No open receipt.
ERR – 109	No transactions in the receipt .
ERR – 110	Action is not allowed
ERR – 111	The total payment amount must be inserted first

ERR – 112	Negative total is not allowed .
ERR – 113	Subtotal must be inserted first
ERR – 114	Change are not allowed
ERR – 115	There is an open receipt .
ERR – 116	There is an open CASH IN/OUT.
ERR – 117	There is not an open CASH IN.
ERR – 118	There is not an open CASH OUT.
ERR – 119	Wrong payment code.
ERR – 120	Wrong DPT code.
ERR – 121	Ticket's decimal quantity is not allowed.
ERR – 122	No zero Discount/Markup is allowed.
ERR – 123	Discount/Markup limit is exceeded.
ERR – 124	No zero PLU/DPT sale is allowed.
ERR – 125	The DPT/PLU is not active.
ERR – 126	The maximum allowed receipt's total is exceeded.
ERR – 127	The maximum sales total is exceeded.
ERR – 128	Coupon discount in subtotal is not allowed.
ERR – 129	No more discount/markup is allowed, a sale must occur first.
ERR – 130	Wrong DATE-TIME Call Service.
ERR – 131	No negative VAT amount is allowed
ERR – 132	Subtotal must be pressed first before Discount/Markup.
ERR – 133	No Discount/Markup is allowed in Subtotal .
ERR – 134	There are no fiscal data for the requested period .
ERR – 135	The Discount/Markup is not active .
ERR – 136	Fiscal Memory is full
ERR – 137	There are no transactions.
ERR - 138	Clerk is not allowed to perform ths action
ERR – 139	No daily transactions
ERR – 140	Wrong DATE, call SERVICE
ERR – 141	Wrong TIME, call SERVICE
ERR – 142	Fiscal MEMORY disconnected
ERR – 143	There daily transactions, issue a Z report first
ERR – 144	Access only to SERVICE
ERR – 145	Paper End in Journal Station
ERR – 146	Paper End in Receipt Station
ERR – 147	Printer Head open
ERR – 148	Printer Disconnected
ERR – 149	Fiscal Memory Error
ERR – 150	You cannot program any more PLUs cause the PLU index number is exceeded
ERR – 151	You cannot program this Discount/Markup
ERR – 152	The Client Code does not exist
ERR - 153	You cannot program 2 different VAT rates to have the same value
ERR – 154	No sales are allowed after a ticket discount
ERR – 155	No more Headers can be programmed
ERR – 156	The PLUS must be zeroed first
ERR - 157	A Z READ must be issued first
ERR - 158	Inactive Payment