HANDHELD READER HHR 3000 PRO V2

HHR 3000 PRO
Programmers Manual

CONTENTS

Compliance with FCC Rules and Regulations

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:
(1) this device may not cause harmful interference, and
(2) this device must accept any interference received, including interference that may cause undesired operation.

Changes or modifications to the equipment not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

NOTE: This equipment has been tested and found to comply with the limits for a Class B digital device pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installations. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on , the user is encouraged to try to correct the interference by one or more of the following measures:

- Reorient or relocate the receiving antenna
- Increase the separation between the equipment and receiver
- Connect the equipment into an outlet on a circuit different from that to which the receiver is connected
- Consult the dealer or an experienced radio/TV technician for help.

The reader must be kept 20cm away from the persons body when the GPRS is active.

HHR 3000 PRO v2 can work with following antennas:

- 50007 Loose exchangeable or fixed long antenna, length 60 cm
- 50008 Loose exchangeable or fixed short antena, length 8,5 cm
- 50009 Loose exchangeable medium antenna, length 34 cm
- 50010 Loose exchangeable XXL antenna, length 115cm

DESCRIPTION

HHR 3000 PRO enables customization functionality by writing user Application Project. It contains definition of:

- Database (table with animal data) stored in HHR memory
- HHR menu items,
- HHR screens which would operate on user database.

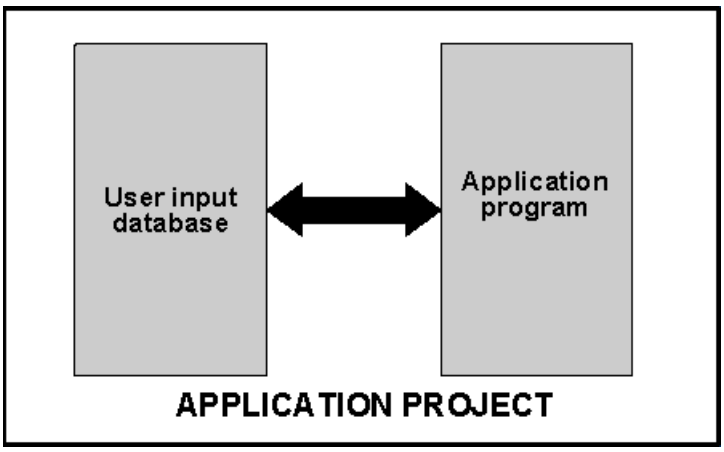
HHR Application Manager PC program enables application project transfer between HHR and PC.

How it works

The Application Development System contains:

- HHR 3000 PRO reader
- Application Project
- HHR Application Manager PC program,

Application Project is based on Application Program written in simple HDS (Hhr Development Script) language and user input database. Those 2 files combined (compiled) by HHR Application Manager are Application Project.



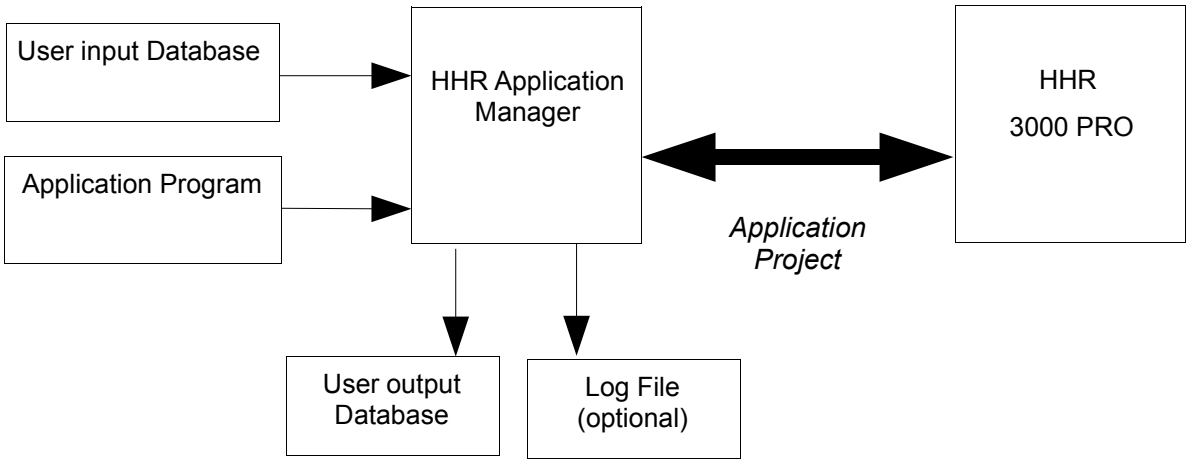
The core of Application Program are macros defining LCD screens. Application Program includes 8 sections – see chapter Sections.

HHR Application Manager is a PC program which combines Application Program and User Database to Application Project. It is used also to transfer data between HHR and PC.

Database definition in Application Program must match User database. It can be empty at start, it is not obligatory to include it to Application Project.

Database can be filled during work with animals.

The structure of Application Development System.



User output Database is a file containing User input database modified during work with HHR 3000 PRO.

Input and output database files are text files with columns isolated by separator defined in Application Program. Database programs (MS Access, MS Excel, Calc, Base, etc.) enables to import text files with separators and convert it to workable table.

There is a possibility to include **Log table** to Application Project. Log is another table which records are created during user-defined events (birth, death).

The structure of Log is defined in Application Program. Log is loaded from HHR to PC together with modified database and will appear as a database text file.

Log example:

VID	Event name	Event value	Date of event
1123456	Treat	Med_1	2007/02/12; 8:00am

GETTING STARTED

Installing the HHR Application Manager software

- Insert the HHR software CD in the appropriate drive. The CD will auto-run.
- Double-click on the folder 'HHR Application Manager' and when this opens double-click on the file 'Install HHR manager.exe' to install HHR on your computer. Follow the on-screen instructions
- The program can now be found under C:\Program Files\BioControl\HHR Application Manager.

Installing the Handheld Reader USB drivers

- Before proceeding ensure that the HHR battery has been charged and that the HHR software is properly installed on your computer.
- Remove the rubber dust cover from the serial port in the base of the HHR and connect the combined data/battery charger cable.
- Connect the data cable to a USB port in your computer.

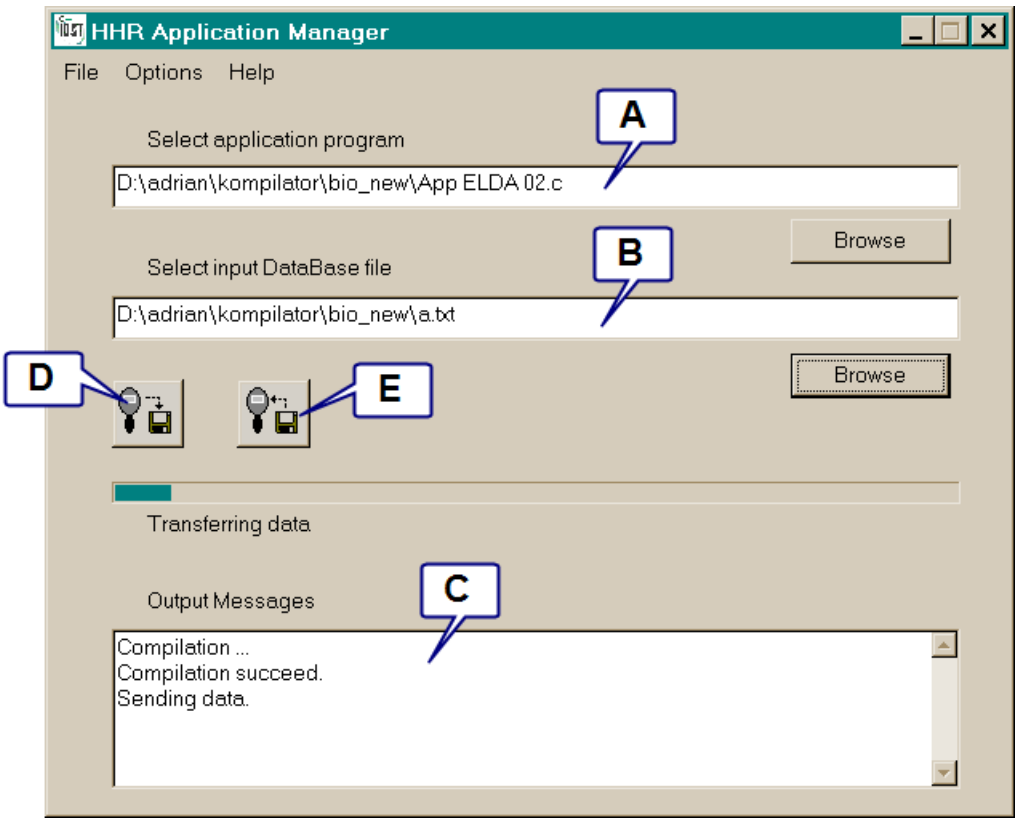
MANAGING HHR 3000 PRO

HHR 3000 PRO is equipped with newest HHR OS (operating system) which enables to run the Application Program on HHR. Very important is to use compatible version of HHR Application Manager and HHR OS. It is possible to update HHR with a newer version of HHR OS (file with extension hex).

To program Your HHR with HHR Operating System (HHR OS).

1. Run "ProgramLoader.exe", Important: HHM Application Manager and ProgramLoader can't be in the same catalog.
2. Connect HHR to USB port using included cable
3. Select appropriate hex file (HHR OS) and press Start
4. Wait until "Programming succeed" appear in Programming Status field.
5. Disconnect Your HHR from USB port.

OPERATING THE HHR APPLICATION MANAGER



Sending an Application Project

1. Select the Application Program by pressing the Browse button beneath “A” field,
2. Select User input database by pressing the Browse button beneath “B” field,
3. Press the “E” button to connect(compile) selected files to Application Project and send it to your HHR,
4. If definition of database in Application Program match User input database and Application Program hasn't got any errors the compilation will be successful and Application Project will be send to your HHR.

Selecting the User input database isn't required, if you don't select it an empty database will be sent to your HHR.

If compilation fails (point 4) errors will appear in Output Messages field (“C” field), you can try to debug the Application Program at indicated line.

Receive a User output database

1. Press the “D” button to receive the database,
2. Select the file(modify an old file, or create a new one) where User output database will be written,
3. Select the User output database file type, and press Save,
4. When the receiving will be finished, User output database will be available at location given in Output Message field (field “C”).

! Converting the MS Excel file to cvs or txt file is very simple, you have to use the operation File->Save as and there select the appropriate for you option (csv or txt) with specified separator, it is important to declare the same separator in Application Program.

Converting from other programs like OpenOffice's Calc or SQL-based programs is similar.

To import the csv type User output database file you have to double-click on file to import it to MS EXCEL, if the file type is txt you have to use operation File->Open and there select appropriate for you type (txt) and use import wizard.

Importing User output database to other programs like OpenOffice's Calc or SQL-based programs is similar.

The matching definition of database and User input database term is a little bit complicated. If you are not familiar with HDS rules, the most important is to make the same amounts of lines in TABLE section and columns in User input database.

Remaining rules will be explained in HDS description.

HDS – HHR Development Script Language

General information

HDS has been developed especially for customizing your HHR in easy way with wide range of possibilities.

Application Program written in **HDS contains 9 sections**, all of those sections are necessary, but if you don't need any you can left it empty

Example, section with values:

```
GLOBAL
0:global1
1:global2
END
```

Example, an empty section:

```
GLOBAL
END
```

HDS allows you to put comments and empty lines in Application Program. Comment are sign for HHR Application Manager that this line is a user-line and won't be taken into account during compilation (connecting Application Program and User input database).

The comment sign is double slash (//).

Example:

```
GLOBAL
1:global2
//2:global3
END
```

Empty line between script line and any white-characters (tabulator, spaces) at the beginning of line are allowed.

IMPORTANT: white-characters aren't allowed between characters in line.

For example:

```
GLOBAL
0:global1
1:global2
//2:global3
2:global4
    3:global2

  4:global2
END
```

Sections

Application Program contains 9 sections:

- **HEADER** – the section where you declare general information about Application Project, like project name, date/time format, loading date and time, etc

  This section must be placed in every Application Program

- **TABLE** – the section where you declare database column by column which will be used in Application Project. Each line contains 7 parameters describing one database column.

  Amount of lines in TABLE section must be equal to amount of columns in User input database. Type of column (date, transponder no, etc) is defined, this type must agree to data in User input database.

- **LOG** – this section describes a LOG table, syntax is analogical to **TABLE** section.

- **GLOBAL** – this section contains global constants for Application Program, you can use those “Globals” to save memory by placing a number of Global instead of long value (farm number or veterinary number etc.) in database. In this way more data can be stored in your HHR.

- **MESSAGE** – this section contains messages that you want to appear on LCD at specified time during work with HHR, user can define the time when messages will be shown on screen. Messages, can be saved in database in similar way like Globals, the number of message is saved in database instead of any long value. In this way more data can be saved in your HHR.

- **MACRO** – this section is the core of Application Project. User defines here how HHR should behave after entering to specified menu item. You can write as many macro as you want but you must remember that every macro takes space in memory and there is one memory for Application Program and User input database.

- **MENU** – the section where user declares menu, sub-menu, etc. HHR functionality for each menu(sub-menu) is related macro.

- **START SCREEN** – this section describes what will be displayed on HHR's LCD just after turning on.

- **WARNING** – this section contains standards warnings(messages) like “Turn on”, “Please wait”, or “Stored”, user can use standard warnings or define his own.

- **GLOBAL** – this parameter defines whereas globals will be loaded from file or it will be included in Application Program.

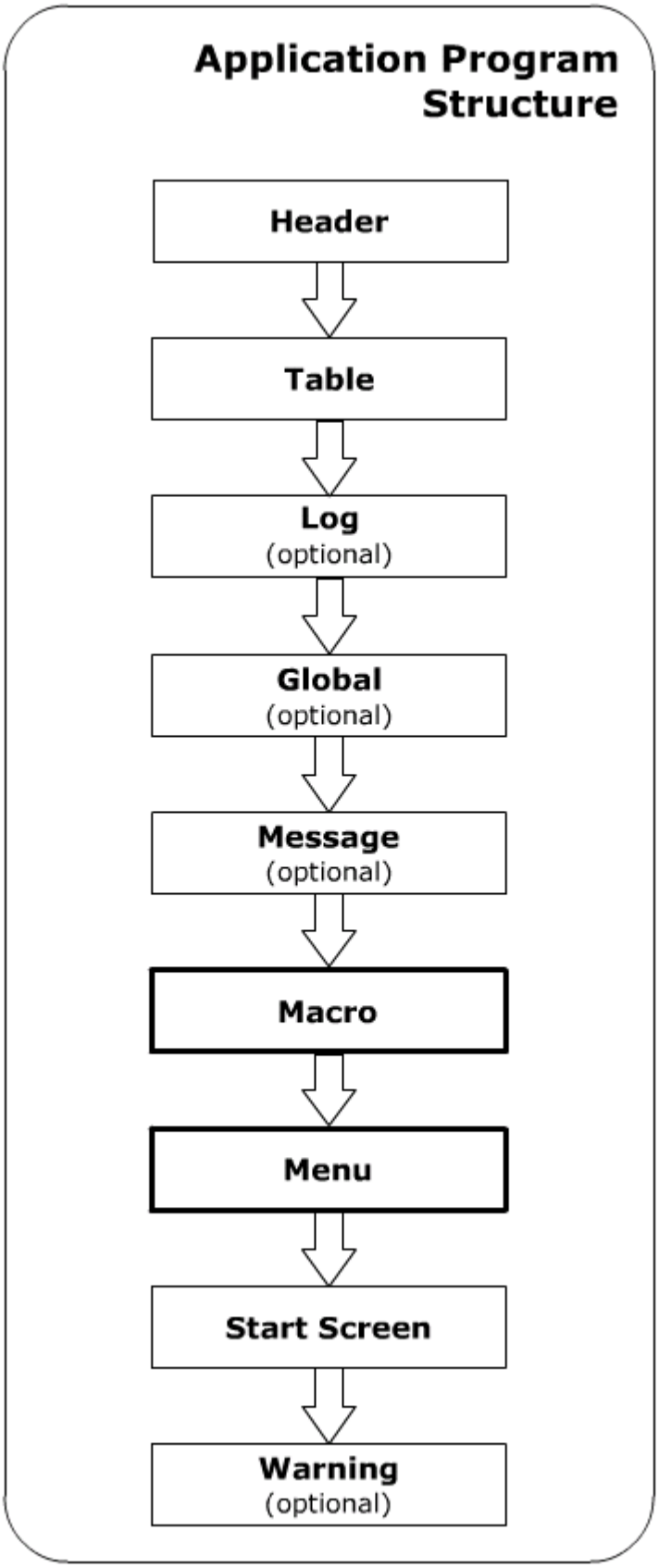
  If you choose to load Globals from file you have to place a “**global in.txt**” file in the same catalog with Application Program.

  During receiving database globals will be automatically loaded from a HHR to “global out.txt” this file will be placed in the same catalog with User Output Database.

  GL_file indicates that globals will be loaded from a text file, and **GL_section** points that globals will be included to Application Program (HHR App will not be looking for any file containing globals)

An example of Application Program with all sections:

```
// application: EXAMPLE
////////////////////////////////////

HEADER
  xxxxxx 01
  09/02/2007
  10:34:59
  dd/mm/yyyy
  hh:mm:ss
  ,
  LOG_false
  .
  BT_false
  GL_section
END

TABLE
  MOTHEREID;R;1;1;1;0000 0000 0000;;
  MOTHERVID;S;1;1;1;&&&&&;;
  DATE;D;0;1;1;0000000000;;
  GOAT1;S;0;1;1;&&&&&;;
  GOAT2;S;0;1;1;&&&&&;;
  MALE1;S;0;1;1;&&&&&;;
END

GLOBAL
  0:001234144423223
  1:001141423412331
END

MESSAGE
 0:value not valid
 1:reenter it
END

MACRO
 begin_macro:LAMBING
      begin_action_area
            ReadNew(MOTHEREID)
            FillExp(DATE,date())
      end_action_area

      begin_screen
            PrintExp(46,0,0,count(*))
            PrintText(115,0,0,"1.1")
            DrawLine(0,6,127,6)
      end_screen
      begin_control_area
      end_control_area
  end_macro
END

MENU
      M0:{Lambing,M1};{Settings,M2}
      M1:LAMBING
      M2:SETTING
END

START_SCREEN
  begin_screen
      PrintText(0,0,0,"TABLE:
EXAMPLE")
      PrintText(30,20,5,"I D & T")
      PrintVersion(20,55,0)
  end_screen
END

WARNING
  w_WAIT:Please/wait!
END
```


Application Program Structure
Header
Table
Log
(optional)
Global
(optional)
Message
(optional)
Macro
Menu
Start Screen
Warning
(optional)

Following sections must be included in application program:

- Header
- Table
- Macro
- Menu
- Start Screen

Rest of the sections might be omitted in Application Program. If you don't want to include any in above 5 sections left it empty.

Example:

```
START_SCREEN
END
```

IMPORTANT !

Sections must be placed in program in the order mentioned in the flow-chart above. Every section in Application Program starts with section name (in capital letters) and ends with END (also capital letters).

HEADER Section

Header has general information about Application Project.

There are max 11 parameters in a header:

1. **Name** - Application Project name, it's used only for a user for project identification, it takes 12 characters max.
2. **Date** – Project creation date, this parameter is also used for a user, it takes 12 characters max.
3. **Time** - Project creation time, used for a user, it takes 10 characters max.
4. **Date Format** – defines way of interpreting the Date type column and way of displaying Date on LCD. It is important to define the same type of date in User input database or User output database. Example: `dd/mm/yyyy`
5. **Time Format** – this parameter is analogical to "Date format", example : `hh:mm:ss`.
6. **Database Columns Separator** – defines character for separating columns in User input database. If you don't include User input database to Application Project "Database Separator" will be used in User output database received from HHR. It can be comma, semicolon, tabulator(\t), space(\s),...
7. **LOG** – defines whereas Log Table is used or not in the Application; LOG_true indicates that Log Table will be used in application and LOG_false that Log Table won't be used. The size of characters in this parameter has got meaning.
8. **Decimal Point** – defines the decimal point used to separate integer part from fractional part in User input database, it is used also during receiving User output database.
9. **Bluetooth** – this parameters defines if Bluetooth in application program is used or not. If enabled (BT_true) then HHR after every transponder read, sends information through Bluetooth protocol to master device with transponder number, date and time of reading.

   Before reading (and sending) the Bluetooth connection must be made. Other option of using Bluetooth is sending information from HHR 3000 PRO to master device at user's command, to use this property user have to disable(BT_false) Bluetooth in HEADER section. Bluetooth essentials are explained in Appendix D.

   BT_true – Bluetooth enabled;

   BT_false – Bluetooth disabled.

10. **Global** – this parameter defines whereas globals will be loaded from file or it will be included in Application Program.

    If you choose to load Globals from file you have to place a "**global in.txt**" file in the same catalog with Application Program.

    During receiving database globals will be automatically loaded from a HHR to "global out.txt" this file will be placed in the same catalog with User Output Database.

GL_file indicates that globals will be loaded from a text file, and **GL_section** points that globals will be included to Application Program (HHR App will not be looking for any file containing globals)

The structure of "global in.txt" file

```
0:global_1
1:vet_1
..
```

11. **Animal Transponder** – this parameter defines whereas only animal transponder will be read by HHR, in other case a warning will appear on HHR LCD. **animal_true** indicated that only animal transponders will be read, **animal_false** indicates that HHR won't be checking whenever just read transponder is an animal one.
12. **Speed Mode** – HHR is able to move to the next(previous) record when user press down(up) arrow during editing a value in field. This option can be enabled by **SM_true**, or disabled by **SM_false**.

If you are using **Speed Mode** you must be aware that none icons are available to put in Screens, the functions which aren't allowed to use are:

- IconDelete
- IconAddRec
- IconFindCol
- IconConfirm
- AddRecConf

Parameters 1-8 must be placed in HEADER section, rest of parameters are optional, if you don't define it, a default value will be used.

Default values:

```
BT_false
GL_section
animal_false
SM_false
```

HEADER example:

```
HEADER
  EXAMPLE 01
  09/02/2007
  10:34:59
  dd/mm/yyyy
  hh:mm:ss
  ,
  LOG_true
  .
  BT_false
  GL_section
END
```

As it was shown in the example the decimal point is . (point) and column separator is , (comma). It is important to declare the same decimal point and database column separator in HEADER section and use those in User input database.

According to HEADER definition above the example of User input database is:

```
1000000098200003542330 0,230.00,09/02/2007
1000000098200003542330 1,231.23,09/02/2007
```

Date format and Time format parameters can contain only special characters like **d,m,h** (for Date format) and **h,m,s** (for Time format). As a separator between days and months or hours and minutes user can use any chars from 32 to 127 number from ASCII table.

For writing first applications in HDS scripts we recommend you to set LOG_false in HEADER section. LOG_true forces more advanced function parameters which are described in appendix C.

Header definition starts with section name **HEADER** in capital letters and ends with **END** word in capital letters.

TABLE Section

In this section user defines Table used in ADS, User defines column by column in each line, each definition takes 7 parameters.

1. **Column name** – Name of column in table, this name will be used MACRO section for table operation. Column name takes 12 characters max. The name can't consist characters , (comma) ; (semicolon) .(point).
2. **Column type** – Type of data in defining column, ADS allows to use 8 types:

Type	*Description*
R	Transponder read
O	Option
M	Message
G	Global
S	String
D	Date
T	Time
A	Any Type

   The meaning of each type will be explained below.

3. **Unique value** – this parameter defines if the value can be repeated in column in other record or it can't; “non unique” indicates that it can, “unique” indicates that value can't be repeated in whole column. This parameter will be useful for Log users because the Unique value defines the key(index) column in User input/output database.

   Unique value takes only one char - “0” if column isn't unique or “1” if data in column is unique.

4. **Editable value** – this parameter defines whereas the column is constant or in can be modified after loading or entering from HHR's keyboard.

   Editable value takes only one char - “0” if column isn't editable or - “1” if data in column has to be editable.

5. **Null value** – this parameter defines if values in columns might take null value, it all comes to left a field empty during adding a new record from HHR's keyboard, or sending User input database to HHR 3000 PRO.

   Null value takes only one char - “0” if column can't be null or “1” if column can be null.

6. **Mask value** – The mask defines the way of saving data in memory and displaying it in HHR's LCD. Mask consist of all alpha-numerical characters(0-127 ASCII range), we can distinguish special characters and standard characters; Special characters are &,0,#,_,/; characters &,0,# work as a “boxes” for data from field. That “box” is a place in LCD where data from memory(database) is placed.

   Maximum mask length takes 25 characters.

- & - displays any character, but if there is nothing to display(field is empty or, all data from field was displayed) nothing won't be displayed,
- 0 - displays numerical characters, but if there is nothing to display(field is empty, all data from field was displayed, data isn't an numerical character) 0 will be displayed,
- # - displays numerical characters(and comma), but if there is nothing to display(field is empty, all data from field was displayed, data isn't an numerical character) nothing won't be displayed,
- $ - displays numerical characters(without comma), but if there is nothing to display(field is empty, all data from field was displayed, data isn't an numerical character) nothing won't be displayed,
- _ - this character is used to display special characters(&,0,$,#,/,_)
- / - this character is used to make a break-line on LCD (enter-key function).

  Standard characters are all characters from ASCII table (range:32-127), standard characters aren't saved in but it is displayed according to mask definition, so that we save memory.

Example:

Mask 000aa##_0	**Display** 102aaa10
Memory 102a131	

Mask example 1.

In example 1 according to mask definition first 3 characters are digits, next it will be 'a' char twice, those 'aa' doesn't have any connection to database it is simply mask definition. After that is a any-character twice, in display 'a1' appears according to database 4th and 5th character. The last character at display comes from "_0" mask definition which means – print special char 0.

Mask 0.00kg _##	**Display** 1.02kg #F
Memory 102F	

Mask example 2.

In example 2 according to mask definition first char on display will be a digit from memory, next will be decimal point. Third and fourth character on display will be two digits from memory "02", after that according to mask "kg #" will appear on display, # comes from "_#" mask definition which means – print special char #, and the last character is any character from memory, but if there isn't any don't display anything, in our case "F" will be displayed.

7. **Default value** – this parameter defines default value after creating a new record.

Example of Table section:

```
TABLE
  MOTHEREID;R;1;1;1;0000 0000 0000;;
  MOTHERVID;S;1;1;1;&&&&&;;
  NRFEMAL;S;0;1;1;#;0;
  NRMALE;S;0;1;1;#;0;
  NRDEAD;S;0;1;1;#;0;
  GOAT1;S;0;1;1;&&&&&;;
END
```

Example of appropriate User Input Database

```
10000000982000035423300,W6W6W,2,3,0,9WWWW
10000000982000022145678,ABC6W,2,0,1,G6J6T
```

The separator between parameters defining column is ";" (semi-colon).

> ***!*** ***The maximums***
>
> 45 columns in database declaration.
>
> 25 characters mask length.
>
> 20 special characters in mask declaration.
>
> 12 character column name.

Data type

As it was already mentioned we can distinguish 7 types of data.

Type	*Description*
R	Transponder read
O	Option
M	Message
G	Global
S	String
D	Date
T	Time
A	Any Type

- **Transponder read – R** – this type indicates that in column will be a transponder number, such field take 23 characters in memory,
- **Message – M** – this type indicates that column will be fulfilled with numbers of messages declared in MESSAGE section, such field take 2 characters in memory and mask will take 2 “0”, user don't have to declare mask it will be declared automatically,
- **Global – G** – this type indicates that column will be fulfilled with numbers of constants(globals) declared in GLOBAL section, such field take 2 characters in memory and mask will take 2 “0”, user don't have to declare mask it will be declared automatically,
- **String – S** – this type allows user to use his own alpha-numerical string, user declares length of field by defining mask,
- **Date – D** – this type indicates that column will contain date, user don't have to declare the mask for this field, ADS will use date format declaration from HEADER section, the length for this type of field is constant and take 10 characters.
- **Time – T** – this type indicates that column will contain time, user don't have to declare the mask for this field, ADS will use time format declaration from HEADER section, the length for this type of field is constant and take 8 characters.
- **Any Type – A** – this type indicates that column will contain transponder numbers, the difference between “A” and “R” type is that in “A” type user define the amount of saved digits by mask, for example:

  EID;A;1;1;1;#### ####;;

  Above, “A” column has been defined, only 8 last digits from transponder number will be saved i memory.

1	0	0	0	0	0	0	0	9	8	2	0	0	0	0	3	5	4	2	3	3	0	0

Only digits with gray background will be saved in memory.

- **Option – O** – this type indicates that column has predefined list of values and user of HHR can't write his own, he can only select one of defined values. As option has limited way of values displayed in LCD, the mask is pointless, so that user declares values of the list separating it by comma (“,”), each option takes 10 characters max, and 20 values is max for one option field;

  Example:

  ```
  OPTION_3;O;0;1;1;WWWWWWWWWW,1WWWWWWWWW,2WWWWWWWW
  WW,3WWWWWWWWW,4WWWWWWWWW,5WWWWWWWWW,6WWWWWWWWW,
  7WWWWWWWWW,8WWWWWWWWW,9WWWWWWWWW;WWWWWWWWWW;
  ```

  In this example column TRANSP_3 is Option type, instead of mask list of values is placed(list of values is underlined in example) user will be able to select one of them during work with HHR, move over you can put default value of field.

  User can define 20 options max in one field, the maximum length of each option is 10 characters.

> ***! The syntax of any line in TABLE section.***
>
> ```
> <col_name max 12ch>;<col_type 1ch>;<unique 1ch>;<edit 1ch>;
> <null 1ch>;<col_mask max 25ch>;<default_val max 25ch>;
> ```
>
> It is important to make **default value match declared mask**.

Example of column definition:

```
MOTHEREID;R;1;1;1;0000 0000 0000;;
```

Above, column MOTHERID has been declared, the type of this column is transponder read, this column is unique (values in column can't be repeated – you can't read the same transponder twice), this column is editable and it can be null. After that mask is declared, mask consist of 3 groups of digits separated with space, there is no default value declared.

Example of TABLE section:

```
TABLE
  MOTHEREID;R;1;1;1;0000 0000 0000;;
  MOTHERVID;S;1;1;1;&&&&&;;
  NRFEMAL;S;0;1;1;#;0;
  NRMALE;S;0;1;1;#;0;
  NRDEAD;S;0;1;1;#;0;
  GOAT1;S;0;1;1;&&&&&;;
END
```

Example of appropriate User input database

```
1000000098200003542330O,W6W6W,2,3,0,9WWWW
1000000098200002214567 8,ABC6W,2,0,1,G6J6T
```

GLOBAL Section

Globals are constants, to which you have access during work with HHR 3000 PRO.

GLOBALS were developed to save space in HHR memory. If you want to place for instance word `Veterinary12345` in database field you can place only number 1 instead of long character string. Numbers and corresponding character string are defined in section Global. Short numbers are stored in memory and corresponding value (character string) is displayed on LCD screen.

> ***! The syntax of GLOBAL section.***
>
> ```
> 0:FarmDolySheep
> 1:Veterinary12345
> 2:AnimalGroup10
> ```

User declares Globals by placing number of global, a colon and value of Global; maximum length of each global is 23 characters. The maximum amount of globals in Application Program is 99.

Global definition starts with section name in capital letters and ends with END word in capital letters.

Example of Global section:

```
GLOBAL
  0:FarmDolySheep
  1:Veterinary12345
END
```

Database

...	AnimGroup	...
	1	
	2	
	3	
	2	

Application Program

```
GLOBAL
   0:AnimalGroup7
   1:AnimalGroup8
   2:AnimalGroup9
   3:AnimalGroup10
END
```

Display

```
AnimalGroup8
AnimalGroup9
AnimalGroup10
AnimalGroup9
```

AnimGroup column has Global type. User can display Global field from column using macros. Using **PrintGlobal function** with appropriate number you will see results like in Display window above.

PutGlobal function, enables to write a number of global.

After receiving database from HHR numbers of globals can be easily converted with global values.

HDS allows you to load Globals from text file during compilation and save Globals to text file during receiving User Output Database. Those text files are “global in.txt” and “global out.txt”. The file “global in.txt” has to be placed in the same directory where Application Program is, “global out.txt” will be placed in the directory where Output Database will be.

The structure of files “global in.txt” and “global out.txt”.

```
global_1
vet_1
..
```

MESSAGE Section

Messages, are small information which informs user about some event. User can define this event, or time when such message appears. Messages, similar to Globals are saved in memory as a number. Messages can have maximum 30 characters and maximal number of messages is 99.

Message definition starts with section name in capital letters and ends with END word in capital letters.

! The syntax of MESSAGE section.

```
0:For sale
1:Treat
2:Separate
```

Message section syntax is analogical to Global section, the only difference is the length of Message, it can take 30 character.

Example of Message section:

```
MESSAGE
 0:For sale
 1:Treat
 2:Separate
END
```

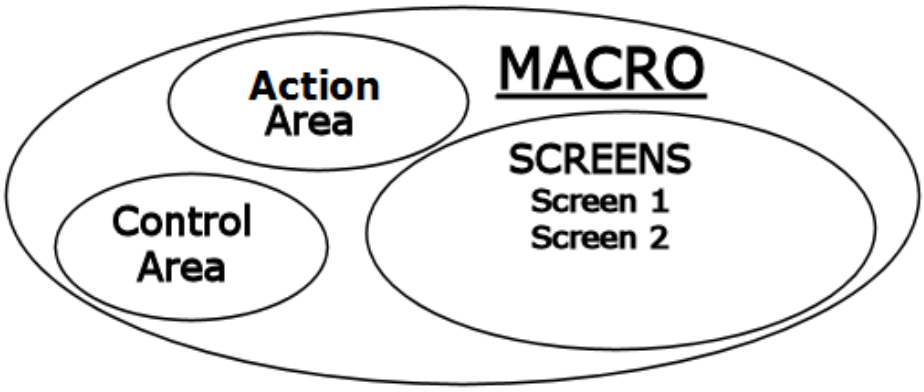
MACRO Section

From the user's point of view macro is a set of screens, with database fields values, which require to be individually completed in order to manage the database.

Macros are the core of Application Program. Organize working of HHR after entering menu. User can create as many macros as he want, but very important is that HHR 3000 PRO has only 256kB EEPROM memory, where application program and user database are placed. **Application Project can't oversize 256kB**.

Macro definition starts with section name in capital letters and ends with END word in capital letters.

Each macro starts with `begin_macro`:macro_name and ends with `end_macro`, macro name is used to identify macros at MENU section. Each macro consist of 3 sections, Action area, Control area and Screens.



Action area consists of function carried out when Special Screen functions are performed or Transponder Read button is pressed(other options available), it depends on definitions included in Action Area.

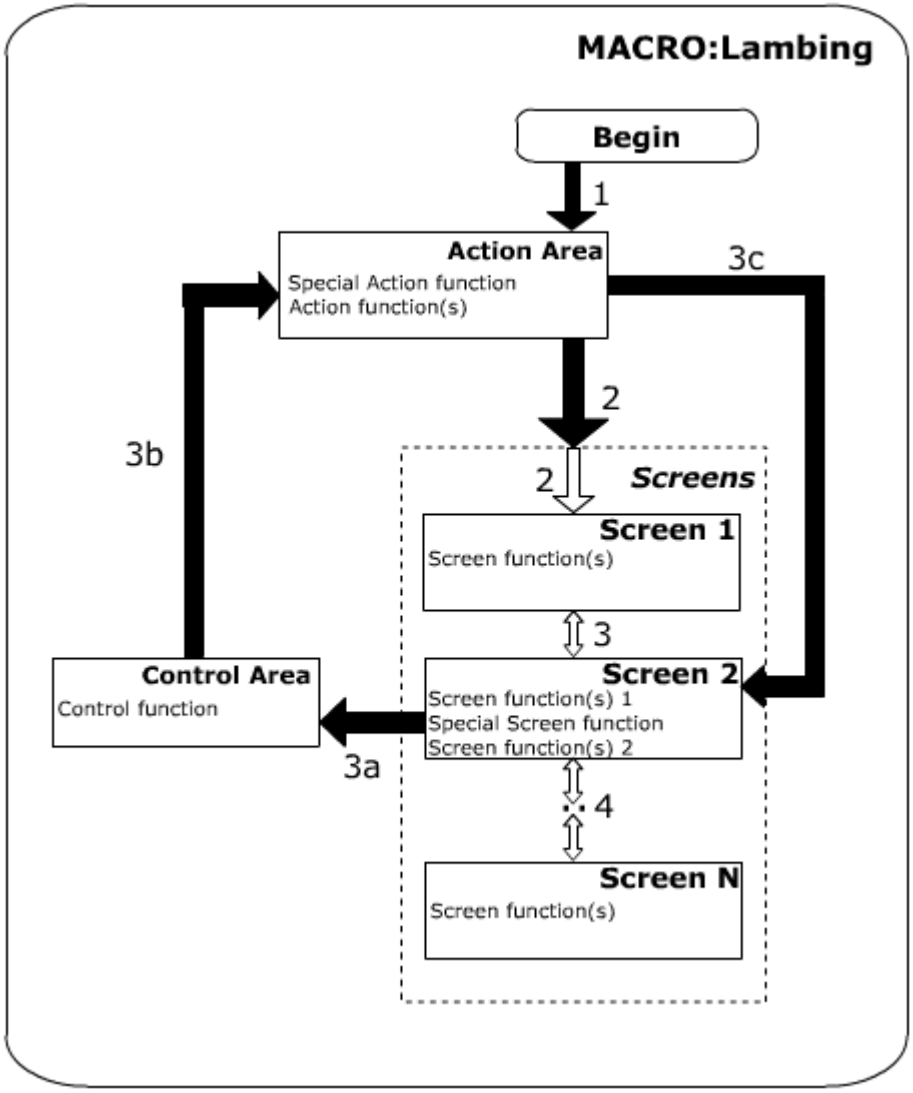
Action area starts with `begin_action_area` and ends with `end_action_area`, Control area starts with `begin_control_area` and ends with `end_control_area`.

Action area and Control area(each of them) can maximally consist of 25 functions. Only 25 database fields can be modified(edited) in each macro.

Action area and Control area are concerned when current record number changes, current record number is simply a number of record on which a user is working at specified time, a number of record can be changed by functions at screens. When Special Screen Function changes a record number Control area runs, and after that Action area. It is very important you to understand that Action Area and Control Area runs only if record is changed by Special Screen Function, it is possible to show it to you as if-clause (decision block).

There are also Special Action Area functions which define HHR to move to Control area and after that to Active area; for example when user press the Transponder Read button.

The Special Action Functions are **ReadNew, ReadSeek, AddRec, Read**. The Special Screen Functions are **EditReadNew, EditReadSeek, IconAddRec, EditNewField, IconConfirm, EditSeekField**. Those Special functions will be described after general macro description.



Flow chart 1

Description of each point on flow chart 1.

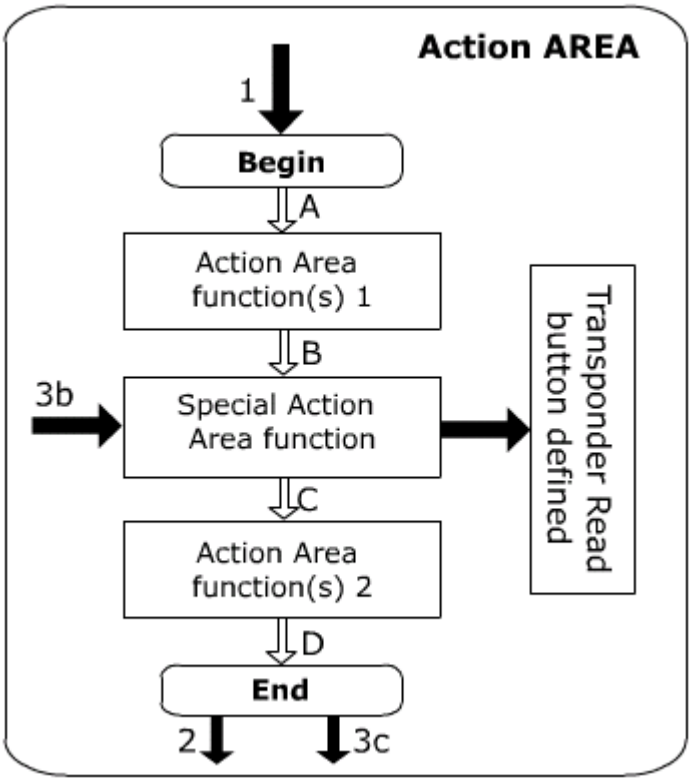
1. User enters macro, HHR is checking whenever a Special Action Function is in Action Area, if it is, HHR sets its registers to know where call-back if:
   - Special Screen Functions would be performed,
   - Transponder Read button would be pressed(TR button must be predefined to work in Action Area with one of Special Action Functions).

Only one Special Action Function is allowed to be in the Action Area, if you put more then one last of listed will be regarded.

At point 1 when user enters macro no functions are performed, HHR is only checking if a Special Action function is in Action Area and after thats goes to first screen.

2. HHR perform all Screen functions included in Screen 1
3. HHR perform all Screen functions 1 from Screen 2, after that HHR run into Special Screen functions then HHR move to Control Area and perform its functions, after that HHR move to Action area and perform its functions. Next HHR returns to Screen 2 and perform Screen functions 2
4. HHR perform all Screen functions in any screen only if there is less then 10 screens in each macro
5. HHR moves to next screen and perform its functions.

As it was already mentioned after entering the macro(and action area) HHR only check if Special Action Area function is in code, if it is HHR sets its registers, if there isn't any nothings happens, next HHR move to screens and run first screen.



Flow chart 2

The Special Screen functions or using defined Transponder Read button make HHR behave the same way. HHR moves from current screen to Control Area and performs functions assigned there, after that HHR moves to Action Area and perform functions and at the end HHR comes back to screen that user was before.

To give you more detailed look let's assume that Transponder Read button is defined by Special Action Area function, then HHR will be working according to

points below(based on flow charts)

- move from current screen(screen 2) to Control Area
- perform all Control Area functions and move to Action Area (3b point at flow chart 2)
- Perform Special Action Area function and Action Area function 2, you may think that Special Action Area function has only one purpose(ex: defining TR button) but those function take many action at one time for example: reading transponder; looking for it in Table, creating a new record and more. For more detailed information refer to Special Action Area functions chapter.
- move from Action Area to screen where HHR was before pressing the Transponder Read button (3c on flow charts).

Example of macro section:

```
MACRO
begin_macro:SETTING
      begin_action_area
            ReadNew(MOTHERID)
            PASTE(DATA)
      end_action_area

      begin_screen
            PrintText(0,0,0,"RECORDS:")
            PrintExp(46,0,0,count(*))
            PrintText(115,0,0,"2.1")
            DrawLine(0,6,127,6)
            PrintText(0,10,1,"Time:")
            SetTime(40,9,1)

            PrintText(0,25,1,"Date:")
            SetDate(40,24,1)

            DrawLine(0,63,127,63)
      end_screen

      begin_control_area
            Copy(table)
      end_control_area

end_macro
END
```

The Special Action function and Special Screen function were mentioned in this chapter many times, those function have the key-role in creating macro.

IMPORTANT:

The Special Action Functions are ReadNew, ReadSeek, AddRec, Read.

The Special Screen Functions are EditReadNew, EditReadSeek, IconAddRec, EditNewField, IconConfirm, EditSeekField.

Keyboard

Keyboard buttons have predefined functionality and there is no possibility to change this functions by the user. These functions are defined in HHR Operating System.

The keyboard rules:

To move from one data field to the next on HHR's LCD use the **Enter key or Down/Up Arrows**. Rectangle around data field is a cursor. When the Enter key is pressed on the last field of a screen, the cursor returns to the first field of that screen.

To enter edit-mode at current cursor position user have to press a **button from numerical keyboard** (from 0 to 9), after it HHR will clear the consistence of frame and allow user to set a new value with numerical keyboard. User can use the **backspace button**(back arrow) to fix the value or finish editing with **Enter key**.

HHR enables to use Speed Mode, which allows user to finish editing with a **Up** or **Down Arrow** and move to next record. The mode can be turned off/on for whole App program in HEADER section.

Functions which manage database (delete record, add record, seek for record) are started by pressing any **button from numerical keyboard** (from 0 to 9) when cursor is on the icon.

If user wants to cancel editing and return to the last value he will have to make a **long press of backspace button**(back arrow).

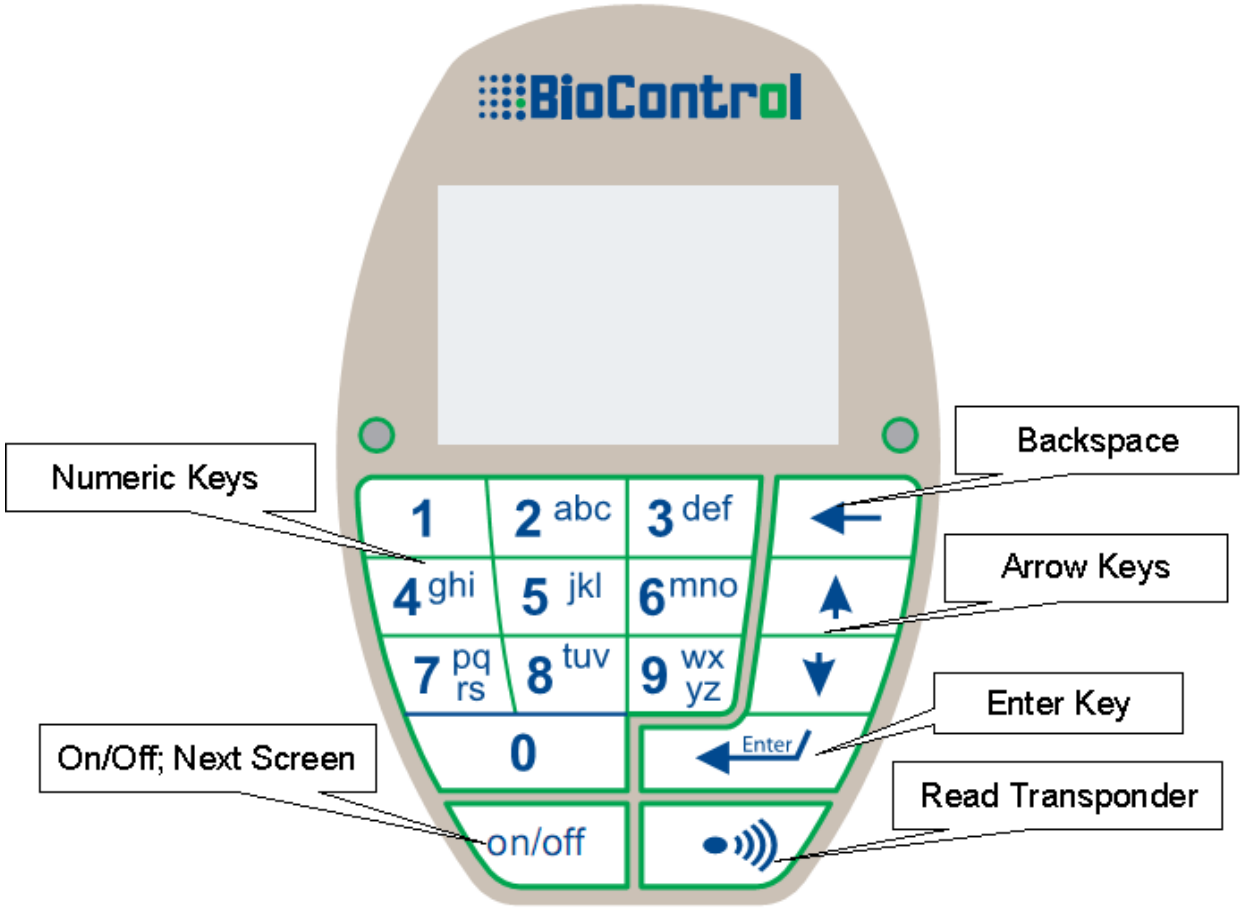
If the cursor is placed on transponder number database field, user can put this value by **Read Transponder button**.

To move to the next screen use **Power On/Off**, screens are changing in a loop if HHR reaches the last screen HHR moves to the first screen.

User can move to next or previous record with **Up and Down Arrow buttons Long Press.**

If you are using Speed Mode you must be aware that none icons are available to put in Screens, the functions which aren't allowed to use are:

- IconDelete
- IconAddRec
- IconFindCol
- IconConfirm
- AddRecConf



Keyboard Rules		
The Key	Speed Mode	Normal Mode
Enter Key	Moves cursor to the next operation input box; finish editing a value in input box; enter to icon function.	
Back Space	Short press: out of macro, menu, delete last character when editing a value in input box. Long press: out of editing a value in input box without changing it.	
0..9	If the cursor is in field(input box) starts editing a value in field(input box)	
Transponder Read	Depending on App Program reads transponder number.	
Power On/Off	Moves to the next screen, if HHR reaches last screen move to the 1st one.	
Arrow keys	Save current record in memory and moves to the previous record (down arrow), or next record (up arrow).	Short press: move to the next field (input box) and icon (+ ; - ; search ; confirm) Long press: Move to the previous (Down arrow) or next (Up arrow) record. When user is editing a value in field(input box) moves between characters.

Creating a Macro and user functionality

Summary of available functions in HHR OS are in Appendix B

Second part of Programmers Manual **“HHR 3000 Pro Functions”**
describes in detail all functions which could be used in macro.

We can make automatic fulfilling a current or new record in database. We can define when such action will be taken - when user press transponder read button, or enter manually a transponder number to selected transponder field, or click on “+” icon to add new record.

The key element in each macro is special screen functions and special action functions.

Special action functions define a position below each action function will be performed. Look at macro NEW_REC placed in example at the end of “HHR 3000 Pro Functions”.

Only functions which are below ReadNew function will be performed if user press Read button (ReadNew function call and go to action area). If you would put anything before ReadNew function in action area it wouldn't be performed.

According to example, ReadNew function defines a transponder read button to work in whole macro, so after pressing it a functions from action area placed below ReadNew function will be performed. In that case HHR's user after pressing the button will add a new record fulfill it automatically with defined in App Program data, and return to screen where user was at the beginning. User won't even see any changes on the screen everything will be done in “background”, the only change that will be performed is that current record might be changed and data in the fields will change.

There is also a **control area** where you can put a Copy function which will copy current record to a buffer, to paste the data to new record. You can use a Paste function in Action area. Control area will be performed by Special Screen Function, but you should do it wisely because HHR will copy and paste any value without controlling consistence of it.

There is only a slice difference between **ReadSeek and ReadNew** (**EditReadSeek and EditReadNew**) but you can use it to make a read-only mode or write and update mode. It is possible because ReadSeek (EditReadSeek) seek for transponder number in database. If HHR doesn't find one, nothing happens, HHR comes back to the screen.

ReadNew (EditReadNew) seek for transponder number in database and if HHR doesn't find it HHR will create a new record, perform all Action functions and put a transponder number to indicated column.

Another very important issue in each macro is understanding the sense of action performed by function for instance:

IconDelete deletes current record and set a first record in database as current record.

IconFindCol displays a new window on HHR's LCD where user can select a

value(and automatically a record), after selecting a value by user, current record number will change to the selected one.

There is also Sorting function in Action area which define after which column HHR will sort data, this function also changes current record number according parameters.

All those functions are sort of database manage functions, thanks to them user can handle database by erasing, adding or modifying records.

The main advantages of HHR script language are:

- Automation – the structure of macros allow to perform set of functions on an event, which may be caused by pressing transponder read button or performing special screen function. HHR allows for automatic fulfilling (part or whole) new record. User can automatically add a information about new record to Log and register there newly read transponder, weight or health status.
- User interface – HHR by its functions like DrawLine, DrawRec and PrintText makes the opportunity to create friendly user interface. The same functions might be used to design a start screen for HHR, such start screen helps to identify the application program that HHR is programmed with.
- Data validation - masks give an opportunity to control entering data. There is no possibility that HHR will allow to put a values incompatible with mask. If you want to close the list of values which will be placed in a column you can use a global type or option type where is no possibility to put inadequate value.
- Log – Log enables user to save information in Log table freely definable, this is useful in registering any animal events, Log is received from HHR to PC the same like Table but it allows to analyze data in shorter time.

Those are the main advantages of HDS, if you want to use maximum abilities of HHR think of using those points in yours programs.

> ***! Read function rule.***
>
> If any of Read function (ReadNew, ReadSeek) has been used in Action Area, any of Read function(EditReadNew, EditReadSeek) can't be used in Screens in the same Macro.
>
> **It is extremely important you to follow this rule, otherwise Application Program won't be compiled by HHR Application Manager.**

MENU Section

Designing menu is very simple, a menu schema is similar to the tree in some way.

A menu definition example

```
MENU
      M0:{New Animal,M2};{View/Edit DB,M3};{Farm & Vet,M4};{Settings,M1}
      M1:{BlueTooth,M6};{Date/Time,M5}
      M2:NEW_REC
      M3:VIEW
      M4:GLOBAL
      M5:DATE
      M6:BLUETOOTH
END
```

The lines are enumerated with M<menu_element_number>: to help you better organize your menu, with enumeration you don't have to define the same menu many times if you want to refer to it more than one time.

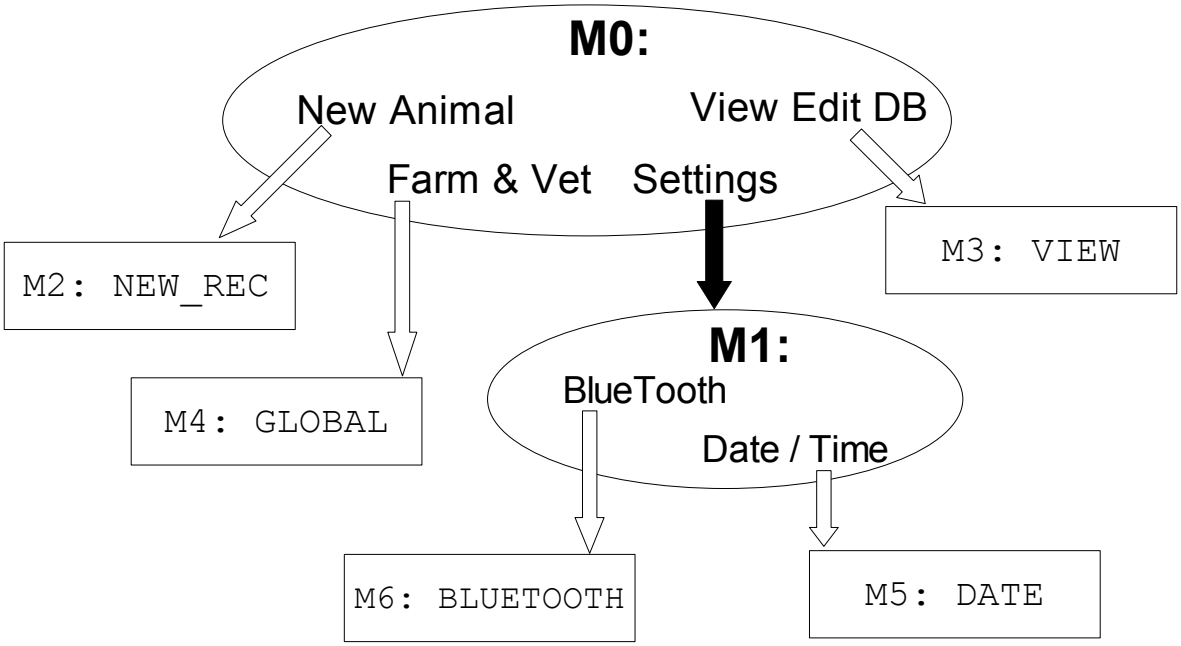
The M0 level is a trunk of menu design, its main menu, the first menu that is shown on the screen, just after a start screen. The menu design syntax is following: just after enumeration and colon you define list of menu elements in curly brackets, each menu element is in bracket. Brackets are separated each other with semicolon, last menu element isn't followed by any char.

Every curly bracket contain two parameters:

1. Menu value which will be displayed on HHR screen,
2. Menu identifier that current menu element refer to, after entering a menu item a sub-menu identified by menu identifier will run.

Each menu or sub-menu item has to be finished with macro, you can connect macro with menu identifier in following way: after menu enumeration and colon you have to enter a macro name defined in macro section, the size of letters is important.

According to the example above we can present this menu like a flow-chart.



The flow-chart presents the menu structure, we have a main menu(M0 level) and one sub-menu(M1 level), the elements in rectangles are macros – the final step for each element in menu or sub-menu.

You can put maximum 10 elements in each menu or sub-menu. The maximum length of menu value which displays on the screen is 12 characters. The maximum level of menu is 4, so the longest way to get to the macro is 4 steps.

START SCREEN Section

This section allows you to define a screen which will appear on HHR's LCD just after turning on a HHR. User can close this screen (move to menu) by pressing any button or screen close itself after a few (5) seconds.

The functions which might help you defining a Start Screen are:

PrintText(x,y,font,text)

PrintText functions simply prints text on HHR's LCD, the parameters:

- x – describe horizontal position of text. X can't take more then 127 (LCD dimensions).
- y – describe vertical position of text. Y can't take more then 63 (LCD dimensions).
- font – describe font size, there are 6 font sizes, you can select a one by choosing a digit from 0 to 5 where 0 is the smallest size and 5 the biggest.
- text – the text which you want to appear on HHR's LCD, according to function syntax you have to take the text in quotation marks.

Example:

```
PrintText(0,0,0,"TABLE: ESPANA 01BT")
```

PrintVersion(x,y,font)

PrintVersion functions prints a number of HHR OS, the parameters:

- x – describe horizontal position of text. X can't take more then 127 (LCD dimensions).
- y – describe vertical position of text. Y can't take more then 63 (LCD dimensions).
- font – describe font size, there are 6 font sizes, you can select a one by choosing a digit from 0 to 5 where 0 is the smallest size and 5 the biggest.
- text – the text which you want to appear on HHR's LCD, according to function syntax you have to take the text in quotation marks.

Example:

```
PrintVersion(25,25,2)
```

DrawLine(x_1,y_1,x_2,y_2)

DrawLine function draws a line on HHR's LCD according to coordinates given in parameters where x_1 and y_1 is first point and x_2, y_2 is the second point.

Example:

```
DrawLine(12,32,12,45)
```

DrawRec(x_1,y_1,x_2,y_2)

DrawRec function draws a rectangle on HHR's LCD according to coordinates given in parameters where x_1 and y_1 is first corner and x_2, y_2 is the opposite corner.

Example:

```
DrawRec(12,32,12,45)
```

Start Screen Example:

```
START_SCREEN
  begin_screen
      PrintText(0,0,0,"TABLE: PRACTISE 02")
      DrawLine(0,6,127,6)

      PrintText(33,20,5,"MY_COMPANY")
      DrawLine(0,56,127,56)
      PrintText(0,58,0,"HHR:")
      PrintVersion(21,58,0)
  end_screen
END
```

WARNING Section

Warning section allows you to customize standard messages (warnings) which appear on HHR's LCD. We can share all the messages to:

- HHR OS control messages.
- User Messages.
- Data entering messages.

> ***! Warning Rules***
>
> Maximum length of warning(value) is 30 characters.
>
> Character "/" are signal for break-line HHR. Maximum amount of break-line in each warning is 3.
>
> The warning syntax is following:
>
> <warning_name>:<warning_value>

Below a list of standard warning is presented, during creating Application Program you don't have to declare all the warning, those which you left undeclared will take a standard value, moreover during creation you don't have to put warnings in sequence(as it is in table), you can put it whatever you like.

List of standard warnings

Name	*Standard value*
w_WAIT	Please/wait!
w_UNIQUE	This value/must be unique!
w_NOEDIT	This value/is NOT editable!
w_NOTNULL	This value/must be not null!
w_LOWBATTERY	Low battery!
w_CHARGEBATTERY	Battery/charging!
w_NOT_FREE_MEMORY	No free memory!
w_NOT_STORED	NOT STORED!
w_STORED	STORED!
w_DATE_GOOD	Date changed.
w_DATE_NO_CHANGE	Date not/changed.
w_DATE_NOT_GOOD	Wrong date!

Name	*Standard value*
w_TIME_GOOD	Time changed.
w_TIME_NO_CHANGE	Time not/changed.
w_TIME_NOT_GOOD	Wrong time!
w_NOTAG	No Tag.
w_READING	Reading transp.
w_SHUTOFF	See you!
w_USBCONNECTED	USB/connected!
w_NEW_REC	Creating/new record!
w_CHOICE	Choice:
w_FIND	Find:
w_TAGNOTFIND	Transponder/number not find!
w_TAGEXIST	Transponder/already exist!
w_YES	YES
w_NOT	NOT
w_ADD_REC	Do you want add the new rec.?
w_DEL_REC	Do you want del current rec.?
w_UNKNOWN_MES	Unknown Message
w_VALEXIST	This value/already exist!
w_BTACTIVE	BlueTooth/is activated
w_BTNOACTIVE	BlueTooth/is closed
w_ON	Turn on.
w_OFF	Turn off
w_MEMORY	FREE MEMORY
w_RECSTORE	New record/stored!
w_RECNOSTORE	New record/not stored!
w_NOFIND	Value/not find!
w_NOANIMAL	No animal code!
w_COUNTRY	Wrong country code!

Example:

```
WARNING
  w_NOTAG:Tag missing.
  w_WAIT:Wait!
  w_UNIQUE:This value/isn't unique!
  w_NOEDIT:You can't edit this value
  w_TIME_NO_CHANGE:Accepted
  w_TIME_NOT_GOOD:Time Error
END
```

Appendix A : Quick Reference

To help you get into the subject or quickly modify already existing Application Program use this quick reference. If anything would be unclear or not precise enough refer to specific chapter in a manual.

The quick reference is going to be build on a program example, explaining all sections according to required order.

- HEADER Section

```
HEADER
  Quick_Ref
  2007/04/15
  18:34:59
  dd/mm/yyyy
  hh:mm:ss
  ;
  LOG_false
  .
  BT_false
  GL_section
END
```

Header section defines standard and constant settings for HDS, like date format(line no. 5), time format(line no. 6), current date and time(lines no. 2 and 3) if you need more information concerning HEADER Section refer to its chapter.

- TABLE Section

```
TABLE
  C_ID_ELEC;R;1;1;1;0000 0000 0000;;
  BREED;S;0;1;1;00;00;
  SEX;O;0;1;1;H,M,;M;
END
```

Table section is used to declare main table in HHR. It's simply a description of all the columns in Table. First parameter is a column name, by this name you refer to column in macros, second one is type, HDS allows you to use several types of column (types of data stored in HHR's memory), the sixth parameter in column definition is a mask definition. It is very important because this value decides how the data will be displayed on HHR's LCD. If you need more information concerning HEADER Section refer to its chapter.

- GLOBAL Section

```
GLOBAL
  0:FarmDolySheep
  1:Veterinary0123
END
```

Globals are sort of user standard values, equivalent of its in programming languages are variables, only a number of a global is saved in HHR data table (not the value), so that we are saving free space in HHR's memory. Maximum length of global is 23, you can declare maximum 30 globals.

- MESSAGE Section

```
MESSAGE
 0:Treat
 1:ForSale
END
```

Messages are small notes displayed on HHR screen that notify user about something. You can add the messages to database and include it in records(ex: “Treat”, “Lambing”), messages like globals aren't saved in database it is defined in Application Program and only a number of message is stored in memory. Whenever you would like to change a message value in App program don't hesitate, the only restriction is that message has to be shorter then 30 characters.

- MACRO Section

```
MACRO
 begin_macro:LAMBING
      begin_entry_area
            ReadNew(MOTHEREID)
            FillExp(DATE,date())
      end_entry_area

      begin_screen
            PrintExp(46,0,0,count(*))
            PrintText(115,0,0,"1.1")
            DrawLine(0,6,127,6)
            PrintField(0,25,1,MOTHEREID)
      end_screen
      begin_exit_area
      end_exit_area
  end_macro
END
```

Macros are the core of Application Program and Project, if you aren't common with macros, we suggest you not to make any changes, because even if program will compile there might be a logic errors like pasting data without copying it and many others.

- MENU Section

```
MENU
      M0:{New Animal,M2};{Settings,M1}
      M1:{BlueTooth,M3}
      M2:NEW_REC
```

```
        M3:BLUETOOTH
END
```

The designing menu is very simple with comparison to creating/modifying a macro, you can freely change the menu values(labels) which will appear on the screen, the only restriction is that in can not be longer then 12 characters. The menu values(labels) you can find in each curly bracket as first parameters, it is separated of menu identifier, which is very important and you shouldn't change it without referring to Menu reference.

- START SCREEN Section

```
START_SCREEN
  begin_screen
      PrintText(0,0,0,"TABLE: PRACTISE 02")
      PrintVersion(21,58,0)
  end_screen
END
```

Start Screen according to the name is a screen which appear just after HHR is turn on, Start Screen section has a few function, mainly a text function so you can change the text displayed on the screen by changing characters strings in quotation marks in each function. Although to make it easier for you we suggest you to refer to Start Screen chapter in manual.

- WARNIG Section

```
WARNING
  w_NOTAG:Tag missing.
  w_WAIT:Wait!
END
```

Warnings are standard messages which appear on HHR screen the list of events and standard values is placed in Warning chapter of this manual, you can change a standard warning value by modifying a string(not more than 30 characters) after colon in each line in warning section.

Appendix B : Function List

Function syntax	*Description*
Screen Function	
PrintText(x,y,font,text)	Prints text on HHR's LCD
PrintField(x,y,font,col_name)	Prints consistence of field on HHR's LCD
PrintExp(x,y,font,expression)	Prints an expression on HHR's LCD
PrintAnimalMark(x,y,font,col_name)	Prints 1st digit from Transponder number on LCD
PrintRetagging(x,y,font,col_name)	Prints 2nd digit from Transponder number on LCD
PrintUserInfo(x,y,font,col_name)	Prints 3-4 digits from Transponder number on LCD
PrintReserved(x,y,font,col_name)	Prints 5-6 digits from Transponder number on LCD
PrintAddInfo(x,y,font,col_name)	Prints 7th digit from Transponder number on LCD
PrintCountryNumber(x,y,font,col_name)	Prints 8-11 digits from Transponder number on LCD
PrintAnimalNo(x,y,font,col_name)	Prints 12-23 digits from Transponder number on LCD
PrintCountryCode(x,y,font,col_name)	Prints country code on HHR's LCD
PrintCountryName(x,y,font,col_name)	Prints country name on HHR's LCD
PrintGlobal(x,y,font,global_no)	Prints global on HHR's LCD
DrawLine(x1,y1,w,h)	Draws a line
DrawRec(x1,y1,w,h)	Draws a rectangle
EditMaskedField(x,y,font,col_name)	Each of this function enables to edit a value in pointed column at current record; during creating Application Program you must remember to match function with column type.
EditReadField(x,y,font,col_name)	
EditChoiceField(x,y,font,col_name)	
EditGlobalField(x,y,font,col_name)	
EditDataField(x,y,font,col_name)	
EditTimeField(x,y,font,col_name)	
EditGlobal((x,y,font,gl_no,gl_mask)	Edit Global according to given mask
IconDelete(x,y,db_ident,oper_el)	Place icon deleting a record
IconFindCol(x,y,col_name,oper_el)	Place icon 'finding a record in table'
EditCounter(x,y,font,counter_no)	Set the maximum value for counter
SetDate(x,y,font)	Set Date
SetTime(x,y,font)	Set Time
SetPin(x,y,font)	Set Pin
SetBTName(x,y,font)	Set device name in Blue Tooth
SetBTOnOff(x,y,font)	Turn Off/On Blue Tooth
SetBTPin(x,y,font)	Set a pin for Blue Tooth
IconBT(x,y)	Put a icon of Blue tooth
Special Screen Functions	
EditReadNew(x,y,font,col_name)	Gets transponder number by TR button

Function syntax	*Description*
EditReadSeek(x,y,font,col_name)	Gets transponder number by TR button
IconAddRec(x,y,font,db_ident,oper_el)	Place icon creating a record
EditNewField(x,y,font,col_name)	Creates a new record with placed value
IconConfirm(x,y,operation_el)	Place confirmation icon for creating record.
EditSeekField(x,y,font,column_name)	Seeks for value in column and moves to found rec.
Control Function	
Copy(db_ident)	Copies current record to memory
Action Function	
FillExp(col_name,expression)	Place a value of expression to pointed column
ShowMessage(col_name)	Prints message on HHR's LCD
Paste(col_name)	Pastes a value to field (required copy in control area)
PutGlobal(col_name,gl_no)	Place a global in field
Sorting(col_name)	Define sorting column at current macro
Join(t_col_name,l_col_name)	Creates a relation between Log and Table
FillAddRec(db_ident)	Creates a record at pointed table
Beep(col_name,counter_no)	“Beep” when amount of values in col > value counter
Empty()	Sets the empty record at entering macro
AddRecConf(db_ident)	Creates a record which needs confirmation
Special Action Functions	
ReadNew(col_name)	Define Transponder read button
ReadSeek(col_name)	Define Transponder read button
AddRec(db_ident)	Creates new record
Read()	Read transponder number and place it in buffer
Expression	
count(col_name|db_ident.*)	Value of non-empty record at given column
incCount(col_name)	Value of non-empty record at given column
date()	Current date
time()	Current time
noofrec(db_ident)	Number of records in Table or Log
mFull()	The amount of full memory
mEmpty()	The amount of free memory

Parameters description:

- x – the horizontal coordinate on the screen, take values 0 – 127.
- x – the vertical coordinate on the screen, take values 0 – 63.
- col_name – name of column declared in TABLE or LOG section.

- font – size of font used to present data, take values 0 – 5.
- db_ident – identifier of database on which you want to operate.
- gl_no – nmber of global on which you want to operate.
- oper_el – a value defining where frame should move after performing a function. 0 - stay on function operation element; 1 – move to previous operation element; 2 - move to next operation element
- gl_mask – define mask of entering global, mask declared according to mask rules in TABLE section.
- t_col_name – name of column declared in TABLE section.
- l_col_name – name of column declared in LOG section.
- text – a text which you want to be displayed.
- expression – sort of function which return value, list of expressions is included in table above.
- x1, y1 – top-left point of line or rectangle, x1 - 0..127 y1 - 0..63
- w, h – width and height of line or rectangle, it can take w – 0..127, h – 0..63

Appendix C : Log

Log is a 2nd table which user can use to save information in database about events, or make space-saving records which contain small amount of information.

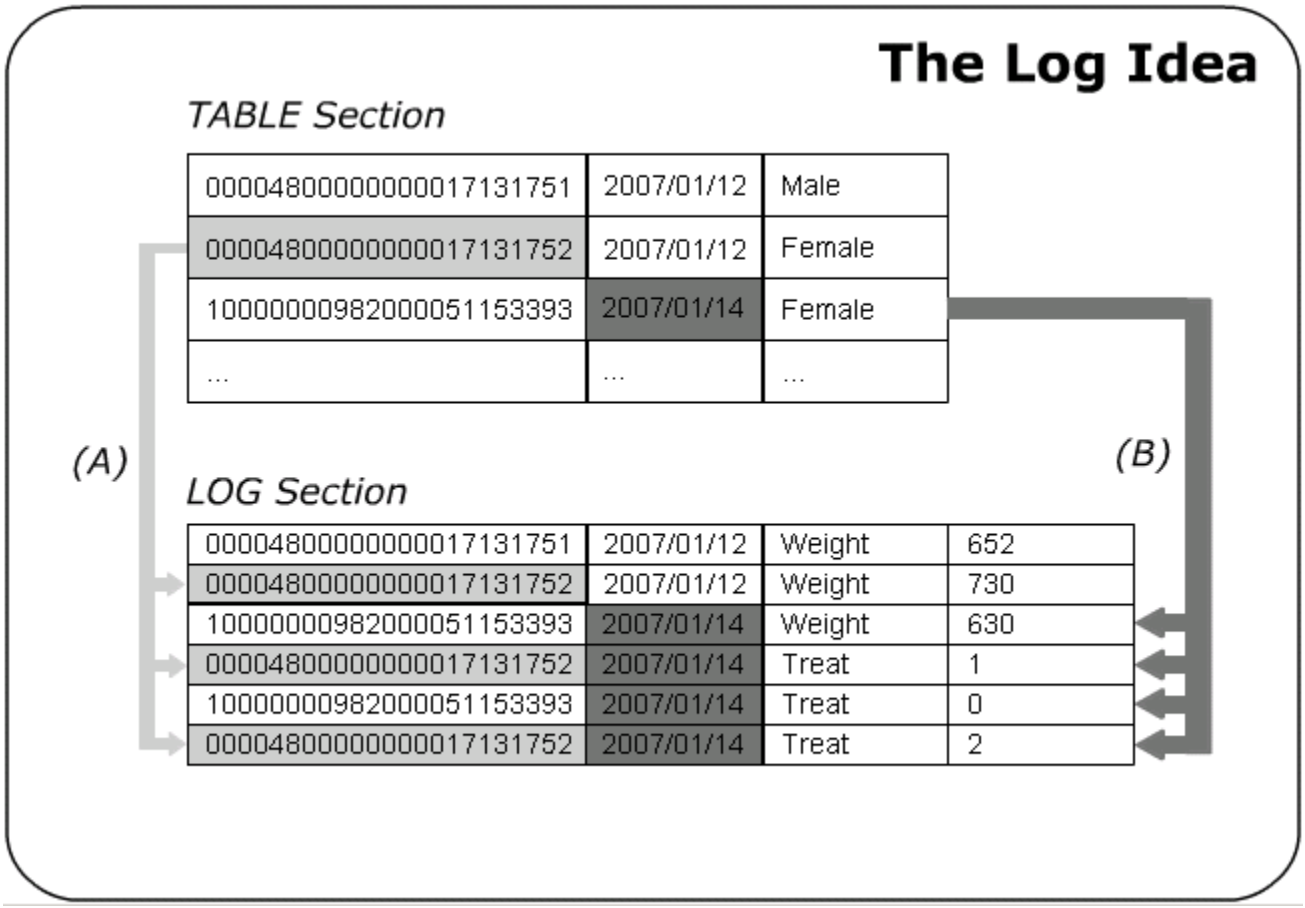
Log is freely definable similarly to Table it takes 7 parameters, the way to define column in Log is analogical to defining columns in Table, refer to Table Section chapter.

> ***The syntax of any line in LOG section.***
>
> ```
> <col_name max 12ch>;<col_type 1ch>;<unique 1ch>;<edit 1ch>;
> <null 1ch>;<col_mask max 25ch>;<default_val max 25ch>;
> ```
>
> It is important to make **default value match declared mask**.

The idea of log has been found to help you operate on information about events(user-defined) concerning one specified column in TABLE section (ex. Transponder number). In the other words you are able to see one record from TABLE and appropriate record from LOG, there is a possibility to display next or previous records from LOG concerning selected record in TABLE. If you are familiar with database technique Table and Log are related by 1 to ∞.

Creating relation between columns in Table and Log is managed by Join function described in Function Assembly.

HHR enables you to display all the data from Log according to relation(Join function), you can manage displaying process by Sorting function, the function is described in Function Assembly.



Flow chart 1.

Table and Log definition for Flow chart 1.

```
TABLE
  MOTHEREID;R;1;1;1;0000 0000 0000;;
  DATE;D;0;1;1;0000000000;;
  SEX;S;0;1;1;######;;
END

LOG
   MOTHEREID;R;1;1;1;0000 0000 0000;;
   DATE;D;0;1;1;00000000000;;
   EVENT;S;0;1;1;#######;;
   VALUE;S;0;1;1;####;;
END
```

The relations are defined by Join function in Action Area, there can only be one Join function in each macro.

The syntax:

```
Join(table.col_name,log.col_name)
```

Join function connects column from Table to column from Log, this connection will be used by HHR to compare the values and operate on records with equal value in indicated columns. The columns given as parameters must agree with type according to the flow chart 1:

- A relation: both columns are transponder read type (R)
- B relation: both columns are date type.

Example:

```
Join(table.MOTHEREID,log.MOTHEREID)
Join(table.DATE,log.DATE)
```

Join function automatically update Joining column at current record, for example if we would get a transponder number and place it in table (ReadSeek(table.col_name) after that jointed Table and Log with Join Function, and created a new record in Log(FillAddRec(log)) then new record in Log would appear with just read transponder number.

Example:

```
ReadSeek(table.EID)
Join(table.EID,log.LEID)
Sorting(log.*)
FillAddRec(log)
//now column LEID in Log in current record number
//is fulfill with transponder number read by
//ReadSeek function
PutGlobal(log.EVENT,1)
FillExp(log.DATE,date())
FillExp(log.TIME,time())
```

According to example above **ONLY AFTER JOIN** function we can automatically put data to Log.

Of course you can use different types of columns in Join function but this variant is used only for very specific needs, and its narrow-range usable. Remember that you must consider all the values which may appear in columns and sense of connecting column in such case.

You can organize way of displaying data from Log by Sorting function, this function defines way of using Up Arrow and Down Arrow buttons.

Function syntax:

```
Sorting(col_name)
```

This function sort displayed data according to alphabetical(ASCII) order, pressing Down Arrow will make HHR move to bigger value(ASCII order) in indicated column, and Up Arrow will make HHR move to smaller value(ASCII value).

Example:

```
Sorting(log.DATE)
Sorting(log.EVENT)
Sorting(log.*)
```

At first example of Sorting function the first record from Log that user will see will be containing the smallest data, by smallest I mean that 2007/01/12 is smaller than 2007/01/14, because “2”(last digit) is smaller in ACSII enumeration then “4”. During comparing values the first character is the most important. In second example first will be displayed columns with “Treat” value and after that records with ”Weight”.

If you want to display data in order like it is in memory put a “*” instead of column name.

Function Sorting may be used without Log, this function define actions taken when buttons are pressed and it is usable even if user is operating only on Table.

The Log Rules

To enable Log in HHR:

1. Place “Log_true” in HEADER section.
2. Place LOG section with column definition.
3. In every function requiring column name place database identifier, a “.” and column name.

Example:

```
HEADER
  ELDA 01a
  2007/01/29
  18.34.59
  dd/mm/yyyy
  hh:mm:ss
  ;
  LOG_true
  ,
  BT_false
  GL_file
END


TABLE
  EID;A;1;1;1;#### ####;;
END

LOG
   LEID;A;0;1;1;#### ####;;
   EVENT;G;0;1;1;00;;
   VALUE;S;0;1;1;##;;
   TIME;T;0;1;1;000000000;;
   DATE;D;0;1;1;0000000000;;
```

```
END

MACRO
begin_macro:VIEWLOG
      begin_action_area
            ReadSeek(table.EID)
            Sorting(log.*)
            Join(table.EID,log.LEID)
    end_action_area

      begin_screen
          PrintText(0,0,0,"LOG:")
            PrintExp(21,0,0,noofrec(log))
            PrintText(105,0,0,"3.2.1")
            PrintText(53,0,0,"OF:")
            PrintExp(69,0,0,count(log.*))
            DrawLine(0,6,127,6)

           PrintText(0,7,2,"TEID:")
            PrintField(40,7,2,table.EID)
            IconFindCol(75,50,table.EID,0)

            PrintText(0,17,2,"LEID:")
            PrintField(40,17,2,log.LEID)

            PrintText(0,27,2,"Event:")
            EditGlobalField(40,27,2,log.EVENT)

            PrintText(0,37,2,"Value:")
            EditMaskedField(40,37,2,log.VALUE)

            PrintText(0,47,2,"Time:")
            EditTimeField(40,47,2,log.TIME)

            IconFindCol(75,50,table.EID,0)
            IconDelete(100,50,log,0)
            IconAddRec(53,50,log,0)
            DrawLine(0,63,127,63)
      end_screen

    begin_control_area
    end_control_area
end_macro
END
```

Using Log

You can manage Log on HHR by two functions:

- Sorting
- Join

Join function defines relation between Table and Log, if you change a current record number in Table, current record number in Log will change too.

If you create a new record in Table and fulfill column given as parameter in Join function,

Without using this function displayed data form Log wouldn't have been connected with current record from Table.

Function Sorting organize way of displaying data on HHR's LCD by this function we can display all records from Log (according to Join function) by switching records with Up and Down Arrow buttons. Or otherwise we can display all records from Table with Up and Down Arrow buttons.

The first circumstance of using Sorting function is presented in example above and to make second one work it is enough to change parameters in Sorting function to :

```
Sorting(table.*)
```

To help you better understand Log and Table integration look back at the Flow chart 1 at the begging of this appendix.

The “A” relation connects two R-type columns, the Sorting function will organize sequence of displaying records from Log. For the “B” relation two D-type columns are connected and again displaying date will be organized by Sorting function.

For each of relation above Sorting after some column in Table is available, then you could change current record number in table without changing current record number in Log.

Example:

```
MACRO
begin_macro:VIEWLOG
      begin_action_area
            ReadSeek(log.MOTHEREID)
            Sorting(log.*)
            Join(table.MOTHEREID,log.MOTHEREID)
    end_action_area

      begin_screen
          PrintText(0,0,0,"LOG:")
            PrintExp(21,0,0,noofrec(log))
            PrintText(105,0,0,"3.2.1")
            PrintText(53,0,0,"OF:")
            PrintExp(69,0,0,count(log.*))
            DrawLine(0,6,127,6)

           PrintText(0,7,2,"Table EID:")
            PrintField(40,7,2,table.MOTHEREID)
            IconFindCol(75,50,table.MOTHEREID,0)

            PrintText(0,17,2,"Log EID:")
            PrintField(40,17,2,log.MOTHEREID)

            PrintText(0,27,2,"Event:")
            EditGlobalField(40,27,2,log.EVENT)

            PrintText(0,37,2,"Value:")
            EditMaskedField(40,37,2,log.VALUE)

            PrintText(0,47,2,"Time:")
```

```
                EditTimeField(40,47,2,log.DATE)

                IconDelete(100,50,log,0)
                IconAddRec(53,50,log,0)
                DrawLine(0,63,127,63)
          end_screen

      begin_control_area
      end_control_area
end_macro
END
```

Appendix D : Bluetooth Mode

HHR is able to use Bluetooth Mode to send a frame containing just read transponder number current date and time. To enable Bluetooth mode it is necessary to:

- **BT_true** – set BT_true flag in HEADER section,
- **BT functions** - use BT function in macro to send data with Bluetooth protocol. The BT functions are:
    1. Read;
    2. ReadNew;
    3. ReadSeek;
    4. EditReadNew;
    5. EditReadSeek;

- **Activation** - before sending any data through Bluetooth you must activate a Bluetooth in HHR with SetBTOnOff()

The most important, if you want to use Bluetooth mode, is to set BT_true and use BT function, because only this function sends a frame containing transponder number, current date and time.

The BT function is special action function and special screen function each of those define Transponder Read button to work, so there if there is BT_true in Header and Bluetooth is activated HHR will send a frame containing information about just read transponder number. More detailed information about function used with BT you can find in HHR 3000 PRO Functions Assembly.

The activation of Bluetooth is essential for HHR to send data through Bluetooth, if user won't activate it the same data will be send through USB protocol on programming connector. Therefore it is the best to make a small menu containing the function which will activate Bluetooth. Apart from it you can define HHR BT name or BT Pin or HHR icon. The functions which enable making it are:

- SetBTOnOff;
- SetBTName;
- SetBTPin;
- IconBT.

If you are interested in detailed reference to those function refer to Function Assembly document.

Below, a fragments of code enabling Bluetooth will be presented

Example:

```
HEADER
  EXAMPLE APP 02BT
  2007/02/28
  10:34:59
  dd/mm/yyyy
  hh:mm:ss
  ;
  LOG_false
  .
  BT_true
  GL_section
END
.
.
MACRO
 begin_macro:SENDBYBT
      begin_action_area
            Read()
      end_action_area
      begin_screen
            PrintText(0,0,1,"ID National:")
            PrintCountryNumber(100,0,1,#)
            PrintAnimalNo(2,9,4,#)
            DrawLine(0,23,127,23)
            PrintText(0,25,1,"Retagging:")
            PrintRetagging(117,25,1,#)
      end_screen
      begin_control_area
      end_control_area
 end_macro

begin_macro:BLUETOOTH
      begin_action_area
      end_action_area

      begin_screen
            PrintText(0,0,0,"RECORDS:")
            PrintExp(46,0,0,count(*))
            PrintText(115,0,0,"2.1")
            DrawLine(0,6,127,6)
            PrintText(0,10,2,"BlueTooth:")
            SetBTOnOff(67,9,2)
            PrintText(0,26,2,"Name:")
            SetBTName(43,26,2)
            PrintText(0,42,2,"Pin code:")
            SetBTPin(55,42,2)
            DrawLine(0,63,127,63)
      end_screen

      begin_control_area
      end_control_area

 end_macro
END
```

```
.
.

MENU
      M0:{New Animal,M1};{Attentions,M2};{Ins.
Bolus,M3};{View/Edit DB,M4};{ReadAndSend,M5};
{Settings,M6}
      M1:NEW_REC
      M2:ATT
      M3:INSBOLUS
      M4:VIEW
      M5:SENDBYBT
      M6:{Bluetooth,M7};{Date/Time,M8};{Farm & Vet,M9}
      M7:BLUETOOTH
      M8:DATE
      M9:GLOBAL
END
```

The example above is a fragment from an example Application Project “EXAMPLE APP 02BT.txt”.

The most interesting for use are **BLUETOOTH** Macro and **SENDBYBT**, which are presented in example, **SENDBYBT** simply send just read transponder number, **BLUETOOTH** Macro enables to activate Bluetooth, set a name for HHR in Bluetooth network and BT Pin. Those Macros has been placed in MENU section, the menu elements important for us has been made bold.

Bluetooth Frame Structure

Bluetooth frame consist 3 values: transponder number, current date, current time;

- First character - “|”
- Transponder number which take 23 characters.
- Table separator declared in header section
- Current Date
- Table separator declared in header section
- Current Time
- End line characters, at C/C++ language it is “\n” (it take two characters 0x0D and 0x0A)

Frame Example:

|12345678901234567890123;2007-02-06;23:59:59\n

Establishing Connection HHR-PC

To establish connection HHR-PC, initialize Bluetooth device connected to PC(PC is a host device), check on which virtual COM port Bluetooth has mounted itself, next connect with Hyper Terminal to HHR on Bluetooth COM port.

The Settings: 115200 baud, 8 data bit, 1 stop bit, No parity, Hardware Flow Control Enabled.