

iINNOVA HIT[®] *Plus*

CASH REGISTER

Programming manual

© INNOVA SA, Warszawa 2004

| | |
|---|----|
| 1. PROGRAMMING THE INNOVA HIT PLUS CASH REGISTER..... | 4 |
| 1. General Rules..... | 4 |
| 2. Databases..... | 6 |
| 2.1. Package Database..... | 7 |
| 2.2. Access Key Database..... | 7 |
| 2.3. Cashier Database..... | 7 |
| 2.4. Form of Payment Database..... | 8 |
| 2.5. Discount (surcharge) Database..... | 8 |
| 2.6. PLU Database..... | 9 |
| 3. INNOVA HIT PLUS CASH REGISTER COMMAND LIST..... | 9 |
| 3.1 Record Entry in the PACKAGE DATABASE..... | 9 |
| 3.2 Record Entry in the ACCESS KEY DATABASE..... | 10 |
| 3.3 Record Entry in the CASHIER DATABASE..... | 10 |
| 3.4 Record Entry in the FORM OF PAYMENT DATABASE..... | 10 |
| 3.5 Record Entry in the DISCOUNT/SURCHARGE DATABASE..... | 10 |
| 3.6 Record Entry in the PLU DATABASE..... | 10 |
| 3.7 Request to Send #Pr Record From the #Pb Database..... | 11 |
| 3.8 Send Totalizer or receipt data Request..... | 11 |
| 3.9 Request to Send Record from the Fiscal Memory..... | 13 |
| 3.10 Send Header Request..... | 15 |
| 3.11 Request to Read and Send the RTC Time..... | 15 |
| 3.12 Send Status (flag) Request..... | 15 |
| 3.13 Request to Send the Current PTU Rates..... | 16 |
| 3.14 Debug Mode Selection..... | 17 |
| 3.15 Record Removal From the PACKAGE DATABASE..... | 17 |
| 3.16 Record removal From the ACCESS KEY DATABASE..... | 17 |
| 3.17 Record Removal From the CASHIER DATABASE..... | 18 |
| 3.18 Record Removal From the PAYMENT FORM DATABASE..... | 18 |
| 3.19 Record Removal From the DISCOUNT/SURCHARGE DATABASE..... | 18 |
| 3.20 Record Removal From the PLU DATABASE..... | 18 |
| 3.21 Setting the Cash-Register Setup Byte..... | 18 |
| 3.22 Header Notation..... | 19 |
| 3.23 RTC Clock Notation..... | 19 |
| 3.24 PTU Rate Notation..... | 20 |
| 3.25 Request to Send #Pr Record From the PLU Database..... | 20 |
| 3.27 Send Identification Request..... | 21 |
| 3.27 End of Communication..... | 22 |
| 4 ON-LINE WORKING MODE..... | 22 |
| 4.1 GENERAL ASSUMPTIONS..... | 22 |
| 4.2.1. Reading a database..... | 23 |
| 4.2.3. Reading inventory and sales from the PLU Pr record..... | 24 |
| 4.2.4 Request to Send #Pr Record From the PLU Database..... | 25 |
| 4.2.5 Send Totalizer or receipt data Request..... | 26 |
| 4.2.6 Request to Send Record from the Fiscal Memory..... | 27 |
| 4.2.7. Setting the Cash-Register Setup Byte..... | 29 |
| 4.4. Messages Automatically Sent From the Cash Register After Each transaction..... | 30 |
| 5. Error List..... | 32 |
| 5.1 Errors That Allow for Continued Operation After Cancelling..... | 32 |
| 5.2 Fatal Errors Reported After the Power is Turned Back On..... | 34 |

5.3 “Read Only” Status.....34

1. PROGRAMMING THE INNOVA HIT PLUS CASH REGISTER

1. General Rules.

The command syntax of the INNOVA HIT PLUS cash-register programming “language” is as follows:

```
ESC P <n1>; <n2>, ..<nk> <char> <b1><b2>..bn> <check> ESC \
```

where:

<n1>; <n2>...<nk> = an optional list of up to 8 numerical parameters, each with the range of 0..65535 (other than e.g. in fiscal printers, where up to 16 parameters with the range of 0..255 can be set), separated with “,”,

<char> = the command ID character for the range '@'... 'Z' (#\$40..#\$5A, also different from fiscal printers, where there are two characters, including one “lower-case letter”),

<b1><b2>...<bn> = a byte string stored in the cash register’s RAM, in the relevant database. The string is DANE BINARNE (BINARY DATA), it is memory cached, followed by a “control byte” check <check> (a database entry is only allowed if the control byte has the correct value).

NOTE: - implementation of the syntax analysis procedure results in the following restriction: for each command, the length of the <b1>...<bn> string is fixed, which means e.g. that there are separate commands for each database record, even though reading is possible using one command for all the databases.

<check> - the control byte coded as two HEX digits (EXOR over all characters after ESC P for that byte with the initial value of = #255), based on the following algorithm (the definition is the same as for fiscal printers):

```
begin
  check := 255;
  for i:= 3 to length(sequence)-4 do
    check := check xor byte(sequence[i]);
end;
```

meaning that, for sequence control, we do not take the first 2 bytes (ESC P) and the last 4 bytes (2 hex digits representing the control byte, and the ESC/ sequence terminator).

NOTE: the HEX value calculated in the loop above should be translated into two HEX digits (ASCII characters from the following set: '0'..'9', 'A'..'F', 'a'..'f').

The length of the <b1><b2>..**bn**> string is fixed and depends on the command. For some commands, it may be = 0, and for others it is the same as the length of the record in the relevant database. The internal data structure in a record **IS NOT CONTROLLED BY THE CASH REGISTER** during remote setup (with a few exceptions, such as that the messages

cannot include control characters, and the BCD data is subject to format limitations. Also, additional checks are made for the PLU record field). The database structure and the different fields are described below (for the current version). Generally, databases support string data or numerical data, and sometimes binary attribute data. In order to make the so-defined interface between the service program and the cash-register databases more resistant to the inevitable changes in the data structure, the possibility of reading all the database and record field structures was provided for. You can read a full attribute-data set (e.g. with “random” access to record readout in the fiscal memory).

NOTE:

The following errors can appear:

“*WRONG CHARACTER*” (error no. 20) – if a character other than a digit, semicolon, space, or upper-case letter is entered in the <n1>..<> sequence (i.e. the following code sets are allowed in this place in the sequence: #30..#39, #3B, #20, #40..#5F),

“*CONTROL BYTE ERROR*” (error no. 21)

“*PARAMETER ERROR*” (error no.22) – e.g. database number >7,

“*DATA ERROR*” (error no. 23) – e.g. a control code in the text field, or a formal BCD data error.

“*COMMAND ERROR*” (error no. 24)– the letter string does not correspond to any actual command,

“*PLU CHANGE BLOCKED*” (error no. 25)– change of the PTU (Polish equivalent of VAT) name / rate or PLU cancellation for non-zero totalizers,

“*PROGRAMMING ERROR*” (error no. 26) – operation not allowed in the cash register mode (for example in the fiscal mode)

“*NON-ZERO TOTALIZERS*” (error no. 27) – operation not allowed because of non-zero totalizers (daily report required)

“*PLU name ERROR*” (error no. 28) – wrong number of characters in the PLU name or PLU name is not unique

“*FISCAL MODE OPERATION ERROR*” (error no. 31) – PC programming not allowed in the fiscal mode

If one of these errors occurs, depending on the operating mode selected, either the error message will appear on the display, or the error code will be sent to the computer, and the sequence will always be ignored to the end, i.e. ESC, or to the beginning of the next sequence, i.e. ESC P.

2. The <n1>..<> parameter string usually has the following form:

Pk; Pr; Pb

Pk = message number (for the command or used for retransmission, =0...255),

Pr = record number,

Pb = database number,

where the Pb parameter only makes sense for database read commands, and the Pk parameter can be ignored (e.g. ESC P 0;1 ...).

3. The hardware layer guarantees the following connection requirements for the RS-232 interface:

- transmission parameters: 9600, 19200 or 38400 baud,8,N,1,
- receiver handshake: XON-XOFF,
- receiver buffer: 256 byte FIFO, XOFF after 192 characters, XON under 128 characters,
- transmitter handshake: none.

4. The data transmitted to the cash register are usually 'binary' (numbers in the BCD code, flag bytes or text strings), and the data sent from the cash register are subject to restrictions due to the potential collision with the XON-XOFF codes, and are usually hexadecimal.

5. Occurrence of ESC while transmitting the <b1><b2>..WRONG CHARACTER (BLEDNY ZNAK) appears.

2. Databases.

INNOVA HIT PLUS cash register supports up to eight databases. All the databases are tables with a fixed (i.e. predetermined in the compilation stage) number of elements with a fixed (i.e. predetermined in the compilation stage) structure.

The fields of seven databases are described in detail below.

The databases are identified by the numerical parameter “database number“, Pb, which can have the following values:

- Pb=0 : package database
- Pb=1 : shortcut key database.
- Pb=2 : cashier database,
- Pb=3 : (forbidden)
- Pb=4 : payment form database,
- Pb=5 : discount / surcharge database,
- Pb=6 : (reserved)
- Pb=7 : PLU database,

The database records are identified by the “record number” parameter Pr=0..N-1, where N is the number of records in the database. The record fields can be as follows:

- A. "string[N]": a text string composed of N – characters + terminator (code #255 in this version),
- B. “amount”, numerical field, (of less-accuracy): 5 bytes, BCD (10 digits), without the format mark reserved for amounts – 2 decimal points (in the compilation phase, you can also select representations without decimal points),
- C. “byte” – a variable to be used with different flags,
- D. “number” – 2-byte BCD value (4 digits), reserved for counting different occurrences (e.g. cancellations, entry reversals, etc.),
- E. special formats, e.g.:
 - "EAN": 7 bytes, BCD (max. 14 digits per barcode),
 - "ilosc_s": the quantity of goods for statistics, 3 bytes, BCD - xxx.xxx

- "nr_plu" : PLU number, 2 bytes, BCD,

The record field types are reported by the cash register after a command to return field definitions as (for instance):

'S18' = "string[18]",

'K5' = "amount, 5 bytes, BCD",

'B1' = "binary data, 1 byte",

'N2' = "number, 2 bytes BCD",

'I6:3' = "quantity, BCD, format xxx.xxx",

'S2' = "special format', 2 bytes"

There is also a special record status – “empty record”, usually identified by the contents of the FIRST field in the record:

- “EMPTY NAME” (i.e. the first byte in the name = #255), database 0,2,4,5,7

- “NO SUCH A PLU EXISTS” (i.e. PLU = 0000), database 1,

2.1. Package Database.

Records = 15

Record length = 40

| Field Name | Type | Bytes | Meaning |
|------------|----------------|-------|--|
| NAME | string[18],S18 | 19 | Name of deposit package – FOR STATISTICS ONLY !+terminator |
| PRICE | amount, K5 | 5 | Package price |
| ACCEPTED | amount, K5 | 5 | The value of accepted packages |
| ISSUED | amount, K5 | 5 | Value of packages issued |
| BALANCE | amount, K5 | 5 | Package amount balance |
| ATTRIBUTE | byte, B1 | 1 | (MSB = “change”) |

2.2. Access Key Database.

Records = 20

Record length = 3

| Field Name | Type | Bytes | Meaning |
|------------|---------------|-------|--|
| PLU | BCD number,N2 | 2 | PLU number, 0: record is empty |
| ATTRIBUTE | byte b1 | 1 | (MSB = "change"), 0..3: default quantity, 4: whole transaction |

2.3. Cashier Database.

Records = 8

Record length = 120

| Field Name | Type | Bytes | Meaning |
|------------|------|-------|---------|
|------------|------|-------|---------|

| | | | |
|------------------------|------------------|---|------------------------------------|
| NAME | string[6],S9 | 6 | 6 name characters + terminator |
| PASSWORD | BCD number,N2 | 2 | password: BCD4 |
| SALES | amount, K5 | 5 | sales |
| ACCEPTED DEPOSITS | amount, K5 | 5 | accepted deposit charges |
| REFUNDED DEPOSITS | amount, K5 | 5 | refunded deposit charges |
| CANCELLATIONS | BCD number,N2 | 2 | number of cancellations, BCD4 |
| CANCELLATION AMOUNT | amount, K5 | 5 | value of cancellations |
| REVERSALS | BCD number,N2 | 2 | number of reverse entries, BCD4 |
| REVERSAL AMOUNT | amount, K5 | 5 | the value of reverse entries |
| DEPOSITS | amount, K5 | 5 | cash deposits to the cash register |
| WITHDRAWALS | amount, K5 | 5 | withdrawals from the cash register |
| FORM OF PAYMENT 1 | amount, K5 | 5 | sales / form of payment 1 |
| FORM OF PAYMENT 2 | amount, K5 | 5 | sales / form of payment 2 |
| FORM OF PAYMENT 3 | amount, K5 | 5 | sales / form of payment 3 |
| FORM OF PAYMENT 4 | amount, K5 | 5 | sales / form of payment 4 |
| TAKINGS | amount, K5 | 5 | Takings |
| DISCOUNT TOTAL1 | amount, K5 | 5 | discount total #1 |
| DISCOUNT TOTAL2 | amount, K5 | 5 | discount total #2 |
| DISCOUNT TOTAL3 | amount, K5 | 5 | discount total #3 |
| DISCOUNT TOTAL4 | amount, K5 | 5 | discount total #4 |
| CHARGES1 | amount, K5 | 5 | value of charges #1 |
| CHARGES2 | amount, K5 | 5 | value of charges #2 |
| CHARGES3 | amount, K5 | 5 | value of charges #3 |
| CHARGES4 | amount, K5 | 5 | value of charges #4 |
| RECEIPTS | BCD number,N2 | 2 | number of receipts, BCD4 |
| LOGIN TIME | BCD4, S2 | 2 | login time BCD4, (min,hr) |
| RESERVE FIELD | BCD4, S2 | 2 | Reserve field, BCD4 |
| ATTRIBUTE | byte B1 | 1 | (MSB = "change" flag) |

2.4. Form of Payment Database.

Records = 8

Record length = 25

| Field Name | Type | Bytes | Meaning |
|------------|----------------|-------|-------------------------------------|
| NAME | string[18],S18 | 19 | Form of payment + terminator |
| SALE | amount, K5 | 5 | Sales value for the form of payment |
| ATTRIBUTE | byte, B1 | 1 | MSB = "change" flag |

2.5. Discount (surcharge) Database.

Records = 8
Record length = 27

| Field Name | Type | Bytes | Meaning |
|------------|----------------|-------|---|
| NAME | string[18],S18 | 19 | Form of payment name + terminator |
| VALUE | BCD4, S2 | 2 | Discount/Surcharge value, BCD4, xxx.x % |
| AMOUNT | amount, K5 | 5 | Discount/surcharge amount |
| ATTRIBUTE | byte, B1 | 1 | MSB = "zmiana" (change) flag LSB=0: percent, =1:amount |

NOTE: - the format of percentage discount/surcharge for the transaction: 0xx.x[%] :
discount, 1xx.x[%] surcharge

2.6. PLU Database

Records = 9600 or 19840 (version dependent)
Record length = 46

| Field Name | Type | Bytes | Meaning |
|----------------|-----------------------|-------|--|
| PLU NAME | string[16],S16 | 16 | name 0..NAME_LENGTH bytes |
| EAN | BCD number, N7 | 7 | barcode 7 bytes: BCD13 |
| PRICE | amount, K5 | 5 | price: amount |
| PLU ATTRIBUTE | byte, B1 | 1 | attribute byte =SSSSXVVV, SSSS = package ref. No., VVV = VAT rate ref. |
| QUANTITY SOLD | quantity BCD, I6:3 | 3 | quantity of goods for quantity/value settlements |
| VALUE SOLD | Amount, K5 | 5 | value of the goods for quantity/value settlements |
| STOCK QUANTITY | Quantity, BCD | 3 | Stock quantity |
| STOCK VALUE | Amount, K5 | 5 | Stock value |
| RESERVE | byte, B1 | 1 | reserve byte, MSB= "change" |

3. INNOVA HIT PLUS CASH REGISTER COMMAND LIST

3.1 Record Entry in the PACKAGE DATABASE

ESC P Pk;Pr @ <b1><b2>..

Where:

Pk = message number,

Pr = record number = 0..14,

<b1>..

3.2 Record Entry in the ACCESS KEY DATABASE

ESC P Pk;Pr A <b1><b2><b3> <check> ESC \

Where:

Pk = message number,

Pr = record number = 0..19

<b1><b2><b3> = 3 bytes record content,

3.3 Record Entry in the CASHIER DATABASE

ESC P Pk;Pr B <b1><b2>..**<b120>** <check> ESC \

Where:

Pk = message number,

Pr = record number = 0..7,

<b1><b2>..**<b120>** = 120 bytes record content,

3.4 Record Entry in the FORM OF PAYMENT DATABASE

ESC P Pk;Pr C <b1><b2>..**<b25>** <check> ESC \

Where:

Pk = message number,

Pr = record number = 0..7,

<b1><b2>..**<b25>** = 25 bytes record content,

3.5 Record Entry in the DISCOUNT/SURCHARGE DATABASE

ESC P Pk;Pr D <b1><b2>..**<b22>** <check> ESC \

Where:

Pk = message number,

Pr = record number = 0..7,

<b1><b2>..**<b22>** = 22 bytes record content,

3.6 Record Entry in the PLU DATABASE

ESC P Pk;Pr E <b1><b2>..**<b46>** <check> ESC \

Where:

Pk = message number,

Pr = record number = 0..9599, or 0..19839 (version dependent)

<b1><b2>..**<b46>** = 46 bytes record content,

3.7 Request to Send #Pr Record From the #Pb Database

ESC P Pk;Pr;Pb H <check> ESC \

Where:

Pb=0 : package database

Pb=1 : shortcut key database,

Pb=2 : cashier database,

Pb=3 : (forbidden)

Pb=4 : payment form database,

Pb=5 : discount / surcharge database,

Pb=6 : (reserved)

Pb=7 : PLU database,

Pr record number (the range of the Pr parameter depends on the selected database).

Response:

ESC P Pk;Pr;Pb h <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

<h1h>..<hNl> is a string of 2N hex encoding N bytes of the selected database record (<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.8 Send Totalizer or receipt data Request

ESC P Pk; Pm I <check> ESC \

Where:

Pm = 0 (or lack of the parameter) – send totalizer request

Pm = 1 – send first FIFO position (the highest) request

Pm = 2 – send next FIFO position request (index pre-increment and transmission)

Pm = 3 – first FIFO position request and index post-increment

-

Response for Pm=0:

ESC P Pk i <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

<h1h>..<hNl> is a string of 2N hex encoding N bytes of the totalizer (<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

The totalizer data organization is as follows:

- in this version, seven registers, 5 bytes each (A..G tax rates),
 - 2 bytes for the receipt ref.,
 - 3 bytes for the number of changes in the database,
 - 2 bytes for the number of cancelled receipts,
 - 5 bytes for the amounts of cancelled receipts,

- 1 byte for the check total,
(48 bytes total)

All the data except for the check total is in BCD format.

The check total is calculated as the binary arithmetic sum (one-byte, i.e. modulo 256 sum), then logically negated (CRC := not(B1+B2+..+BN)).

Response for Pm=1..3 (ticket data)

ESC P Pk; 0; Pkas; Pnpar; Pnp; Pv p <check> ESC \

ESC P Pk+1; 1 p <ticket line 1 data> <check> ESC \

ESC P Pk+2; 2 p <ticket line 2 data> <check> ESC \

...

ESC P Pk+Pnp; Pnp p <ticket line Pnp data> <check> ESC \

NOTE:

If the fifo is empty the following sequence is sent:

ESC P Pk; 0; 0; 0; 0; 0 p <check> ESC \

(i.e. Pnpar = 0 & Pnp = 0)

where:

Pk – sequence number sent in the ESC P Pk; Pm I... sequence

NOTE:

1. the number is incremented (modulo 256) for each position sent
2. for automatic transmission the starting transmission number is set to 0,

Pkas – cashier number

Pnpar – receipt number within a fiscal day (1 .. max)

Pnp – receipt position (line) number (1..max)

Pv = 1 – FIFO overfill occurred

Pv = 0 – OK no data loss occurred

<ticket line n data> - data vector (17 bytes) with the following HEX format:

pp pp ii ii cc cc cc cc cc bb bb bb bb oo aa

where:

pp pp – four digits of the PLU number, BIN (HEX), counted from 0 for 9600 PLU (version dependent)

or

ppppp – five digits of the PLU number, BIN (HEX), counted from 0 for 19840 PLU (version dependent),

ii ii ii – 6 digits of amount. Always format xxx.xxx (real amount = ii ii ii x 0.001, regardless the stock data format

cc cc cc cc cc – PLU price

bb bb bb bb bb – position gross sale

oo – package number joined (0 if no package is associated with a given PLU)

aa – line attribute (2 bits are only relevant)

xxxxxx00 – PLU sale

xxxxxx01 – package sale

xxxxxx10 – package withdrawing

3.9 Request to Send Record from the Fiscal Memory

ESC P Pk;Pr J <check> ESC \

where:

Pr = 0..2047 = the record number in the fiscal memory,

NOTE:

1. the number is not controlled, so any number out of the stated range can bring unexpected results,
2. In this version, the cash register gives access to any place in the fiscal memory, according to the map. The response is always the 64-byte content from the address Pr*64. For example, to read the first notation in the fiscal memory (changes in PTU rates during fiscal processing), send Pr = 8.
3. This solution allows controlling memory fields other than those reserved for sequential reading, i.e. the unique number, the clearance record.

Response:

ESC P Pk;Pr j <h1h><h1l>...<h64h><h64l> <check> ESC \

where:

<h1h>..<hNl> is a string of 128 hex encoding 64 record bytes in the fiscal memory (<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

Memory map:

| | |
|---------------|---|
| 00000..000FF | : control field, content FF,FE...,01,00 |
| 00100..0011F | : device identification area. An 11-digit unique number, 32 bytes at the end check total. The area is recorded in the production process. |
| 00120..0013F | : taxpayer identification and fiscal processing area. 10 NIP digits + separators. The data is recorded in the fiscal processing stage. 32 bytes total + check total |
| 00140..001DF | : reserved |
| 001E0..0011FF | : the area with the record “closing” the data recording stage, and including the date and time of the switch to “read-only”. FORMAT: 00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 yy yy mm dd HH MM FF FF FF FF FF FF FF FF FF 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF |
| 00200..1FF7F | : an area for max. 2038 64-byte records containing different strings. See “Organization of Records” |
| 1FF80..1FFBF | : one record is left free = \$FF |
| 1FFC0..1FFFF | : control field, content 00,01,....,3E,3F |

Organization of Records:

All the data (except for the check total 'ss') is in BCD format. The date format assumes BCD4 representation for the year.

1. Daily report:

```

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
01 yy yy mm dd aa aa aa aa aa bb bb bb bb bb cc cc cc cc cc dd dd dd dd
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
dd ee ee ee ee ee ff ff ff ff ff gg gg gg gg gg vv vv zz zz zz ii ii kk
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
kk kk kk kk FF FF FF FF FF FF FF FF FF FF ss

```

Where:

yyyy,dd,mm - date (NOTE: the year format is such that the byte in the higher position, i.e. byte #2 in the record, is hundreds' byte = \$19 or \$20)

aa..aa,...,gg..gg – GROSS sales amounts per tax group,

nnnn – the number of receipts,

zz – number of modifications for the PLU data base

ii – number of annulments

kk – gross sale for annulments

ss – check total negated for the whole record,

2. Change in PTU rates

```

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
02 yy yy mm dd aa aa bb bb cc cc dd dd ee ee ff ff gg gg FF FF FF FF FF
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ss

```

Where:

yyyy,dd,mm - date

aaaa,...,gggg – new PTU rate values, format: BCD4 = xx.x0 [%], \$FFFF – non-active rate,

\$FEFF – exempted rate

ss – check total negated for the whole record,

3. RAM reset

```

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
03 yy yy mm dd HH MM pp kk qq FF FF FF FF FF FF FF FF FF FF FF FF
24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63

```

FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ss

Where:

yyyy,dd,mm - date

HH,MM - hour: minute of RAM reset,

pp – reset type

kk – reset code

qq – receipt was printed, = 0 when in the EEPROM memory a flag is set denoting that after last RAM reset (daily report), any transaction has been performed

ss – check total negated for the whole record,

3.10 Send Header Request

ESC P Pk K <check> ESC \

Response:

ESC P Pk k <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

<h1h>..<<hNl> is a string of 2N hex encoding N bytes of the header (<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.11 Request to Read and Send the RTC Time

ESC P Pk L <check> ESC \

Response:

ESC P Pk l <sh><sl><mh><ml><hh><hl><dh><dl><mnh><mnl>
<ylh><yll><yhh><yhl><check> ESC \

where:

<sh><sl> - seconds, BCD (<sh> = '0'..'9', higher 'nibble', <sl>=lower)

<mh><ml> - minutes, BCD

<hh><hl> - hours, BCD

<dh><dl> - days, BCD

<mnh><mnl> - months, BCD

<ylh><yll><yhh><yhl> - year, BCD4, <ylh><yll> = lower BCD byte (e.g. '99' or '00')

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.12 Send Status (flag) Request

ESC P Pk N <check> ESC \

-,

Response:

ESC P Pk n <eh><el><fh><fl><ih><il>
 <ylh><yll><yhh><yhl><mnh><mnl><dh><dl> <check> ESC \

where:

<eh><el> - debug mode,

'00' : (default), display on, message disabled by RS,

'01' : display off, message disabled by RS,

'02' : display on, message enabled by RS,

'03' : display off, message enabled by RS,

<fh><fl> - fiscal mode / training mode

'00' : training mode,

'01' : fiscal mode,

<ih><il> - 6 bits of device setup:

Pm0=0(default), entering in PLN,

Pm0=1 entering in grosz,

Pm1=0(default), automatic lock for transaction data transfers

Pm1=1 data is automatically transferred after the transaction

Pm2=0(default) OFF-LINE working mode

Pm2=1 ON-LINE working mode

<ylh><yll><yhh><yhl><mnh><mnl><dh><dl> -

date of last record in the fiscal memory,

<check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

NOTE: - bytes are transmitted as far as terminator #255, the terminator itself is not transmitted in special cases, if the header is not defined then the transferred string is empty

3.13 Request to Send the Current PTU Rates

ESC P Pk O <check> ESC \

Response:

ESC P Pk o <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

<h1h>..<<hNl> is a string of 2N hex encoding IL_STAWEK* 4 bytes of the PTU rate vector (
 <xxh> is the "higher", and <xxl> the "lower" digit),

<check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P).

The organization of the PTU rate vector data is as follows:

- in this version, 7 rates 2 bytes each (A..G tax rates), the lower byte represents the hundredths of a percent, and the higher byte represents the hundredths of the floor function, with the tax rate in this version calculable with up to 0.1% accuracy (even if HIT sends back for example 28 14, i.e. 14.28%, all calculations will use the 14.2% value, that is, the rightmost digit will be TRUNCATED)

- for non-active rates, the FF FF value will be sent back – which is "incorrect" for the BCD representation,

- for the exempted rate the value of FE FF will be send

3.14 Debug Mode Selection

ESC P Pk;Pm P <check> ESC \

Where:

Pm=0 : (default), display on, message disabled through RS,
Pm=1 : display off, message disabled through RS,
Pm=2 : display on, message enabled through RS,
Pm=3 : display off, message enabled through RS,

Message sent through the RS (after each command):

ESC P PkPe x <check> ESC \

where:

Pk = message number,
Pe = error number, 0 : OK,
<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),
x = command code, UPPER CASE LETTER!

NOTE: - the next 6 commands remove records from the databases, in that they record the default ROM data, which will be ignored by the other application procedures, e.g. for record selection.

3.15 Record Removal From the PACKAGE DATABASE

ESC P Pk;Pr Q <check> ESC \

Where:

Pk = message number,
Pr = record number = 0..14,
<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.16 Record removal From the ACCESS KEY DATABASE

ESC P Pk;Pr R <check> ESC \

Where:

Pk = message number,
Pr = record number = 0..9 (or 19)
<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.17 Record Removal From the CASHIER DATABASE

ESC P Pk;Pr S <check> ESC \

Where:

Pk = message number,

Pr = record number = 0..7,

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.18 Record Removal From the PAYMENT FORM DATABASE

ESC P Pk;Pr T <check> ESC \

Where:

Pk = message number,

Pr = record number = 0..7,

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.19 Record Removal From the DISCOUNT/SURCHARGE DATABASE

ESC P Pk;Pr U <b1><b2>..

Where:

Pk = message number,

Pr = record number = 0..7,

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.20 Record Removal From the PLU DATABASE

ESC P Pk;Pr V <check> ESC \

Where:

Pk = message number,

Pr = record number = 0..9599, or 0..19839 (version dependent)

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.21 Setting the Cash-Register Setup Byte

ESC P Pk;Pm W <check> ESC \

Where:

Pm=0..63, with the different bits meaning:

Pm.0=0 (default), entering in PLN,

Pm.0=1 entering in grosz,

Pm.1=0 (default), automatic lock for transaction data transfers

Pm.1=1 data is automatically transferred after the transaction

Pm.2,Pm.3=0 or 2 (default) medium printing energy

Pm.2, Pm.3=1 decreased printing energy

Pm.2,Pm.3 = 3 increased printing energy

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.22 Header Notation

ESC P Pk X <b1><b2>..

Where:

Pk = message number,

<b1><b2>..

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

NOTE:

1. always send 161 bytes, or shorten the header by putting in the terminator “earlier”,
2. the only control code allowed is CR (#13, end of line)
3. the sequence does not work in the fiscal mode

3.23 RTC Clock Notation

ESC P Pk Y <s><mm><h><d><mn><yl><yh><check> ESC \

Where:

Pk = message number,

where:

<s> = seconds, BCD

<m> = minutes, BCD

<h> = hours, BCD

<d> = days, BCD

<mn> = months, BCD

<yl><yh> = year, BCD4, <yl> = lower BCD byte (e.g. '99' or '00')

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

NOTE:

The sequence does not work in the fiscal mode

3.24 PTU Rate Notation

ESC P Pk Z <b1>...<b14> ESC \

Where:

Pk = message number,

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

NOTE:

Send $2 * (\text{ilosc_stawek}) = 14$ bytes in the BCD format, for an inactive rate send #255 #255

The sequence does not work in the fiscal mode

3.25 Request to Send #Pr Record From the PLU Database

ESC P Pk;Pr] <check> ESC \

Where:

Pk – message number

Pr – record number = 0..MAX_PLU_NUM – 1 (9599 or 19839 version dependent)

-

Response:

ESC P Pk;Pr0 } <h1h><h1l>...<hNh><hNl> <check> ESC

where:

<h1h>..<hNl> is a string of 2N hex encoding N bytes of the selected PLU database record (<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

or:

ESC P Pk;Pr1 } <check> ESC \

if the record is EMPTY,

or:

ESC P Pk;Pr2 } <check> ESC \

data read error (e.g. wrong record number),

A sequence allowing for sending the next PLU record which was marked during transaction by bit no. 5 of the PLU record attribute:

ESC P Pk; Pm [<check> ESC \

Where:

Pm=0 (or lack of this parameter) – the non-empty record is send

Pm<0 the next non-empty record for which for which a transaction has been performed since the last computer transmission

Response:

ESC P Pk;Pr;0 { <h1h><h1l>..<>hNh><hNl> <check> ESC \

Or:

ESC P Pk; plu_max-1;1 { <check> ESC \

If the non-empty record do not exist.

Where:

plu_max is the maximum value of PLU number for a given cash register version.

NOTE:

1. Using the ESC P Pk;Pr]... sequence, the first PLU number from which a transfer is started is set and the cash register remembers this information.
2. Entering the computer communication mode, the PLU number is automatically reset and in response for the ESC P Pk [...] sequence the data of the second PLU record will be obtained,
3. Each transmission of the PLU record data resets the attribute bit “there was a sale”.

3.26 Send the cash register manager password

ESC P Pk \ <check> ESC \

Response:

ESC P Pk | <b1><b2><check> ESC \

Where:

Pk – message number

<b1><b2> - BCD data of the password

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

3.27 Send Identification Request

ESC P Pk ^ <check> ESC \

-

Response:

ESC P Pk ~ <string> <check> ESC \

Where:

<string> has the format:

<device name> / vv.m / nnnnnn /rx

where:

'vv' = version number,

'm' = model number,

'nnnnnn' = manufacturer's number.

x – hardware version symbol (= 3 for 9600 PLU or =5 for 19840 PLU)

3.27 End of Communication

ESC P _ <check> ESC \

Where:

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

4 ON-LINE WORKING MODE

4.1 GENERAL ASSUMPTIONS

On line mode for the INNOVA HIT PLUS cash register is based on the successive calling of a corresponding process in the main loop of the device software. Using this mode is possible after setting the specified bit in the cash configuration. From this moment on the cash register program allows for performing a number of operations in any stage of the its functioning. In the on-line mode a subset of the off-line commands is allowed and moreover, the process may automatically send messages on completing transaction (sale). On each correctly formulated command the on-line mode processor sends a sequence consisting of the following parts:

- the tested record data (or its part data)
- additional information containing the error code (regardless the debug mode option) with the following format:
ESC P Pk;Pe X <check> ESC \
Where:
Pk – message number
Pe – error code (0 if no error occurred)
X – command coce (capital letter)
<check> - two digits HEX corresponding the check sum

The operations to be performed in the on-line cash register mode may be divided into the following groups:

1. Sending the record data from any database,
2. Sending stock information from the PLU database
3. Sending the next programmed PLU record from the PLU database
4. Sending the totalizers information or the data of one of 15 recently performed receipts (sales)
5. Sending information from the cash register fiscal memory
6. Modifying the cash register configuration
7. Programming new PLU and package records

4.2 Command List

4.2.1. Reading a database

ESC P Pk;Pr;Pb H <check> ESC \

- request to send #Pr record from the #Pb database
where:

Pk = message number,

Pr = record number = 0..

Pb=0 : package database,

Pb=1 : shortcut key database,

Pb=2 : cashier database,

Pb=3 : (forbidden)

Pb=4 : payment form database,

Pb=5 : discount / surcharge database,

Pb=6 : (reserved)

Pb=7 : PLU database,

Pr - the range of the Pr parameter depends on the selected database.

Response:

ESC P Pk;Pr;Pb h <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

<h1h>..<<hNl> is a string of 2N hex encoding N bytes of the selected database record (<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> = 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

4.2.2. Read a PLU database record

ESC P Pk;Pr J <check> ESC \

- request to send #Pr record from the PLU database

where:

Pk = message number,

Pr = record number = 0..MAX_PLU_NUM-1, (9599 or 19839 – version dependent)

Response:

ESC P Pk;Pr0 } <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

<h1h>..<<hNl> is a string of 2N hex encoding N bytes of the selected PLU database record (<xxh> is the “higher”, and <xxl> the “lower” digit),

or:

ESC P Pk;Pr1 } <check> ESC \

- if the record is EMPTY,

-

or:

ESC P Pk;Pr2 } <check> ESC \

- data read error (e.g. wrong record number),

<check> = 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

4.2.3. Reading inventory and sales from the PLU Pr record

ESC P Pk;Pr M <check> ESC \

request to send a fragment of the #Pr record from the PLU database

Response:

ESC P Pk;Pr0 m <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

<h1h>..<hNl> is a string of 2N hex encoding N bytes of the selected PLU database record (
<xxh> is the “higher”, and <xxl> the “lower” digit),

or:

ESC P Pk;Pr1 m <check> ESC \

if the record is EMPTY,

-

or:

ESC P Pk;Pr2 m <check> ESC \

data read error (e.g. wrong record number),

-

<check> = 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

NOTE:

The response content has 32 bytes corresponding to the following fragment of the plu record:

| Field Name | Type | Bytes | Meaning |
|----------------|----------------------|-------|---|
| QUANTITY SOLD | quantity BCD,I6:3 | 3 | quantity of goods for quantity settlements - SOLD |
| SALES VALUE | amount, K5 | 5 | value of the goods for quantity settlements - SOLD |
| STOCK QUANTITY | quantity BCD,I6:3 | 3 | quantity of goods in the “WAREHOUSE” |

| | | | |
|-----------------|------------|---|--------------------------------------|
| WAREHOUSE VALUE | amount, K5 | 5 | value of goods in the “WAREHOUSE” |
|-----------------|------------|---|--------------------------------------|

4.2.4 Request to Send #Pr Record From the PLU Database

ESC P Pk;Pr] <check> ESC \

Where:

Pk – message number

Pr – record number = 0..MAX_PLU_NUM – 1 (9599 or 19839 version dependent)

-

Response:

ESC P Pk;Pr0 } <h1h><h1l>...<hNh><hNl> <check> ESC

where:

<h1h>..<hNl> is a string of 2N hex encoding N bytes of the selected PLU database record (

<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

or:

ESC P Pk;Pr1 } <check> ESC \

if the record is EMPTY,

or:

ESC P Pk;Pr2 } <check> ESC \

data read error (e.g. wrong record number),

A sequence allowing for sending the next PLU record which was marked during transaction by bit no. 5 of the PLU record attribute:

ESC P Pk; Pm [<check> ESC \

Where:

Pm=0 (or lack of this parameter) – the non-empty record is send

Pm<>0 the next non-empty record for which for which a transaction has been performed since the last computer transmission

Response:

ESC P Pk;Pr;0 { <h1h><h1l>..<hNh><hNl> <check> ESC \

Or:

ESC P Pk; plu_max-1;1 { <check> ESC \

If the non-empty record do not exist.

Where:

plu_max is the maximum value of PLU number for a given cash register version.

NOTE:

4. Using the ESC P Pk;Pr]... sequence, the first PLU number from which a transfer is started is set and the cash register remembers this information.
5. Entering the computer communication mode, the PLU number is automatically reset and in response for the ESC P Pk [...] sequence the data of the second PLU record will be obtained,
6. Each transmission of the PLU record data resets the attribute bit “there was a sale”.

4.2.5 Send Totalizer or receipt data Request

ESC P Pk; Pm I <check> ESC \

Where:

Pm = 0 (or lack of the parameter) – send totalizer request

Pm = 1 – send first FIFO position (the highest) request

Pm = 2 – send next FIFO position request (index pre-increment and transmission)

Pm = 3 – first FIFO position request and index post-increment

-

Response for Pm=0:

ESC P Pk i <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

<h1h>..<hNl> is a string of 2N hex encoding N bytes of the totalizer (<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

The totalizer data organization is as follows:

- in this version, seven registers, 5 bytes each (A..G tax rates),

- 2 bytes for the receipt ref.,
- 3 bytes for the number of changes in the database,
- 2 bytes for the number of cancelled receipts,
- 5 bytes for the amounts of cancelled receipts,
- 1 byte for the check total,

(48 bytes total)

All the data except for the check total is in BCD format.

The check total is calculated as the binary arithmetic sum (one-byte, i.e. modulo 256 sum), then logically negated (CRC := not(B1+B2+..+BN)).

Response for Pm=1..3 (ticket data)

ESC P Pk; 0; Pkas; Pnpar; Pnp; Pv p <check> ESC \

ESC P Pk+1; 1 p <ticket line 1 data> <check> ESC \

ESC P Pk+2; 2 p <ticket line 2 data> <check> ESC \

...

ESC P Pk+Pnp; Pnp p <ticket line Pnp data> <check> ESC \

NOTE:

If the fifo is empty the following sequence is sent:

ESC P Pk; 0; 0; 0; 0; 0 p <check> ESC \

(i.e. Pnpar = 0 & Pnp = 0)

where:

Pk – sequence number sent in the ESC P Pk; Pm I... sequence

NOTE:

3. the number is incremented (modulo 256) for each position sent
4. for automatic transmission the starting transmission number is set to 0,

Pkas – cashier number

Pnpar – receipt number within a fiscal day (1 .. max)

Pnp – receipt position (line) number (1..max)

Pv = 1 – FIFO overflow occurred

Pv = 0 – OK no data loss occurred

<ticket line n data> - data vector (17 bytes) with the following HEX format:

pp pp ii ii ii cc cc cc cc cc bb bb bb bb oo aa

where:

pppp – for digits of the PLU number, BIN (HEX), counted from 0 for 9600 version dependent or

ppppp - five digits of the PLU number, BIN (HEX), counted from 0 for 19840 version dependent

ii ii ii – 6 digits of amount. Always format xxx.xxx (real amount = ii ii ii x 0.001, regardless the stock data format

cc cc cc cc cc – PLU price

bb bb bb bb bb – position gross sale

oo – package number joined (0 if no package is associated with a given PLU)

aa – line attribute (2 bits are only relevant)

xxxxxx00 – PLU sale

xxxxxx01 – package sale

xxxxxx10 – package withdrawing

4.2.6 Request to Send Record from the Fiscal Memory

ESC P Pk;Pr J <check> ESC \

where:

Pr = 0..2047 = the record number in the fiscal memory,

NOTE:

4. the number is not controlled, so any number out of the stated range can bring unexpected results,
5. In this version, the cash register gives access to any place in the fiscal memory, according to the map. The response is always the 64-byte content from the address Pr*64. For example, to read the first notation in the fiscal memory (changes in PTU rates during fiscal processing), send Pr = 8.

6. This solution allows controlling memory fields other than those reserved for sequential reading, i.e. the unique number, the clearance record.

Response:

ESC P Pk;Pr j <h1h><h1l>...<h64h><h64l> <check> ESC \

where:

<h1h>..<hNl> is a string of 128 hex encoding 64 record bytes in the fiscal memory (<xxh> is the “higher”, and <xxl> the “lower” digit),

<check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

Memory map:

00000..000FF : control field, content FF,FE...,01,00

00100..0011F : device identification area. An 11-digit unique number, 32 bytes at the end check total. The area is recorded in the production process.

00120..0013F : taxpayer identification and fiscal processing area. 10 NIP digits + separators. The data is recorded in the fiscal processing stage. 32 bytes total + check total

00140..001DF : reserved

001E0..0011FF : the area with the record “closing” the data recording stage, and including the date and time of the switch to “read-only”.

FORMAT:

| | | | | | | | | | | | | | | | |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 |
| yy | yy | mm | dd | HH | MM | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF |
| 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
| FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF | FF |

00200..1FF7F : an area for max. 2038 64-byte records containing different strings. See “Organization of Records”

1FF80..1FFBF : one record is left free = \$FF

1FFC0..1FFFF : control field, content 00,01,....,3E,3F

Organization of Records:

All the data (except for the check total ‘ss’) is in BCD format. The date format assumes BCD4 representation for the year.

4. Daily report:

| | | | | | | | | | | | | | | | | | | | | | | | |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| 00 | 01 | 02 | 03 | 04 | 05 | 06 | 07 | 08 | 09 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 |
| 01 | yy | yy | mm | dd | aa | aa | aa | aa | aa | bb | bb | bb | bb | bb | cc | cc | cc | cc | cc | dd | dd | dd | dd |
| 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 |
| dd | ee | ee | ee | ee | ee | ff | ff | ff | ff | ff | gg | gg | gg | gg | gg | vv | vv | zz | zz | zz | ii | ii | kk |

48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
kk kk kk kk FF FF FF FF FF FF FF FF FF FF ss

Where:

yyyy,dd,mm - date (NOTE: the year format is such that the byte in the higher position, i.e. byte #2 in the record, is hundreds' byte = \$19 or \$20)
 aa..aa,....,gg..gg – GROSS sales amounts per tax group,
 nnnn – the number of receipts,
 zz – number of modifications for the PLU data base
 ii – number of annulments
 kk – gross sale for annulments
 ss – check total negated for the whole record,

5. Change in PTU rates

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
02 yy yy mm dd aa aa bb bb cc cc dd dd ee ee ff ff gg gg FF FF FF FF FF
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
FF FF
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ss

Where:

yyyy,dd,mm - date
 aaaa,....,gggg – new PTU rate values, format: BCD4 = xx.x0 [%], \$FFFF – non-active rate,
 \$FEFF – exempted rate
 ss – check total negated for the whole record,

6. RAM reset

00 01 02 03 04 05 06 07 08 09 10 11 12 13 14 15 16 17 18 19 20 21 22 23
03 yy yy mm dd HH MM pp kk qq FF FF FF FF FF FF FF FF FF FF FF FF FF
 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
FF FF
 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ss

Where:

yyyy,dd,mm - date
 HH,MM - hour: minute of RAM reset,
 pp – reset type
 kk – reset code
 qq – receipt was printed, = 0 when in the EEPROM memory a flag is set denoting that after last RAM reset (daily report), any transaction has been performed
 ss – check total negated for the whole record,

4.2.7. Setting the Cash-Register Setup Byte

ESC P Pk;Pm W <check> ESC \

Where:

Pm=0..63, with the different bits meaning:

Pm.0=0 (default), entering in PLN,

Pm.0=1 entering in grosz,

Pm.1=0 (default), automatic lock for transaction data transfers

Pm.1=1 data is automatically transferred after the transaction

Pm.2,Pm.3=0 or 2 (default) medium printing energy

Pm.2, Pm.3=1 decreased printing energy

Pm.2,Pm.3 = 3 increased printing energy

<check> - 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

4.3 Programming in the ON-LINE MODE

4.3.1 Record entry in the PLU database

ESC P Pk;Pr E <b1><b2>..**<b46><check>** ESC \

Where:

Pk – message number

Pr – record number (0..9599 or 0..19839 version dependent)

<b1>..**<b46>** - 46 bytes of record content

4.3.2 Record Entry in the PACKAGE DATABASE

ESC P Pk;Pr @ <b1><b2>..**<b40><check>** ESC \

Where:

Pk – message number

Pr – record number (0..14)

<b1>..**<b40>** - 40 bytes record content

4.4. Messages Automatically Sent From the Cash Register After Each transaction.

INNOVA Hit Plus cash registers is equipped with two modes of sending receipts data:

1. automatic
2. on request

In either mode, the receipt acceptance causes writing the receipt information into the 15 level (31 for 19840 PLU version) FIFO.

NOTE:

If the automatic mode is set when the FIFO is partly or totally filled, than its content will be immediately send (up to 15 receipts).

In the “on request” mode the data is send due to the following sequence:

ESC P Pk;Pm I <check> ESC \

Where:

- Pm = 0 (or lack of the parameter) – send totalizer request
- Pm = 1 – send first FIFO position (the highest) request
- Pm = 2 – send next FIFO position request (index pre-increment and transmission)
- Pm = 3 – first FIFO position request and index post-increment

NOTE:

Such a concept of sequence enables the receipt retransmission in the case of communication error. Firstly, the sequence with Pm=1 should be send, and if it is properly accepted than the sequence with Pm=2 will follow. In the case of an error occurrence, either a retransmission with Pm=1 or request for the next receipt (Pm=2) may be performed.

Response for Pm=0:

ESC P Pk i <h1h><h1l>...<hNh><hNl> <check> ESC \

where:

- <h1h>..<<hNl> is a string of 2N hex encoding N bytes of the totalizer (<xxh> is the “higher”, and <xxl> the “lower” digit),
- <check> 2 HEX digits representing the control byte as usual, i.e. XOR with the initial value of #255, starting from the 3rd character (after ESC P),

The totalizer data organization is as follows:

- in this version, seven registers, 5 bytes each (A..G tax rates),
 - 2 bytes for the receipt ref.,
 - 3 bytes for the number of changes in the database,
 - 2 bytes for the number of cancelled receipts,
 - 5 bytes for the amounts of cancelled receipts,
 - 1 byte for the check total,
- (48 bytes total)

All the data except for the check total is in BCD format.

The check total is calculated as the binary arithmetic sum (one-byte, i.e. modulo 256 sum), then logically negated (CRC := not(B1+B2+..+BN)).

Response for Pm=1..3 (ticket data)

ESC P Pk; 0; Pkas; Pnpar; Pnp; Pv p <check> ESC \
 ESC P Pk+1; 1 p <ticket line 1 data> <check> ESC \
 ESC P Pk+2; 2 p <ticket line 2 data> <check> ESC \
 ...
 ESC P Pk+Pnp; Pnp p <ticket line Pnp data> <check> ESC \

NOTE:

If the fifo is empty the following sequence is sent:

ESC P Pk; 0; 0; 0; 0; 0 p <check> ESC \

(i.e. Pnpar = 0 & Pnp = 0)

where:

Pk – sequence number sent in the ESC P Pk; Pm I... sequence

NOTE:

5. the number is incremented (modulo 256) for each position sent
6. for automatic transmission the starting transmission number is set to 0,

Pkas – cashier number

Pnpar – receipt number within a fiscal day (1 .. max)

Pnp – receipt position (line) number (1..max)

Pv = 1 – FIFO overfill occurred

Pv = 0 – OK no data loss occurred

<ticket line n data> - data vector (17 bytes) with the following HEX format:

pp pp ii ii ii cc cc cc cc cc bb bb bb bb oo aa

where:

pppp – for digits of the PLU number, BIN (HEX), counted from 0 for 9600 version dependent or

ppppp - five digits of the PLU number, BIN (HEX), counted from 0 for 19840 version dependent

ii ii ii – 6 digits of amount. Always format xxx.xxx (real amount = ii ii ii x 0.001, regardless the stock data format

cc cc cc cc cc – PLU price

bb bb bb bb bb – position gross sale

oo – package number joined (0 if no package is associated with a given PLU)

aa – line attribute (2 bits are only relevant)

xxxxxx00 – PLU sale

xxxxxx01 – package sale

xxxxxx10 – package withdrawing

5. Error List

There are the following types of errors:

- errors 1 to 19 are due to incorrect input values. They are automatically cancelled after 5 s, or by pressing the “C” key.
- errors 20 do 31 are due to mistakes in programming from the computer. They are automatically cancelled after 2 s.
- errors 101 do 117 are warnings about errors in the fiscal processing module. They can be cancelled by pressing “C”
- errors 200 to 219 are fatal errors. If one of those errors occurs, “C” will remain the only active key, used to turn off the cash register.
- errors 241 to 247 read-only. They can be cancelled by pressing “C”, and the cash register will switch to a special “Read-Only” menu see page “Operating manual”

5.1 Errors That Allow for Continued Operation After Cancelling,

After an error occurs, the cash-register display shows an `ERROR` message with the error number or letter symbol.

The table below shows the list of errors that can appear on the cash-register display:

| Symbol (No.) | Description |
|---|--|
| 1 | The entered number is too high |
| 2 | Error in programming the PTU rates – e.g. two identical rates were programmed or the maximum number of records was exceeded (30) |
| 3 | Error in the periodical report dates (e.g. the end date is earlier than the start date). |
| 4 | No cashier uses the entered password |
| 5 | No such PLU number |
| 6 | No such packaging |
| 7 | Receipt buffer overflow (too many items) |
| 8 | Maximum transaction amount exceeded |
| 9 | Totalizer overflow |
| 10 | No such discount |
| 11 | No such form of payment |
| 12 | Attempted sale at an inactive PTU rate |
| 13 | Attempted sale at zero GROSS amount |
| 14 | Attempted sale at zero deposit charge |
| 15 | The periodic report cannot be generated, no records in the memory |
| 16 | Activation of fiscal memory (fiscalization) impossible – fiscal device |
| 18 | Attempt at generation of a monthly report for an “unclosed” month |
| 19 | No service seal (for fiscalization) |
| Errors in programming from the computer | |
| 20 | Wrong character |
| 21 | Control byte error |
| 22 | Wrong parameter |
| 23 | Data error, e.g. wrong BCD format |
| 24 | Wrong command flag |
| 25 | PLU programming error – operation not allowed due to non-zero totalizers |
| 26 | Programming error – the operation is not allowed in the current mode (e.g. fiscal mode) |
| 27 | Operation not allowed – non-zero totalizers |
| 28 | Non-unique product name |
| 29 | Time change by more than +/- 1 hour or attempt at another time change |
| 30 | Attempt at entering the serial number and PUK for a cash register which already has those numbers entered |
| 31 | Programming operation not allowed in the fiscal mode |
| Warnings about errors in the fiscal processing module | |
| 104 | Printing error |

| | |
|-----|--|
| 105 | Totalizer overflow: transaction cannot be completed (the receipt can either be cancelled automatically, or by the cashier) |
| 107 | Entry error in the fiscal module – wrong date format – e.g. if the date is earlier than that of the previous entry, the date of the last entry is automatically copied |
| 108 | The cash register is in fiscal mode - certain operations are blocked |
| 109 | Wrong date – the date is earlier than the date of last record in the fiscal mode |
| 111 | Attempted sale with the service seal on |
| 112 | Low memory warning |
| 113 | The RTC clock is not set |
| 114 | No PTU rates – only in non-fiscal modes |
| 116 | No header |
| 117 | EEPROM not initiated: this error is reported in case of attempted entry into sale if the manufacturer’s code and PUK have not been entered in the cash-register’s EEPROM |

NOTE :in case of errors: 101,104,107 call technical service!

5.2 Fatal Errors Reported After the Power is Turned Back On

(the errors block the cash register, service has to be called!)

| | |
|-----|--|
| 201 | ROM check total error |
| 203 | Error in communication with RTC |
| 207 | Fatal error in the fiscal memory (no check fields or the record check total) |
| 208 | Incorrect initiation of the fiscal module (no PTU rates or no unique number) |
| 215 | Printer mechanism overheated |
| 217 | Error in the fiscal memory entry |
| 218 | Wrong fiscal memory |

5.3 “Read Only” Status

(in case of read-only errors, the cash register switches to a special menu)

| | |
|-----------|--------------------|
| 240 - 247 | “Read Only” status |
|-----------|--------------------|