Release date:	March, 5th 2001
Revision:	2.0

TR40xx Communications Protocol

This document describes communications protocol supported by
TR4020 (V3.20) and TR4030 (V3.30) Online Time Recorders

Help us improve this document!
Send your comments and suggestions to feedback@proxdata.com

Revision history:

August, 22nd 2000	Original document release
March, 5th 2001	V3.30-a - Added TR4030 info

ProxData

Table of contents

Section 1. Introduction

1.1. Scope of this Manual

This Manual describes networking with and communications protocol of a TR4020 and TR4030 Online Time Recorders. Collectively, the TR4020 and the TR4030 (but not the TR4000) will be referred to as "TR40xx" throughout this Manual whenever possible.

The TR4020 and the TR4030 are an improved versions of a TR4000 Time Recorder with built-in Floppy Disk Drive. While retaining all the features of its predecessor, both Terminals offer online communications capability: the TR4020- via RS232 ports, the TR4030- through the built-in Ethernet and RS232 ports.

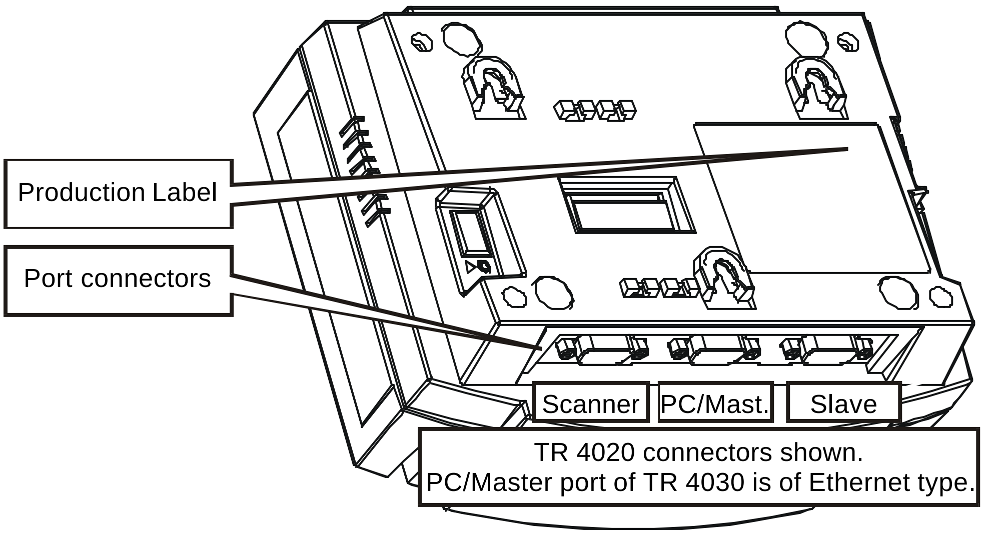
To make use of the TR4020's new hardware features, you need to have the TimeRec firmware version 3.2 or higher loaded into the Terminal. The TR4030 needs the firmware version 3.3 or higher. Apart from being able to handle online communications, new firmware has a lot of functionality improvements (see *TR4020/4030 User's Manual* for details).

The new firmware auto-detects the hardware configuration and is able to work with original TR4000 as well (with all network-related features disabled). However, online communications is only possible if this firmware is running on the TR40xx. Thus, two conditions must be met in order to be able to communicate with the Terminal online:

- You must use the TR40xx (not TR4000)
- Your TR4020 must be loaded with the TimeRec3.2 firmware (or later). Your TR4030 must be loaded with the TimeRec3.3 firmware (or later).

1.2. Checking the Terminal's model number.

TR4000, TR4020, and TR4030 are supplied in the same housing. Exact model number can be checked by reading a production label on the back of the unit. The TR40xx and the TR4030 also differ from the TR4000 in that they have port connectors located on their connector plate (see the drawing below). Connector plate of the TR4000 is empty.


Production Label
Port connectors
Scanner
PC/Mast.
Slave
TR 4020 connectors shown.
PC/Master port of TR 4030 is of Ethernet type.

1.3. Checking currently loaded firmware version

Current firmware version is displayed on startup, when you switch the TR40xx on. The startup screen looks somewhat like this:

```
MONITOR v2.0
Starting app:
TimeRec V3.20
Please, wait…
```

Firmware version is printed on the third line (`TimeRec V3.20` in this example). For communications to work correctly, you need to have V3.20 or higher (i.e. "3.21", "3.30", etc..) for the TR4020 and V3.30 or higher for the TR4030. If this condition is not met, then you must upgrade your TR40xx's firmware.

1.4. Upgrading your Terminal’s firmware

ProxData provides firmware upgrades at no extra charge. All new versions are posted at our website (www.proxdata.com). You may also request to receive the latest firmware version via e-mail. Contact ProxData at info@proxdata.com or Giga-TMS at gigatms@ms3.hinet.net. Upgrade process is described in details in the *TR4020/4030 User’s Manual*.

Section 2. Networking with the TR40xx

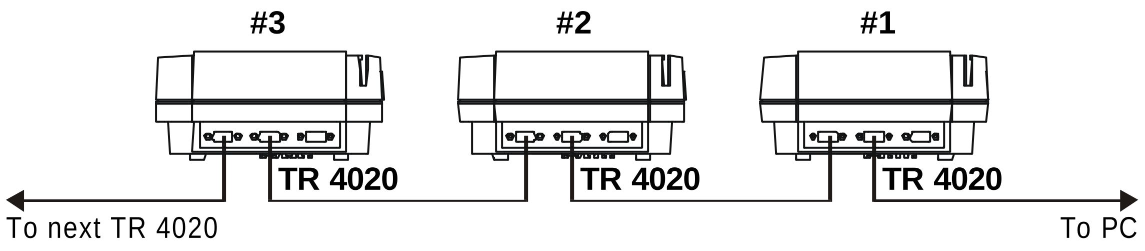
2.1. Daisy-chain network

TR40xx Terminals are interconnected in a so called daisy-chain manner. Each Terminal has two ports reserved for communications: one labeled “PC/Master” and another one labeled “Slave”. Third port (“Scanner”) is used for external scanner (reader) attachment and has nothing to do with online communications. On the TR4020, all three ports are of RS232 type, while the PC/Master port of the TR4030 is an Ethernet port and remaining two ports are of RS232 type.

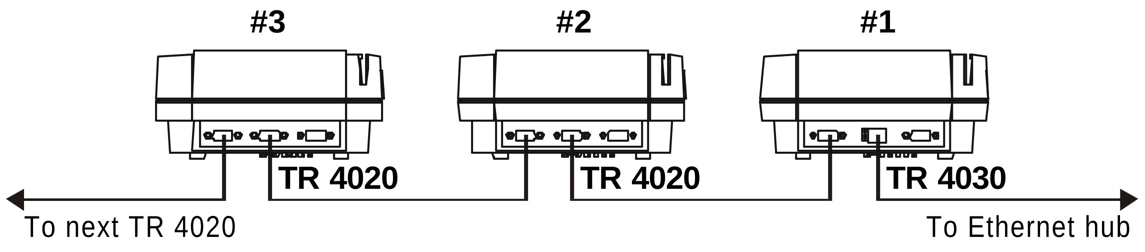
When viewed from the top, the TR40xx has its PC/Master port in the middle of the connector plate while the Slave port resides on the left, as shown on the drawing below.

There are two possible network topologies- one involving the TR4020 Terminals only; and the one comprised of a single TR4030 and several TR4020s.

TR4020-only network. On a daisy-chain network consisting of TR4020 Terminals only, the serial cable runs from the Slave port of one Terminal to the PC/Master port of the Next Terminal. PC (Host) computer connects to the PC/Master port of the very first Terminal like shown on the picture below.



TR4030+TR4020 network. On the network with TR4030 Terminal, the final connection between the Terminal #1 and the PC is via the Ethernet Local Area Network (LAN) network as shown on the drawing below. The TR4030 is effectively plays a role of a “gateway”- it routes the data from the PC to itself and the daisy-chain RS232 network “behind” it. The data arrives to the TR4030 in an encapsulated form- it is embedded in the standard UDP datagram (UDP protocol is a part of the standard set of TCP/IP communications protocols).



With the daisy-chain network, you needn’t assign a unique network number (network address) to each Terminal on the network (this is not to be confused with the IP-address of the TR4030- it still must be assigned one unused IP-address for the LAN to work correctly). The TR4020 communications protocol is organized in such a way that the Terminal closest to the PC on the TR4020-only network (or the TR4030 on the TR4030+TR4030 network) has a network number of 1 (rightmost Terminal on the drawings above). Next Terminal on a daisy-chain is #2, then #3 and so on.

Because of this automatic address resolution, the TR4020 daisy-chain network is free from network number conflicts that plague “bus” networks like RS485 (having two Terminals with the same network number typically leads to communications conflicts because both Terminals are trying to respond to commands from PC at the same time). The tradeoff is that the number of Terminals on a daisy-chain network cannot be very large. Each “leg” of the network introduces a small delay in data transmission (about 0.7ms one way). The further away the addressed Terminal is from PC (LAN), the longer it takes for the command to travel to it and for the reply to get back. The maximum practical number of Terminals on one daisy-chain network has been determined to be around 20 which should be enough for most reallife applications. Nevertheless, the maximum number permitted by the TR40xx communications protocol is 72 and even this can be increased upon request.

2.2. Communications parameters and wiring

Communications parameters for the RS232 daisy-chain network are fixed at **38400-8-N-1, RTS/CTS handshaking** and cannot be changed (to avoid possible malfunction caused by two Terminals being preset differently). Note that RTS/CTS handshaking is necessary. Other lines (DTR, DSR, etc.) are not used.

Communications cables used must be 9-pin, Male-to-Female, and of "direct" type. "Direct" means that pin 2 on the Male side is connected to pin 2 on the Female side. Same for pins 3, 7 and 8. Here is the interconnection table:

Male side (DB9 conn.)	Female side (DB9 conn.)
Pin2	Pin2
Pin3	Pin3
Pin5	Pin5
Pin7	Pin7
Pin8	Pin8

The PC/Master port of the TR4030 is a standard UTP Ethernet port.

2.3. Using a terminal software to manually test communications

The TR4020 communications can be tested with a standard "terminal" software. This is the software that allows you to send and receive data via PC's COM port. Quick-testing the TR4030 is quite complicated as it involves sending and receiving UDP datagrams. There is no universally available software that can perform this function.

Popular terminal programs are **Term95**, **QMODEM**, and **HyperTerminal**. The latter is especially widespread because it comes as a standard part of a Windows95/98 package. For this reason all examples below assume that you are using **HyperTerminal**. It can be found in a **Start/Programs/Accessories/Communications/HyperTerminal** of your Windows95/98. If it is not there, then you need to install it. Open **Start/Settings/Control Panel** and choose **Add/Remove Programs** (be sure to have your Windows distribution handy, you will be asked for it). In the **Add/Remove Programs Properties** click on **Windows Setup** tab. Next, choose **Communications** from **Components** list and press **Details**. In the **Communications window**, select **HyperTerminal** (it must be "checked") and press **OK** twice. You will be asked to insert Windows95/98 CD if necessary. The rest of the installation will be finished automatically.

Once the **HyperTerminal** is on your System, perform the following steps:

- Attach at least one TR4020 to the free serial port of your PC. Note, that the serial cable should run to the PC/Master port of the TR4020 (it is the middle connector, see the drawing in Section 2.1)
- Switch the TR4020 on
- Launch **HyperTerminal**
- When the **Connection Description** dialog opens, type any string (1 character minimum) and press **OK**
- When the **Connect to** dialog opens, select an appropriate COM port from the **Connect Using** drop-down box (for example, **"Direct to COM1"**)
- When the **COM*x* Properties** dialog appears, set communications parameters as follows: **Bits per second:** 38400, **Data bits:** 8, **Parity:** None, **Stop bits:** 1, **Flow control:** CTS/RTS. Click **OK**- the **HyperTerminal's** main window will appear
- Choose **File/Properties** from the main menu, the **Properties** dialog will open
- Click on the **Settings** tab, then press **ASCII Setup** button- the **ASCII Setup** dialog will open
- In the **ASCII Setup** dialog, make sure that 2 check boxes are selected (checked): **Echo typed characters locally**, and **Append line feeds to incoming line ends**
- Press **OK** twice to close dialog windows.

Now you are all set to test communications with the TR4000. Type the string below and press **<Enter>**. Make sure your input entire string correctly, without any corrections- the

TR4020 does not understand keys like **<Backspace>**. If you make a mistake, start over from the beginning of the string. Here is the test string:

`●11Eteststring`

The first character is a so-called STX (start of transmission). You send it by pressing <**Ctrl>-<B>**. STX appears on the **HyperTerminal's** screen as a little smiley face. It is used to indicate the beginning of a *data packet*. The end of packet is marked by the CR character. This character is invisible and is send every time you press **<Enter>**. Thus, actual data packet sent to the Terminal is:

STX	1	1	E	t	e	s	t	s	t	r	i	n	g	CR

If you've set and typed everything correctly, then a *reply* from the Terminal should appear under your *command* as follows:

`●11Ateststring`

Just like your *command packets*, *reply packets* begin with STX and end with CR. Although invisible, the CR is still there.

So, what does the above message exchange means? Immediately following the STX character in your command packet is the *destination address*. "1" means that the Terminal closest to the PC is addressed (first Terminal). Next character is a *source address*. It should be set to exactly the same value as the destination address. Next, there is a *command code* ("E"). Command codes can consist of one or two characters. This particular command means "Echo". When the TR4020 Terminal receives this command, it sends back whatever data is found between "E" and CR. In the above example, the "teststring" string followed. That is how this string is also present in the reply from the Terminal.

What about reply packet? Immediately following STX character is a destination byte. It is supposed to be "1". Next, there is a source byte that must also equal "1". Details of why this is so are provided in the Section 2.4. Next goes reply status code. It informs you of the command processing status (result). "A" means that command was completed successfully. The rest of the packet is a copy (echo) of the original data you have sent.

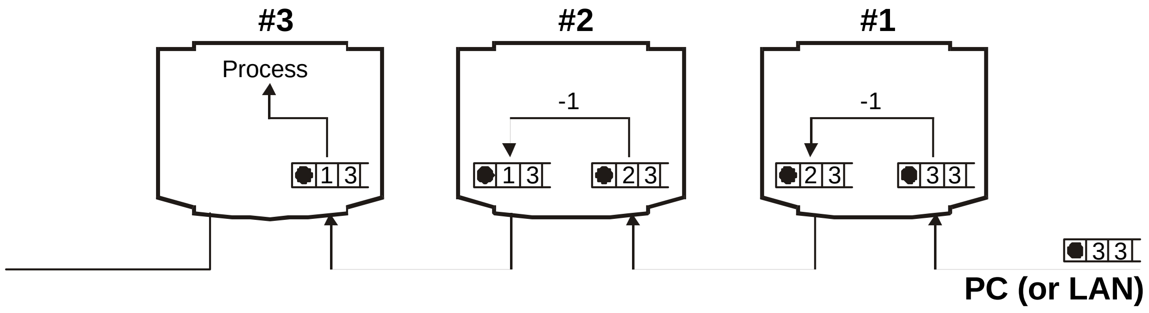
Echo command doesn't do much of a useful work. It is there solely for testing purposes.

2.4. Addressing on a daisy-chain network

If you are wondering by now why destination and source addresses are like they are (especially in reply packets) then look no further- this Section will provide a detailed explanation for this.

As mentioned earlier, the network numbers for TR40xx Terminals need not be assigned- they "happen naturally". The Terminal closest to the PC on the TR4020-only network or the TR4030 on the TR4030+TR4020 network always has the network number of 1. Next Terminal is 2, and so on (up to 72 Terminals).

Here is how addressing mechanism works. Each command packet has a destination address (addresses start from 1). Supposing, there is a command packet that must reach the Terminal #3 (see the drawing below):



Original command packet from the PC will have its destination address set to 3 (once again, destination address is the character immediately following STX, which is represented by "●" here. Also shown is a source address, but it is not used until reply needs to be sent back to the PC.

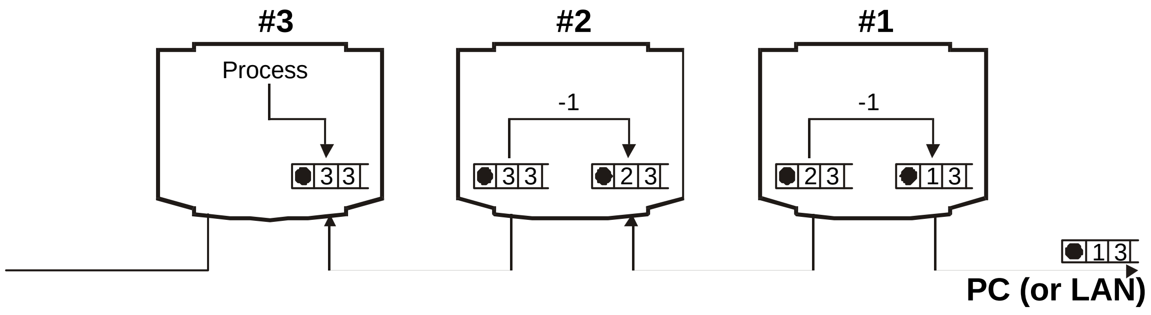
The algorithm each Terminal follows when having received the command packet from the PC (Master) is as follows: if the destination address of the packet equals 1, then the packet is intended for this Terminal. If it is more than one, then this packet must be retransmitted (routed) to the next Terminal. In the example above, the Terminal #1 receives the packet, evaluates the destination byte (3) and finds out that this packet must be retransmitted. The Terminal then sends this packet out through its Slave port. Prior to doing this, the Terminal #1 *subtracts 1 from the destination address*. Hence, when Terminal #2 receives this very packet, the destination address equals 2 already.

On having received the command packet, the Terminal #2 evaluates the destination address, finds out that this packet must be retransmitted, subtracts 1 from the destination address, and sends the packet out through its Slave port.

By the time the Terminal #3 gets the packet, this packet's destination address is already reduced to 1. Therefore, Terminal #3 accepts this packet for processing. This is how command packet with destination address 3 gets processed by the Terminal #3.

The situation with reply packets is similar. When preparing a reply, the Terminal #3 copies the source address of the command packet into the destination address of the reply packet. It also copies the same number into the source address of the reply packet. Source address of the reply packet is not used for routing, but it can be utilized on the PC side to check whether the reply packet have really originated from an addressed Terminal.

Here is how the reply packet travels back to the PC:



As reply hops from Terminal to another Terminal, its destination address is getting decreased at each leg. By the time it reaches the PC, the destination address is supposed to be 1. Source address of the reply packet remains the same- it equals the destination address of the original command packet, reply to which this reply packet represents. This fact can be used by the PC software as an additional check measure. If the reply from the Terminal contains unexpected destination and source addresses, then this packet should be discarder as erroneous. To summarize:

- Destination address of the reply packet must always be 1
- Source address of the reply packet must always equal destination (and source) address of the command packet

2.5. Broadcasting packets

So far, nothing has been said about what happens if you send out a command packet with its destination address set to 0. This is a so-called broadcasting address. Such broadcasting packets are processed by all *reachable* Terminals on the network. Commands sent in broadcasting mode are never replied to (having Terminals reply to them would lead to many Terminals talking at the same time).

Because broadcasting commands are never replied to, their use is somewhat limited. Obvious limitations are:

- You cannot receive any data from any Terminal when you are using a broadcasting address. Again, this is because Terminals never reply to such packets (although they *do* process them)
- Next, you cannot even make sure that your broadcast has reached all Terminals on the network (again, because there are no replies from individual Terminals).

Broadcastings commands can be used when you need to perform some action quickly- they spare you from the necessity to send the same command to all Terminals one by one.

This can come handy- especially if you are typing all commands manually, using a software like **HyperTerminal**.

Section 3. Command and reply packets format

Now that you are familiar with the general style of the TR40xx communications, it is time to provide a “strict” description of command and reply packets.

3.1. Types of packet encapsulation

All TR40xx daisy-chain packets start with STX character (ASCII code 2, entered by pressing **<Ctrl>-<B>** in **HyperTerminal** and looking like a little smiley face on the PC screen). As for the packet endings, there are two possible choices:

- CR character (ASCII code 13, entered by pressing **<Enter>**)
- ETX (ASCII code 03, entered by pressing **<Ctrl>-<C>** and looking like a little black rectangle approximately half the height of a normal character)

Until now, all examples have used the CR character to mark the packet end. Using ETX differs in that packets with ETX ending are expected to have a *checksum and length fields*. Together, these two fields form four *check characters* that reside in the packet immediately following all packet data and preceding the ETX character. Check characters can be used to verify packet integrity and protect against noise, transient conditions on the network and other disruptions.

Why have two packet encapsulation options? This was done to ease the manual programming of the Terminals while leaving you the possibility of creating a robust PC software:

- When typing the TR40xx commands manually using the **HyperTerminal**, it is not convenient for you to calculate the check character values for every packet you are sending
- When creating PC software, you can go with “protected” packets to improve communications reliability.

As a rule of thumb, the TR40xx always replies with the same type of reply packet as the type of command packet being replied to: if you send a “protected” command packet, then the Terminal will answer with a “protected” packet as well. If you have used a “simple” command packet, then the Terminal will reply with a “simple” reply packet too. There is a **single exception** to this rule, however:

When the TR40xx is in the Programming Mode, it responds with a “Busy” (“B”) status code to any packets sent by the Terminal (see Section 4.2). Reply packets with “Busy” reply code are always of a simple type, regardless of the type of command packet being replied to.

In general, it is not recommended to base your assumptions about reply packet type basing on the type of command packet you send. Your software should always check the last character of the reply packet (CR or ETX) in order to determine reply packet’s type.

3.2. Structure of a simple packet (without check characters)

Simple packets have STX, destination address, source address, packet body, and CR:

STX	Dest.	Src.	Packet body				CR
O2H	‘3’	‘3’	‘E’	‘A’	‘B’	‘C’	0DH
	33H	33H	45H	41H	42H	43H	

Example above shows an “Echo” command for Terminal #3. It is important to understand, that “3” is an ASCII character 3, not a code 03H. Actual code, therefore, is 33H (ASCII code for the character “3”). There is a reason for this: no character between STX and CR can be allowed to occasionally become equal to 02H (STX) or 0DH (CR). If this was allowed, then the receiver of the packet would be unable to distinguish between the special characters marking the beginning and end of packets from characters within packets.

For the above reason (and also to simplify manual input in programs like **HyperTerminal**), both destination and source addresses start from code 30H (ASCII for “0”). Want to address Terminal #6? Send out the code 30H+6H=36H (i.e. ASCII “6”).

As was mentioned earlier, the TR40xx communications protocol allows up to 72 Terminals to be daisy-chained. This means that the destination and source character codes can fall outside of “0” to “9” range. For example, if you need to address the Terminal #12 then the destination character code should be 30H+12=3CH. This is an ASCII code for the"<" character!

This may seem to be extremely inconvenient at a first glance. However, consider the following:

- First 9 Terminals are addressed using convenient "1" to "9" characters. It is highly unlikely that you are playing with a network of more than nine TR40xxs using just **HyperTerminal** and your fingers
- If you are creating some sort of a PC program to handle the TR40xx communications, then this addressing codes are very easy to implement. All you have to do is take the number of the Terminal being addressed and add 30H as a base.

3.3. Structure of a protected packet

Protected packets are different in that they have a checksum and length fields in their body, and that they end with ETX character:

STX	Dest.	Src.	Packet data				Checksum		Length		ETX
02H	'8'	'8'	'E'	'A'	'B'	'C'	'0'	'B'	'0'	'8'	03H
	38H	38H	45H	41H	42H	43H	30H	42H	30H	38H	
			45H+41H+42H+43H=10BH→0BH→'0', 'B'						08→'0','8'		

The checksum field immediately follows the packet data and is calculated as follows:

- First, the codes of all data characters in the packet are summed up. Data characters are all characters except STX, destination, source, checksum and length fields, and ETX
- Next, only the least significant byte (LSB) of the checksum is evaluated, all higher bytes are thrown away
- Finally, a HEX string representation of a checksum byte becomes a 2-character checksum field

For a packet above, the packet data is comprised of "E", "A", "B", and "C" characters. The sum of their ASCII codes is 10BH. The LSB of this value is 0BH. Thus, the checksum field consists of 2 characters: "0" and "B". Once again, these are *not* codes, these are ASCII characters. Actual ASCII codes of these characters are 30H and 42H.

The length field records the length of all fields preceding this field except STX. This includes destination, source, packet data, and both checksum characters. For the example above, the length equals 8. The length is recorded just like the checksum- as a 2-character Hex string: "0" and "8".

3.4. Additional notes on packet integrity protection

Some of our Customers have asked us why we had chosen to protect the data packets with a combination of the checksum and length fields. We were repeatedly told that there are other reliable industry solutions like CRC-16 that can provide very good data protection.

There are two reasons why we haven't settled on a standard CRC-16 check field:

- CRC-16 requires a lot of processing power. The TR40xx would have had to spend as long as 3 milliseconds on a CRC-16 calculation (or verification) of every data packet. This may seem like a very short time, but multiplied by a number of commands/replies needed to download entire database contents, for instance, this would have resulted in an extra minute or two needed to complete the upload task (app. 30% overhead!)
- CRC-16 is a very good choice for situations where random data errors may occur. This is not the case with modern RS232 lines. Under regular Windows environment, the most common error is "data overrun". This means that some packet bytes get lost. Having a combination checksum/length check fields is a fast and efficient way of detecting overruns.

3.5. Maximum packet length and allowable character codes

Maximum command or reply packet length including the "wrap" (STX, ETX/CR) cannot exceed 255 characters. For protected packets (which are longer), this leaves 246 characters for the packet data. All character codes within a packet must be in the 01H…7FH range, excluding STX (02H), ETX (03H), and CR (ODH), since these characters are used for packet encapsulation.

3.6. TCP/IP encapsulation

For the TR4030+TR4020 network, Reply and Command packets travel between the PC and the TR4030 Terminal within the UDP datagram. The TR4030 plays the role of a gateway between the TCP/IP network and the RS232 daisy-chain network “behind” the TR4030. Arriving command packets are stripped of all UDP and IP wraps and sent out to the daisy-chain network. Replies are wrapped and sent out to the PC as UDP datagrams.

TCP/IP encapsulation is straightforward: entire Command and Reply packets including STX, ETX, CR, etc. are transmitted as UDP data. One UDP datagram is generated for each Command or Reply packet.

Section 4. Command groups. Reply status codes

Starting from this Section, the wrap part of all packets will sometimes be omitted from our discussion. This means, that STX, destination, source, checksum and length fields, and CR/ETX will not be shown or mentioned unless necessary. Instead, the discussion will concentrate on the packet data itself. Once again, the packet data is everything between the source address and the CR for simple packets or the source address and the checksum field for protected packets. For the packet below, the data portion is "AABC"

STX	**Dest.**	**Src.**	**Packet data**				**CR**
O2H	**'1'**	**'3'**	**'A'**	**'A'**	**'B'**	**'C'**	**0DH**
			Packet data				

4.1. Command and reply packets

All packets on the TR40xx network are either command or reply packets. Command packets carry the command code and some relevant data if applicable. Reply packets carry the reply status code and also may return some data.

It is quite impossible to discuss the TR40xx's commands without getting familiar with the range of possible replies (i.e. error situation) first. For this reason, all available reply status codes will be explained ahead of commands.

4.2. Reply status codes

Reply status code is a single ASCII character. It tells you whether the command you have sent to the TR40xx was processed successfully and, if not, what was the reason for that. Reply status code always follows the source field in the reply packet. Table below lists all available reply status codes:

A	41H	**Command was completed successfully**. If the kind of command being replied to is supposed to return some data, then this data will immediately follow "A" in the reply packet. "A" is the **only** status code that may be accompanied by some data from the Terminal
D	44H	**Access denied**. This happens when you are trying to change some data that is read-only by nature, or if you are attempting to change a password-protected data without having *logged in* first
F	46H	**Command execution failed**. This may be caused by hardware malfunction or data corruption inside of the Terminal
I	49H	**Invalid command**. Either command code or data portion of the command packet is invalid. This *does not* include check error
C	43H	**Check error**. Command packet's checksum/length field verification failed *on the Terminal side*
E	45H	**End**. This reply status code has two meanings. When you are adding a records to the datatable of the TR40xx, this code means that the table is full and new record cannot be added. When you are retrieving datatable records, this code is generated when the end of table is reached
N	4EH	**Command is not applicable or data not available**. This code is rather difficult to explain in details for now. In general, it means that the data you are looking for does not exist or apply to current situation. It will all become clear when you learn about **Item Lookup** ("IL") and **Item Default** ("ID") commands
K	4BH	**Key violation**. Some datatables may contain *key fields*. No two records in the datatable are allowed to have identical key fields. This code is generated when you attempt to add a record whose key field's content duplicates that of another existing record in this datatable
B	42H	**Busy**. The Terminal is in the Programming Mode and your command cannot be processed. This is because concurrent Terminal programming via Programming Mode and online session is not allowed

4.3. Types of TR40xx commands

All TR40xx commands can be divided into 4 groups:

- **Login commands**. This group is used to login and logout, also get current login status. Login is a process in which you gain access to the Terminal's internal data by supplying a valid login password. Most of the TR40xx's internal data is password protected for write operations. This means that in order to change this data, you need to login first. Some data cannot even be retrieved when not logged in
- **Data retrieval and manipulation**. Commands in this group are used to get the Time and Attendance data accumulated by the Terminal. There are also commands to erase the data, restore it back, and some others
- **Item manipulation**. All TR40xx functioning parameters (like current date and time, login password, bells schedule, default shift, etc…) are referred to as "Items". There is an extensive set of commands that not only allow you to manipulate these Items, but also to "learn" about them. Learning means that there is a way to find out what particular items a given Terminal supports
- **Miscellaneous**. Included in this group are several "other" commands that are rather hard to classify

Sections below discuss all four command groups in details.

Section 5. Login commands

Commands in this group include:

All commands and replies are shown without packet wrap (STX, destination, source, etc.)

Cmd	Usage	Reply	Comment
LI	LI ***pp…p***	A | D | F	**Log In**. Log into the Terminal
LO	LO	A	**Log Out**. Log out from the Terminal
LS	LS	A ***s***	**Login Status**. Check current login status

Pp…p	login password (can be an empty string)
s	login status: "I"-currently logged in, "O"- currently logged out

5.1. Why login?

Most of the TR40xx's internal data is password protected. You need to log into the Terminal first in order to gain access to the protected data. Login is a process in which you gain access to the Terminal ("open" it) by supplying a valid login password.

Having to login onto the Terminal in order to change (and in some cases even view) the data protects the Terminal's functioning parameters and the Time and Attendance data from unauthorized access and/or altering.

Login plays another important function that should not be overlooked. The TR40xx features a so-called Programming Mode, which allows the User to setup the Terminal using its keypad and LCD. For obvious reasons, working in the Programming Mode and accessing the Terminal online at the same time is not allowed (to avoid "competition"- it's when the User at the Terminal and the online User both try to alter the same data). Furthermore, the User at the Terminal has a priority over the online User- once Programming Mode is entered, the Terminal cannot be accessed online- all commands received while the TR40xx is in the Programming Mode are discarded and replied to with a "B" (Busy) code.

Now, when you are accessing the TR40xx online, you cannot prevent somebody else from entering the Programming Mode. What you can do is let the User at the TR40xx know that online session is in progress. With you logged in, the User at the Terminal who is about to enter the Programming Mode will get a warning message first. The message looks somewhat like this:

```
Online session is in progress!
Entering Programming Mode will
abort it. Continue still?
Yes                         No
```

Mind you, the User at the Terminal can still kick you out, but at the very least he will be notified of your presence. For this reason, any online session with the TR40xx should *always* start with login.

5.2. Logging in and out

The TR40xx always powers up in the logged out mode. When exiting the Programming Mode, the Terminal also resets to "logged out".

When not logged in, you may still have a *read* access to *some* of the TR40xx's internal data. To gain full access to the Terminal you need to login first. Use the **Log In** command and a valid password.

You need to login even if your password is not set (i.e. it is an empty string). In this case, you type in the **Log In** command without any password string.

Access denied ("D") reply status code is returned when login password is incorrect. It is also noteworthy, that **Log In** command may fail ("F" reply). This happens when the TR40xx's internal FLASH disk fails or disk data becomes corrupted. Terminal Initialization is the only solution in this case (see TR40xx User's Manual, Appendix D).

Currently, the TR40xx doesn't have login timeout. This means, that once having entered the logged in mode, the TR40xx will remain in it indefinitely, unless switched off or forced into the Programming Mode.

To protect the Terminal's internal data, always finish your online session with **Log Out** command.

5.3. Checking current login status

PC software you are creating may want to check current login status prior to attempting certain commands. This way you can make your software user-friendlier. Login Status command returns "I" in case the Terminal is in the logged in mode, or "O" if otherwise:

5.4. How do I *set* the login password?

Notice that this group of commands does not include any command to set (modify) the login password. This is because login password is just another Item (programmable parameter) that can be accessed and modified using Item manipulation commands (see Section 7).

5.5. Sample online session

Here is how a typical online session involving **Log in** and **Log Out** command might proceed. **Initialize the Database** ("RI") command is used to illustrate the access denied situation:

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

PC:	●**11RI**	*You want to erase entire Time and Attendance database*
T:	●**11D**	*Umps! Access denied*
PC:	●**11LIPass1**	*You login (your password is "pass1")*
T:	●**11D**	*Denied again! This is because the password is case-sensitive!*
PC:	●**11LIpass1**	*Try to login once more*
T:	●**11A**	*OK this time*
PC:	●**11RI**	*Now, erase this database!*
T:	●**11A**	*Finally, it's gone*
PC:	●**11LS**	*Check login status for no special reason*
T:	●**11AI**	*"I"- logged in, as expected*
PC:	●**11LO**	*Log out*
T:	●**11A**	*Done*
PC:	●**11LS**	*Check logout status once again*
T:	●**11AO**	*"O"- currently logged out*

● *is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>*

Section 6. Data retrieval and manipulation commands

Commands in this group include:

All commands and replies are shown without packet wrap (STX, destination, source, etc.)

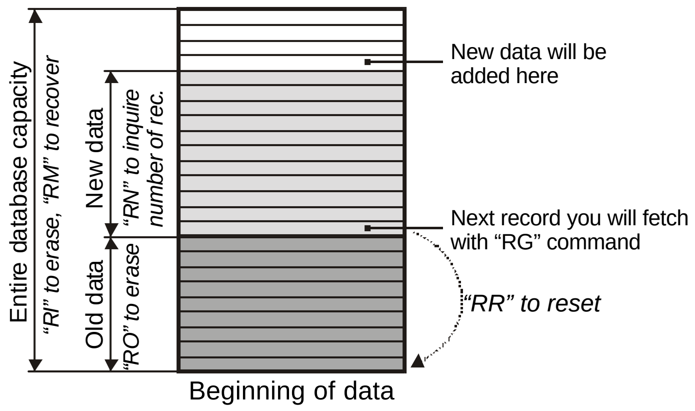
Cmd	Usage	Reply	Comment
RN	LN	A ***nn…n*** | D | F	**Number Of Records**. Open read transaction (if not opened), get the number of records to read
RG	RG	A ***rr…r*** | E | D | F	**Get Record**. Open read transaction (if not opened), fetch next record
RH	RH	A	**Confirm Record**. Confirm record acceptance by the PC
RA	RA	A | D	**Abort Read Transaction**. Close read transaction without updating the database (without marking fetched records as “old”)
RC	RC	A | D | F	**Commit Read Transaction**. Close read transaction and update the database (mark fetched records as “old”)
RO	RO	A | D | F	**Delete Old Records** from the database
RR	RR	A | D | F	**Reset Old Records**. Make all records in the database look as “new”
RI	RI	A | D	**Initialize The Database**. Delete all records and/or restore the database in case it was corrupted
RM	RM	A | D	**Recover The Database**. Make the database appear as if it was 100% full with records

nn…n	number or records remaining to be read in the current read transaction (max. 99999)
rr…r	database record (TAB-delimited fields), see Appendix B for current record format

6.1. Time and Attendance database

The TR40xx stores the Time and Attendance data in its internal database memory. The data consists of records that follow each other in exactly the same order that they were created in. Associated “housekeeping” mechanism “remembers” the total amount of records in the database as well as the amount of “new” records.

“New” data is the data that was added since the last time you have uploaded the database during an online session or copied the data onto the Floppy Disk. When you are uploading the database data (using **Get Record** and **Confirm Record** commands as explained in 6.3), you are only getting the new data by default. Nevertheless, the old data (i.e. the data that you have uploaded online or saved to disk at least once before) is still there and you can re-read it again by “resetting” it. **Reset Old Records** command makes all database data look like “new”. Subsequent use of **Get Record** and **Confirm Record** commands will let you upload entire database contents.


Entire database capacity
“RI” to erase, “RM” to recover
New data
“RN” to inquire number of rec.
Old data
“RO” to erase
New data will be added here
Next record you will fetch with “RG” command
“RR” to reset
Beginning of data

Other commands available allow you to:

- **Delete Old Data** to free up some database space

- **Initialize The Database**. This command completely initializes the database. It is useful in case you want to delete all records or restore database functionality after the database went "bad"
- **Recover The Database**. This command makes the database appear 100% full with records. This is useful when the database goes "bad" and you desperately need to retrieve some records. More details on this command usage are provided in 6.6.

6.2. Read transactions

Data upload takes place in a so-called read transaction. The read transaction is opened automatically on the *first* time you use the **Get Record** command to retrieve the first unread (new) record or the **Number Of Records** command to determine the number of records *still waiting* to be retrieved in the *current* read transaction.

When the read transaction is opened, a database *snapshot* is made. Snapshot means that the status of the database on the moment of transaction opening is memorized. Once the read transaction is opened, the number of records that you can download during this transaction will remain unchanged, even if new records continued to flow into the database while your read transaction was in progress.

You retrieve the database records one by one using the **Get Record** command followed by the **Confirm Record** command. **Get Record** fetches the next unread (new) record of the database. **Confirm Record** tells the TR40xx that the record just fetched was indeed received by the PC. If we represent the database table as a list of records with a *pointer*, pointing at the next unread record in the list, then **Get Record** command's function is to fetch the record currently pointed at by the pointer, while **Confirm Record** command's function is to advance the pointer to the next unread record (but no further than that).

This record-by-record upload method is bulletproof. It makes sure that no records ever get "lost" on the way from the TR40xx to the PC. If your PC fails to receive some record, it may send the **Get Record** command again to retrieve the same record. And the **Confirm Record** command may only advance the pointer *no further* than the next unread record, no matter how many times you repeat this command between two subsequent **Get Record** commands.

You retrieve records "in a loop", until your **Get Record** command is replied to with the "E" (End) status code. This will mean that you have uploaded all new records already. Note, that "E" status code can only be returned as a reply to the **Get Record** command. **Confirm Record** *always* returns "A".

For the database to get updated, you need to explicitly close the read transaction. This is done by using the **Commit Read Transaction** command. When you issue this command, all records that you have uploaded *and confirmed* (!!!) in this read transaction will be marked as "old".

You can also abort the read transaction by using the **Abort Read Transaction** command. Transaction will be closed quietly without marking any new records as old.

Read transaction is also aborted automatically by the following commands: **Delete Old Records**, **Reset Old Records**, **Initialize The Database**, and **Recover the Database**. All of the above commands will safely abort the transaction before executing.

The **Number Of Records** command is not really necessary to use. You may always repeat the **Get Record**-**Confirm Record** commands in a loop until you get the "E" reply status code. However, you may need to know the number of records you are about to upload beforehand. You need this, for example, to be able to display the progress bar on the PC screen (showing the percentage of upload done so far).

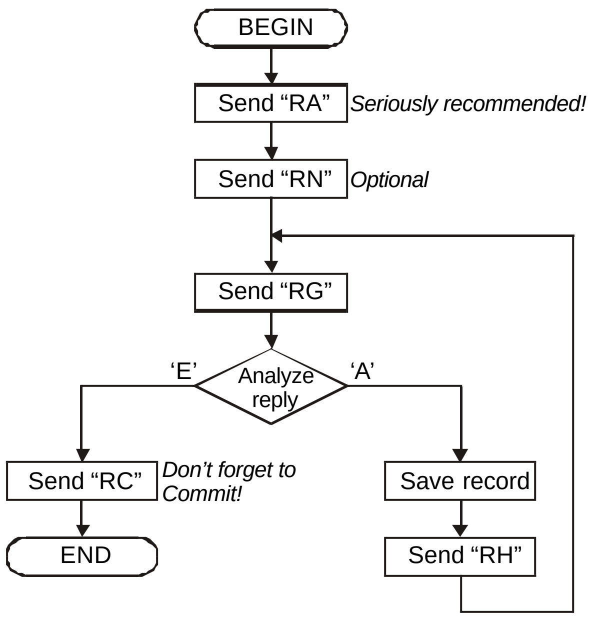
Note, that the **Number Of Records** command always returns the number of *remaining* new records to be fetched in the current read transaction. Every time you get a record, the number is decreased by 1. Therefore, in order to determine the total number if records in the current read transaction correctly, you must *open* the read transaction with the **Number Of Records** command.

6.3. Typical upload scenario

The block diagram below illustrates a typical database data upload procedure. Note, that the first command issued is the **Abort Read Transaction**. This is always a good practice to start from aborting the old transaction. This way you can make sure that your next **Number**

Of Records or **Get Record** command will really open a fresh transaction, not continue the existing one.

The diagram below is simplified. It doesn't show the reply analysis for every command issued to the TR40xx.



Let us now illustrate the above diagram by a sample online session. The session assumes that you are logged in already and that the database contains three records: "record1", "record2", and "record3". We deliberately use a simplified record data. The real data format can change in the future, as TR40xx gets more and more functions. However, data retrieval method will be kept the same. Current database data format can be found in the Appendix B.

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

PC:	●**11RA**	*Abort whatever read transaction might be in progress*
T:	●**11A**	*Done*
PC:	●**11RN**	*Open read transaction, inquire total number of records*
T:	●**11A3**	*Have three records*
PC:	●**11RG**	*Fetch a record*
T:	●**11Arecord1**	*Got first record*
PC:	●**11RH**	*Confirm first record*
T:	●**11A**	*Terminal acknowledged*
PC:	●**11RG**	*Fetch a record*
T:	●**11Arecord2**	*Got second record*
PC:	●**11RH**	*Confirm second record*
T:	●**11A**	*Terminal acknowledged*
PC:	●**11RN**	*Inquire the number of records once again*
T:	●**11A1**	*Notice that the value reflects the number of records yet to be read*
PC:	●**11RC**	*Close the transaction*
T:	●**11A**	*OK*
PC:	●**11RG**	*Reopen the transaction*
T:	●**11Arecord3**	*Got the third record*
PC:	●**11RG**	*Repeat*
T:	●**11Arecord3**	*Still got the same (third) record*
PC:	●**11RH**	*Confirm third record*
T:	●**11A**	*Terminal acknowledged*
PC:	●**11RG**	*Attempt to fetch next record*
T:	●**11E**	*No more records*

PC:	●11RA	*Abort the transaction*
T:	●11A	*OK*
PC:	●11RN	*Inquire the number of records again*
T:	●11A1	*Still one record remaining because last transaction was aborted*

● *is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>*

6.4. Understanding concurrent log retrieval and record adding

The TR40xx allows you to retrieve the database data while new records are being added to the database. Records added past read transaction opening moment will not be visible in the *current* transaction. You need to close (commit or abort) and reopen the transaction in order to be able to see the updated number of records. This mechanism prevents the read transaction from going on indefinitely (due to new record addition).

6.5. Cleaning up the database space

Closing the read transaction updates the read pointer position but it doesn't actually delete old records from the database. If these records are not deleted explicitly, the database will eventually become full and unable to accept new records. Use the **Delete Old Records** command to reclaim database memory space. You may also use the **Initialize The Database** command to delete all database records.

6.6. Database failures and recovery

Despite all safety measures taken to preserve the database content, it is theoretically possible that the database will get corrupted. Internal database housekeeping is able to detect this situation. Should this happen, the Fail ("F") status code will be returned on any database-related command except the ones described below. Corrupted database is inaccessible unless repaired. The database can be repaired in 2 ways:

- **Initialize the Database** command completely re-initializes the database. The database *appears* to have no records after that
- **Recover the Database** command re-initializes the database and makes it *look* like 100% of its capacity is occupied.

Both commands do not modify actual contents of the database FLASH memory in any way. They simply adjust the housekeeping data (read and write pointers, etc…). This is similar to file deletion on the PC: only file record gets modified, the sector data itself is not immediately erased (although it may be overwritten by a fresh data at any time).

As mentioned earlier, **Recover The Database** command makes the database look like it is 100% full. You can subsequently retrieve all the records in the database using the usual **Get Record-Confirm Record** command pair. Naturally, the data retrieved may not be entirely consistent. By the time you start recovery, some new records may have already overlapped the old ones. Some FLASH memory locations may be in the erased state and contain no data at all. Despite these limitations, **Recover The Database** command gives you the possibility to recover at least *some* data. As for the possible garbage output, the TR40xx takes care of this: it checks every record prior to sending it out. If some fields of the record appear to be wrong, then their content is automatically substituted for a default one (see Appendix B for more details).

6.7. How do I get other database statistics (like total number of records)?

The database-related command group only contains a command to inquire the number of new records in the current read transaction. All other statistics are provided in the form of read-only Items and are retrieved just like the values of other TR40xx's parameters (see Section 7 for more details). These database-related Items' values can be read out in a non-intrusive manner. This means that reading them does not abort the read transaction in progress (just like adding new records does not abort the read transaction). Database statistics always reflect the *latest* database situation, not the one that existed on the moment the read transaction was opened.

Section 7. Item manipulation commands

Commands in this group include:

All commands and replies are shown without packet wrap (STX, destination, source, etc.)

Cmd	Usage	Reply	Comment
IC	IC	A ***cc…c***	**Item Count**. Total number of Items on the Terminal
IL	IL ***mm…m***	A ***nn…n*** | I | N | F	**Item Lookup**. Find Item number by name
I?	I? ***rr…r***	A ***ii…i*** | I | N | F	**Item Info**. Get Item's info string
IU	IU	A | F	**Update** Items that work out of RAM
Commands applicable to "Array" Items only (VAL, STR, TIME, DATE types)			
IS	IS ***rr…r ss dd…d***	A | D | I | F | N	**Item Set**. Set the Item to a new value
IG	IG ***rr…r ss***	A ***dd…d*** | D | I | F | N	**Item Get.** Get Item's value
ID	ID ***rr…r***	A ***dd…d*** | D | I | N	**Item Default**. Get default value for this Item
Commands applicable to "Table" Items only (TAB type)			
I*	I* ***rr…r tt***	A ***ff…f*** | I | N | F	**Field Info**. Get info string for one field of a datatable
IT	IT ***rr…r***	A | I | N	**Item Top**. Reset table record pointer to the top of the datatable
IF	IF ***rr…r***	A ***dd…d*** | D | I | E | F | N	**Item Fetch**. Get next record of a datatable
IH	IH ***rr…r***	A | I	**Item Confirm**. Confirm the reception of a datatable record
IA	IA ***rr…r dd…d***	A ***pp…p*** | D | I | E | F | N	**Item Add**. Add record to the datatable
IE	IE ***rr…r pp…p→dd…d***	A | D | I | F | N	**Item Edit**. Edit a datatable record
IR	IR ***rr…r pp…p***	A | D | I | F | N	**Item Remove**. Delete a datatable record

cc…c	Total number of Items found on the Terminal (1-100)
mm…m	Item name (string, case sensitive)
nn…n	Item number (0-99)
rr…r	Item reference. Can be supplied either as a name, or as a number. When supplied as a name string, it must be enclosed in quotation marks (i.e. "TIME", "VERSION"). When supplied as a number, it must *always* be in a 2-digit form (i.e. 01, 15, 99)
Ii…I	Item info string. Consists of several TAB-delimited fields describing Item's properties
ss	Member. Must *always* be in a 2-digit form (i.e. 00, 12)
dd…d	Item's data. It can be a single value or a record string with TAB-delimited fields
tt…t	Field number. Must *always* be in a 2-digit form (i.e. 00, 01)
ff…f	Field info. Consists of several TAB-delimited fields describing one data table's filed
pp…p	Record number. Absolute record position in the datatable

7.1. The concept of Items

The TR40xx has a number of functioning parameters that are referred to as "Items". Each item can be as simple as a single value, or as complex as a database table. Some items are read-only, and some can be written to. Items may also differ in "access level", i.e. whether or not you need to be logged in order to read or modify them.

The TR40xx provides two methods for Item manipulation:

- **Traditional**. You know (by reading this Manual) what Items the Terminal is supposed to have, so you write your software to work with this pre-fixed set of Items. This approach is simple and lets you finish your PC software rather quickly. Unfortunately, it is quite inflexible: you will have to modify the PC software every time some new Items are added to the new release of the TR4000 Terminal

- **Auto-learn**. You use a special set of commands to *learn* about all available Items from the Terminal itself. This way you can create a kind of "Terminal Explorer" that is inherently ready to work with any set of Items. This kind of software is more difficult to create, but it definitely pays to do it: you won't have to make changes every time we update our Terminals!

7.2. Properties of Items

This Section discusses all Items except those of table type. Table Items are covered in 7.8-7.10.

Each Item is characterized by a number of properties, namely its *type*, *size (number if members)*, *access level* and *update* properties.

The following Item types are currently supported (this list may be expanded in the future!):

- **VAL** (value). A value can be in the 0…9999 range
- **STR** (string). A string can have the length from 0 to as many characters as you can fit into the data packet without exceeding the maximum total length of 255 characters. All characters in the string must be in the 20H…7FH range or 09H (TAB character)
- **TIME**. Items of this type return and accept string of a "***hh***:***mm***:***ss***" format
- **DATE**. Items of this type return and accept string of a "***DD***:***MM***:***YYYY***" format
- **TAB** (table). Items of this type contain entire datatables within them. See 7.8-7.10 for complete discussion of datatables.

Each Item (except those of table type) is actually an array. It may contain as many as 100 different values. Each Item is characterized by *Size*, i.e. the total number of members within an Item. A lot of Items consist of a single member.

Access rights flags define read and write (get and set) protection separately for each Item:

- Item may be accessible for read at any time (no login required), only when you are logged in, or never (write-only Item)
- Item may be accessible for write at any time (no login required), only when you are logged in, or never (read-only Item)

Some Items, when used by the TR40xx, are copied from the Terminal's internal FLASH disk into the RAM on startup. Further references to such Items' values are made using their RAM copies. Those Items are said to require *update*. It is not enough to set their new value. A special **Update** command must be executed in order to update these Item's values in the Terminal's RAM.

7.3. Using traditional Item handling method

Let's start right from the example. Supposing, you need to read and set the TR40xx's time. From the Appendix A of this Manual, you learn that there is, indeed, an Item called "TIME". Example session below assumes that you are logged in already:

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

PC:	**●11IG"TIME"00**	*Get current time*
T:	**●11A12:30:16**	*The Terminal returned current time*
PC:	**●11IS"TIME"0018:00:00**	*Try to set the time again*
T:	**●11A**	*Done!*

● *is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>*

First, you read about the Item "TIME". From the Appendix B you learn that this is an Array Item of type TIME and that it consists of a single value.

The first command (IG"TIME"00) means that you try to get a value (IG) of the Item, whose name is "TIME". Even though this Item consists of a single value, you *still need* to specify a member (00). This number must always consist of 2 digits, so add a leading 0 if necessary!

What you get in return is a string representation of this Item's current value.

Next, you try to change the Item's value. You send the **Item Set** command and specify Item name ("TIME"), subscript (00), and the new value ("18:00:00").

There is one extra command provided that allows you to fetch Item's default value. The sample session below fetches and sets the default value for the "BELLDUR" Item. This Item is of VAL type, contains 1 member, and its default value is 10:

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

PC:	**●11ID"BELLDUR**	*Fetch default value*
T:	**●11A10**	*Default value is 10*
PC:	**●11IS"BELLDUR"0010**	*Try to set this value*
T:	**●11A**	*OK*

● *is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>*

In the **Item Default** command, notice the absence of the closing quotation mark and member. Default values exist for the Item as a whole, not for each of this Item's member. Therefore, specifying exact member is not necessary. As for the quotation mark, it is not necessary in cases where no other data follows the name.

Some Items do not have a default name by nature. It is obvious, for example, that Items like TIME and DATE cannot have any meaningful defaults. Requesting default values for such Items returns "N" status codes (not applicable/ not available). Finally, you see the use for this obscure status code that we couldn't quite explain in 4.2.

All examples below have dealt with single-member Items. It so happens that our current TR40xx just doesn't have any real array Items. Nevertheless, the subscript is reserved, should never be omitted, and will definitely be used in the future. Just as an example, here is how you set the 99th member of an imaginary Item "ITEM1" (type=STR) to "ABC":

This is just an example. It won't work with the current TR40xx

PC:	**●11IS"ITEM1"99ABC**	*Set member 99 of ITEM1 to "ABC"*
T:	**●11A**	*OK!*

7.4. Learning about Items

The TR40xx provides a way to learn about all Items it supports. The **Item Info** command fetches a so-called info strings separately for every Item.

The info string consists of several TAB-delimited fields. Here is the string format::

#	Field	Comment
1	Name	Item's name. It's the same name you use in commands like **Item Set**, **Item Get**, etc.
2	ID	Unique ID that you can use to assign a help topic of your PC software's online help to a particular Item. ID's are guaranteed to be unique not only within a certain Terminal, but throughout entire TR4000 family. We will never use the same ID for different Items
3	Type	Item's type. This field is a string that can have the following values: "VAL", "STR", "TIME", "DATE", "TAB". This corresponds to value, string, time, date, and table types. The list of supported types may be expanded in the future
4	Size	Number of array members within this Item. It can be in the 1-99 range
5	Flags	This field actually consists of several flags that are described below
6	Definition	This is a string that can be used to display the Item's meaning. Your PC software may print it next to the Item's value for User's convenience
7	Help	Additional information on the Item's meaning. Your software may display it on request, when the User selects the Item or asks for help
8	Selection	This field is optional, it may only exist for Items of type VAL. It may consist of

	stings	one or several semicolon-delimited strings. Each string corresponds to one possible value of this Item, starting from 0. The purpose is to provide meaningful selection strings instead of bare values whenever necessary. Example below will illustrate the usage of selection strings

To better illustrate how info strings can be used in your software, let's get one string first, then see how it is utilized in our **TR4000 Control Center for Windows**. **Control Center** is a free open-source software that can be downloaded from out website (www.proxdata.com).

Example below uses the EXTBRATE (External Scanner Baud Rate) Item. This Item is of VAL type, it consists of a single member. Basically, this is a selector: every possible value corresponds to a certain standard baudrate: 0-1200bps, 1-2400bps, 2-4800bps, 3-9600bps, 4-19200bps. Since 0,1,2,3, and 4 are not just values, but "selections", it would be nice to display what they actually mean. This is where selection strings come handy.

Try the following example:

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

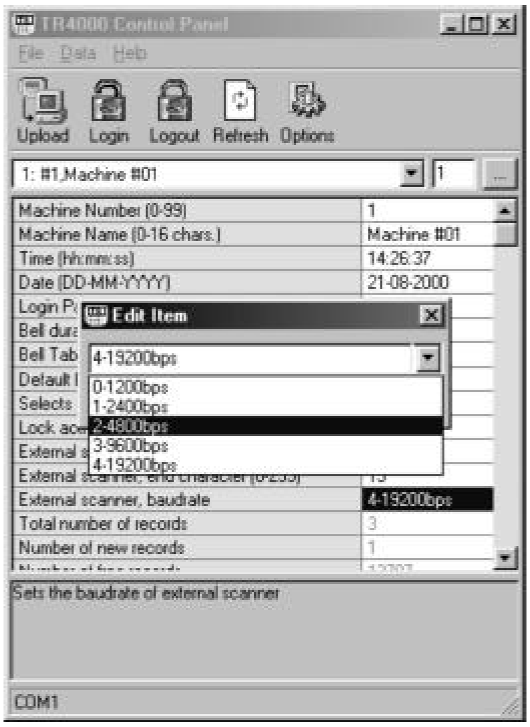
PC: T:	**●11I?"EXTBRATE** **●11AEXTBRATE → 10170 → VAL → 1 → 41 → External scanner baudrate → Sets the baudrate of external scanner → 0-1200bps; 1-2400bps; 2-4800bps; 3-9600bps; 4-19200bps**

● is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>
→ is the TAB character

The first field of Info String carries the name of this Item ("EXTBRATE"). It is followed by the unique ID (10170). Next is the Type field (VAL). Next field informs us that this Item only contains 1 member. Next is a Flags field (to be discussed later). Items definition is "External scanner baudrate". Help you can display on User's request is "Sets the baudrate of external scanner". Finally, there are several selections, each corresponding to a specific possible value of this Item.

Every Selection string is *guaranteed* to start with actual value, followed by hyphen, followed by comment, i.e. "1-enabled".

Sample screenshot below illustrates how TR4000 Control Center uses this Info String:



1: #1,Machine #01 | 1

Machine Number (0-99)	1
Machine Name (0-16 chars.)	Machine #01
Time (hh:mm:ss)	14:26:37
Date (DD-MM-YYYY)	21-08-2000
Login P	
Bell dur	
Bell Tab	
Default I	
Selects	
Lock ac	
External s	
External scanner, end character (0-255)	13
External scanner, baudrate	4-19200bps
Total number of records	3
Number of new records	1

4-19200bps
0-1200bps
1-2400bps
2-4800bps
3-9600bps
4-19200bps

Sets the baudrate of external scanner

COM1

All Items are displayed in a data grid, each Item's Definition field is at the left column, and the value is in the right column. EXTBRATE Item is selected for editing, so extended info about it (from the Help field) is shown under the grid. Current Item's value is displayed using a corresponding selection string. In the Edit dialog box, the drop-down displays all available selections.

7.5. Flags field

Flags field consists of several 1-bit flags:

Bit0	R	Read- Item's value can be retrieved at any time, regardless of login status
Bit1	RP	Read Protected- Item's value can only be retrieved when logged in
Bit2	W	Write- Item's value can be set at any time, regardless of login status
Bit3	WP	Write Protected- Item's value can only be set when logged in
Bit4	---	*<This field is currently unused>*
Bit5	FU	Update required after changing this Item's value

Note: when both R and RP, or W and WP are set, then R and W take precedence

R, RP, W, and WP flags define the access rights for this Item. Update flag tells you whether you need to use the **Update** command after having changed the Item's value (See Section 7.2.).

To determine which fags are set for a particular Item, convert the value of a Flags field to its Binary representation, then check individual bits. For example, the value of the Flags field for Item EXTBRATE is 41 (decimal). In binary form, it looks like 101001. The following flags are set: Bit0 (the rightmost one)- R, bit 3- WP, bit 5- FU. This means that this Item's value can be retrieved at any time, that you need to be logged in if you want to change this Item's value, and that you need to use the **Update** command to force the TR40xx use the newly set value. It is *that* simple! :)

7.6. Using Item numbers instead of names

It may be convenient to use names like EXTBRATE or TIME when setting and getting Items manually, but you need something different if you want to create an auto-learning piece of code. You auto-learning program cannot know what set of Items a given Terminal supports and what their names are, right?

For this reason, all of the TR40xx's Items may be referenced by number. All commands that accept Item names as parameter will also accept bare numbers. Item number should always be formatted to consist of 2 digits, so add leading zero when necessary!

There is an **Item Lookup** command that helps you find the Item's number by its name. Example below illustrates this:

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

PC:	`●11ILBELLDUR`	*Lookup the number for "BELLDUR". Notice that there are no opening and closing quotation marks*
T:	`●11A5`	*The number is 5*
PC:	`●11IG0500`	*Read this Item's value. Notice that "05", not "5" must be input*
T:	`●11A10`	*The value is 10*

● *is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>*

Handling Items by their numbers is very convenient when you are writing a piece of code that displays all the Items in a list, like our **TR4000 Control Center** does (see screenshot in Section 7.5.). Our **Control Center** doesn't know (or care about) Item names or what these Items actually mean. All Items are handled formally, using the Info strings. The **Item Count** command is used to find out the total number of Items (N), then Item Info command is used to fetch the info strings for every Item from 0 to N-1.

7.7. A word of caution on using Item numbers

This is so important that we have even put it into a separate Section!

Unlike Item names, Item numbers can change between Terminal versions. We may, for example, insert a new Item in front of the one that used to go by the number “5”, thus turning it into “6”. It won’t bother you when you are designing a “smart” software that gets all Items along with their info strings every time it connects to a certain Terminal. However, if you are writing a traditional fixed software, then you better avoid using Item numbers. Do use Item names instead, they are guaranteed to stay the same!

7.8. Tables

Tables are complex Items. Each Table Item contains entire datatable within it. Just like with “simple” Items, you can learn about tables and their structure.

Each datatable consists of fields. Fields only differ in their type. All other properties (access level, update) apply to the datatable Items in general, not to individual fields.

Field Info command fetches the info string for a given field of a given table. The Info string includes:

#	Field	Comment
1	ID	Unique ID that you can use to assign a help topic of your PC software’s online help to a particular field. ID’s are guaranteed to be unique not only within a certain Terminal, but throughout entire TR4000 family. We will never use the same ID for different Items
2	Type	Item’s type. This field is a string that can have the following values: “VAL”, “STR”, “TIME”, “DATE”. This corresponds to value, string, time, and date types. The list of supported types may be expanded in the future
3	Definition	Field name that you can display at the caption of the datatable
4	Help	Additional information on this field. Your software may display it on request, when the User selects the field or asks for help
5	Selection stings	This Info String’s field is optional, it may only exist for data table fields of type VAL. It may consist of one or several semicolon-delimited strings. Each string corresponds to one possible value of this field, starting from 0. The purpose is to provide meaningful selection strings instead of bare values whenever necessary. Example below will illustrate the usage of selection strings

How to find out the number of fields in a given table? The number is written in this Item’s info string, in the size field. For all Items other then Tables, this field carries total number of array members for the Item. For table Items, however, it records the total number of fields in the table. Example below illustrates the process of learning about the TABDEFEV (Default Event Table):

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

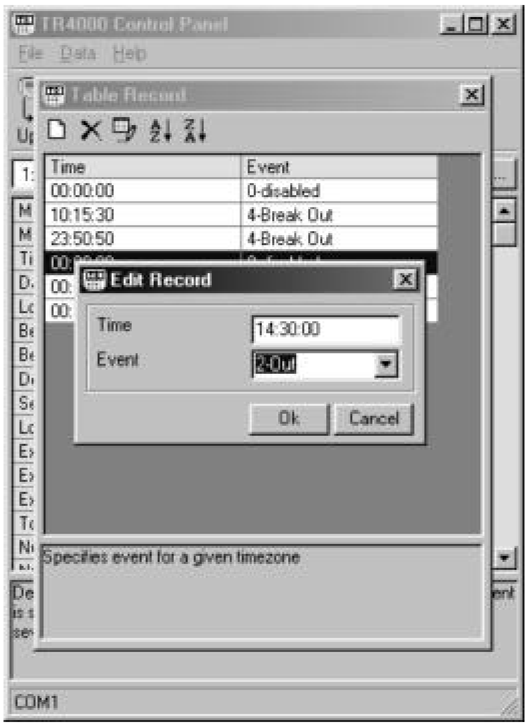
Some Terminal replies in this example have been abbreviated for clarity

PC:	**●11ILTABDEFEV**	*Lookup the number for “TABDEFEV”*
T:	**●11A7**	*The number is 7*
PC:	**●11I?07**	*Get info string*
T:	**●11ATABDEFEV→10070→TAB→2→41→…**	*This table has 2 fields* ***(reply abbrev.)***
	●11I*0700	*Fetch info string for the first field*
	●A20010→TIME→Time→Specifies…	***(Abbreviated)***
	●11I*0701	*Fetch info string for the second field*
	●A20020→VAL→Event→Specifies…	***(Abbreviated)***

● is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>

→ is the TAB character

First, we lookup TABDEFEV Item’s number. Then, we get this Item’s info string and learn that it is a table, and that the table has two fields. Finally, we use the field Info command twice to get the field info separately for each field. As it turns out, the first field is of TIME type, the second one is VAL. Selection strings are available for the second field. The screenshot below is taken at the moment when the TR4000 Control Center is in the table editing mode.


TR4000 Control Panel

File Data Help

Table Record

Time	Event
00:00:00	0-disabled
10:15:30	4-Break Out
23:50:50	4-Break Out
00:	
00:	
00:	

Edit Record

Time: 14:30:00

Event: 2-Out

Ok Cancel

Specifies event for a given timezone

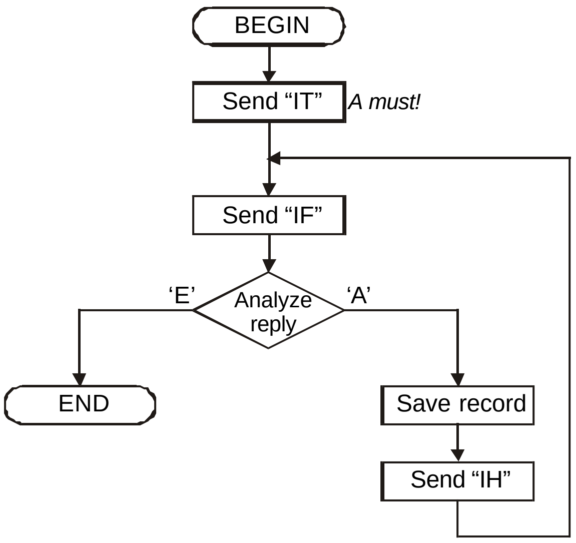
COM1

Once again, every Selection string is *guaranteed* to start with actual value, followed by hyphen, followed by comment, i.e. “1-enabled”.

7.9. Uploading table data

Data table contents uploading process is quite similar to the database uploading process described in 6.3. Two commands, **Item Fetch** and **Item Confirm** are used to read out all records in the specified datatable. Records are fetched in a loop cycle, until the “E” reply code is returned by the Terminal. An **Item Top** command is used to reset the datatable pointer to the top of the datatable. Always reset the pointer prior to rereading the datatable.

All datatable records are returned with TAB-delimited fields. A special record number field is added automatically as the header of each record. You don’t need to display this field on the screen, but you do need to store it along with record’s data (in order to be able to reference a specific record to edit or remove). Records are *always* fetched in the *ascending order* of their respective record numbers, but record numbers *may not be consecutive* (i.e. you may get numbers 0,1, than 3). Record numbers may be in the 0-9999 range. The diagram below illustrates the datatable reading process.



This record-by-record upload method is bulletproof. It makes sure that no records ever get "lost" on the way from the TR40xx to the PC. If your PC fails to receive some record, it may send the **Item Fetch** command again to retrieve the same record. And the **Item Confirm** command may only advance the pointer *no further* than the next unread record, no matter how many times you repeat this command between two subsequent **Item Fetch** commands.

Sample session below illustrates the datatable upload process:

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

PC:	**●11IT07**	*Reset read pointer to the top of the datatable*
T:	**●11A**	*OK*
PC:	**●11IF07**	*Fetch a record*
T:	**●11A0→10:00:00→1**	*Record 0 fetched*
PC:	**●11IH07**	*Confirm record 0*
T:	**●11A**	*Acknowledged*
PC:	**●11IF07**	*Fetch next record*
T:	**●11A2→09:30:00→0**	*Record 2 fetched (notice that record numbers are not consequential)*
PC:	**●11IH07**	*Confirm record 2*
T:	**●11A**	*Acknowledged*
PC:	**●11IF07**	*Fetch another one*
T:	**●11E**	*No more records*

● is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>
→ is the TAB character

7.10. Adding, editing and deleting records

New record may be added using the **Item Add** command. Record number needn't be supplied- the TR40xx will find and utilize the *lowest unused* record number automatically. Thus, if the datatable contained records 0, 1, and 4, then the newly added record will have the number 2 (then 3, then 5). "E" reply status code is returned when the datatable runs out of free record space.

Editing and removing records requires a record number to be supplied by the PC. Two commands used for this purpose are **Item Edit** and **Item Remove**. Attempting to edit a non-existent record will generate the "I" reply status code, while trying to remove a non-existent record will return "A" anyway.

Here is the sample session in which we add, edit, and remove a record:

Sample sessions are shown with packet wraps (i.e. you may type commands below right into the HyperTerminal). They assume that you are working with Terminal #1 (Destination=1)

PC:	**●11IA0712:00:00→1**	*Add a record (notice that there is no record number)*
T:	**●11A2**	*OK, new record number is 2*
PC:	**●11IE072→13:00:00→2**	*Edit record 2*
T:	**●11A**	*OK*
PC:	**●11IR072**	*Remove record 2*
T:	**●11A**	*OK*

● is STX (02H, <CTRL>-<B> under HyperTerminal), all packets end with <CR>
→ is the TAB character

Section 8. miscellaneous commands

This last group includes commands that are hard to classify:

All commands and replies are shown without packet wrap (STX, destination, source, etc.)

Cmd	Usage	Reply	Comment
E	E ***xx…x***	A ***xx…x***	**Echo**. Returns whatever data was sent
P	P	A ***yy…y*** , ***zz…z*** | F	**Poll**. Returns Terminal's machine number and name

xx…x	Alphanumeric data of any kind
yy…y	Machine number (numeric)
zz…z	Machine name (alphanumeric)

8.1. The Echo command

This command may be used for testing purposes- just to make sure that the Terminal is there (we have, in fact, started the protocol description from using this command as an example- see 2.3).

8.2. Polling the Terminal

The **Poll** command provides you with a convenient method of determining if the Terminal is there, while at the same time fetching this Terminal's number and name. Name and number data comes from MACHNAME and MACHNO Items. You may use this command to display the list of available Terminals without having to go into lengthy self-learning process for each Terminal first. The **Poll** command (and, of course MACHNAME and MACHNO Items) is guaranteed to be supported by all future versions of the TR4000 series Terminals.

Appendix A. List of supported Items (TimeRecorder V3.10)

Note: this list is only relevant to the **TimeRecorder V3.10** firmware. Future firmware releases may contain additional Items as well as introduce some changes to the existing ones. We guarantee to keep Item names and IDs intact (that is, for the Items that go into the next firmware version unchanged). However, Item numbers may change in the future.

#0	**MACHNO**	ID=10000 Type=VAL Size=1 Flags=9 (R, WP)

Machine Number (0-99). Specifies the serial number for this Machine. Serial number is added to every Time and Attendance record and is useful for distinguishing the data generated by different Machines.

#1	**MACHNAME**	ID=10010 Type=STR Size=1 Flags=9 (R, WP)

Machine Name (0-16 characters). Specifies Machine name. Names can be used to conveniently distinguish different Machines from each other. Names do not go into the Time and Attendance Database.

#2	**TIME**	ID=10020 Type=TIME Size=1 Flags=9 (R, WP)

Time (hh:mm:ss). Sets Machine's time.

#3	**DATE**	ID=10030 Type=DATE Size=1 Flags=9 (R, WP)

Date (DD-MM-YYYY). Set's Machine's date.

#4	**LOGINPWD**	ID=10040 Type=STR Size=1 Flags=26 (RP, WP)

Login Password (0-8 characters). Login password is used to restrict online access to the Machine.

#5	**BELLDUR**	ID=10050 Type=VAL Size=1 Flags=41 (R, WP, U)

Bell duration (0-99 sec., 0-disabled). Sets the amount of time (in seconds) the bell will be on (once activated).

#6	**TABBELL**	ID=10060 Type=TAB No of fields=1 Flags=41 (R, WP, U)

Bell Table. Defines a daily schedule for the bells. The table consists of a single field- "Time". Up to 32 different daily times for the bell can be set.

#7	**TABDEFEV**	ID=10070 Type=TAB No of fields=2 Flags=41 (R, WP, U)

Default Event Table. Defines default event schedule. Within a given timezone, default event is selected automatically (if enabled) after an ID-card is read or after several seconds of inactivity. The table consists of two fields: Time field that defines the starting time of a timezone, and Event field, that defines an event that will be default for this timezone. Possible event values are: 0- disabled (no automatic default event pre-selection for a given timezone), 1- IN, 2- OUT, 3- BREAK IN, 4-BREAK OUT. Values 1 thru 4 correspond to the functions of keys F1…F4 on the TR40xx's keypad.

#8	**DEFSHIFT**	ID=10080 Type=VAL Size=1 Flags=9 (R, WP)

Selects default shift (1-99, 0-disabled). When default shift is set (not 0), the shift will return to the default value after ID-card is read or after several seconds of inactivity.

#9	**LOCKDUR**	ID=10090 Type=VAL Size=1 Flags=9 (R, WP)

Lock activation duration (0-99 sec., 0-disabled). Defines the amount of time (in seconds) the door lock relay will be kept activated after every ID-card read.

#10	**EXTSCHAR**	ID=10150 Type=VAL Size=1 Flags=41 (R, WP, U)

External scanner, start character (0-255). ASCII code of the start character of the external scanner data packet.

#11	**EXTECHAR**	ID=10160	Type=VAL	Size=1	Flags=41 (R, WP, U)

External scanner, start character (0-255). ASCII code of the start character of the external scanner data packet.

#12	**EXTBRATE**	ID=10170	Type=VAL	Size=1	Flags=41 (R, WP, U)

External scanner, baudrate. Sets the baudrate of external scanner port: 0-1200bps, 1-2400bps, 2-4800bps, 3-9600bps, 4-19200bps.

#13	**NRTOTAL**	ID=10100	Type=VAL	Size=1	Flags=1 (R)

Total number of records. Displays the total number of records currently found in the Time and Attendance Database.

#14	**NRNEW**	ID=10110	Type=VAL	Size=1	Flags=1 (R)

Number of new records. Displays the number of new Database records (i.e. the number of records that were added since the Database was last uploaded or saved to disk).

#15	**NRFREE**	ID=10120	Type=VAL	Size=1	Flags=1 (R)

Number of free records. Displays the number of free (unused) records in the Time and Attendance Database.

#16	**POWER**	ID=10130	Type=VAL	Size=1	Flags=1 (R)

Power status. Displays current power status: 0-Power OK, 1-Backup (battery) power, level OK, 2-Backup (battery) power, battery LOW.

#17	**VERSION**	ID=10140	Type=VAL	Size=1	Flags=1 (R)

Firmware version. Displays the version of a currently loaded firmware.

Appendix B. Database record structure

Note: the information below is only relevant to the **TimeRecorder V3.10** firmware. Future firmware releases may expand or alter the format of the Time and Attendance Database (although we will try to keep backward compatibility if possible).

The database record consists of several TAB-delimited fields:

Field	Length	Comment
Event	1	Event can be 1 (in), 2 (out), 3 (break in), and 4 (break out). This corresponds to the F1…F4 keys on the TR40xx's keypad
Date	10	DD-MM-YYYY
Time	8	HH:MM:SS
ID-code	Variable, max. 41	Apostrophe is added in front of the ID-code to make sure that this field is treated as a string by programs like MS Excel.
Shift	2	Can be in the range from 00 to 99
Machine No	2	Can be in the range from 00 to 99

The TR40xx performs a validity check on every record field prior to sending the record to the PC (or saving it to disk). Should some field's data turn out to be invalid, it will be automatically substituted for a default one. Invalid data may happen due to some internal database malfunction. Also, the database recovery (using **Recover The Database** command, see 6.6) may yield a lot of bad records simply because it covers entire database memory space (and this includes bad, erased and un-initialized areas).

Here are the substitution rules for invalid fields:

Field	Substitution data in case this field's actual data is invalid
Event	"0"
Date	"01-01-1999"
Time	"00:00:00"
ID-code	If ID-code length is out of range (>40 or 0), then "Invalid_ID" will be returned in this field. If the length is OK but some character codes fall outside of the 20H…7FH range, then these characters will be substituted for " "
Shift	"00"
Machine No	"00"