

Interface Emit EPT ActiveX Control 3.4.14+

Properties

Unit

Determines what kind of unit the control should interface against. Possible values are 1 – RTR2 TimeRecorder (Program 12 and 13), 2 – EPT Reading unit connected directly to Com Port on PC, 3 - RTR2 TimeRecorder (Program 1 - 11) .4 MTR 2 and 3, Default is 1 – RTR2.

Modus

Determines the amount of data from the badge which should be sent to the host application. Possible values are 1- Complete, the complete contents of the badge is decoded and sent to the host application, 2 – Only badgenumber is sent (used for intermediate timing, last control etc) This option is only valid if the value of unit is 2. Default is 1 – Complete.

LogFile

Filename for logfile. All read badges are also logged to a text file before it is sent to the host application. The format of the data written to the file is exactly the same as the format of the data sent to host applications. Default filename is <current directory>+EPREAD+<com port number>+.LOG.

Rtr2Start

The date and time that The RTR2 Time Recorder is 0. Used when the value of unit is 1. Default value is current time.

ComPrt

Determines which Com Port on the PC that the RTR2 or the Reading Unit is connected to. Possible values are 1,2,3,4,5,6..... Default value is 1.

iComStarted

Setting this item to 1 activates communication in accordances with current parameter settings. It is an alternative to calling method StartComm in programming environments having problems calling methods. Setting this item to 0 stops communication (alternative to method StopComm).

iSpool

Only valid for MTR. Property for spooling data from MTR. Value –1 spools all data from MTR. Value > 0 means spooling from value. Using property *iSpool* is equivalent to using method *Spool*.

iGetMessage.

Only valid for MTR. If value >0 message with number equal to value is retrieved from MTR. Using this property is equivalent to calling method *GetMessage*.

iSetClock

Only valid for MTR. If set to 1 before calling *StartComm* (*iComStarted*=1) MTR clock is set to value from PC clock. Equivalent to calling method *SetClock*.

iClearMem

Only valid for MTR. If set to 1 history and counters are cleared in MTR. Equivalent to method *ClearMem*.

iGetStatus

Only valid for MTR. If set to 1 MTR returns Statusmessage. Equivalent to method *GetStatusFromMTR*

iNewSession

Only valid for MTR. If set to 1 MTR is shifting session/race. Equivalent to method *MTRNewSession*

Methods

StartComm

Activates communication with selected unit in accordance with current parameter settings.

StopComm

Stops communication.

Spool(messagenumber)

Spool from MTR. From and including messagenumber. -1 as messagenumber means spooling all data from MTR.

GetMessage(messagenumber)

Get message with number *messagenumber* from MTR.

SetClock

Sets MTR clock equal to PC clock. This method only works if called before StartComm.

ClearMem

Clears all history and all counters in MTR. This method only if called before StartComm and directly after MTR is started (battery inserted).

NB. Messages retrieved from MTR via *Spool/GetMessage* (or equivalent properties) are logged to logfile as they were online messages.

GetStatusFromMTR

Get a status message from MTR

MTRNewSession

Tells MTR to shift session/Race.

Events

GetNextBadgeNo

This event returns the last read badgenumber as an integer. Only active if Unit=2 and Modus=2.

GetNextBadge

This event returns the last read badge data as a string. The format of the string is CSV and it contains the following fields. Only active if Modus=2.

Example (Modus=2,Unit=2) :

```
"X","0","0","016452","06.01.98 00:09:32.000","06.01.98 00:09:32.000" ,  
16452,0016,0096,000,00000,040,03627,033,03630,042,03632,077,03633,093,03634,250,0  
3638,250,00000,040,00000,000,00000,000,00000,000,00000,000,00000,000,00000,000,0  
0000,000,00000,000,00000,000,00000,000,00000,000,00000,000,00000,000,00000,000,0  
0000,000,00000,000,00000,000,00000,000,00000,000,00000,000,00000,000,00000,000,0
```

NOTE: THIS SPECIFICATION IS SUBJECT TO CHANGE!!

All information herein is without any WARRANTY to what actual product will/does deliver.
No claims are accepted with reference to any statements made in this document.

Baudrate/parity: 9600-8-N-1

Non-pollled data stream

The MTR sends "MTR-datamessages" directly (without any inquiry from computer) whenever new data is available (=a card is read by the control).

History-file

ECard-data is stored in a "history-file" that works like a "ring-buffer" holding the latest ECard data. The capacity of this ringbuffer depends on the average number of controls each ECard contains. This history file only saves the following information:

Timestamp ("MTR-time" when Ecard was read by MTR)
Card-id
Code&Time-pairs (only non-0 pairs are actually stored in MTR)

When ECard messages are retrieved from history (opposed to when transmitted in real time) the fields Producweek,Produceyear,ECardHeadSum,ASCII_string will be 0 og "Space" (0x20 Hex) and should be ignored.

History-capacity.

The number of ECard that a MTR can hold depends on number of controls stored in each Ecard. The following guide gives an indication of how many ECard a MTR can hold.

8 controls->3200 ECards
10 controls->2700 ECards
15 controls->2000 ECards
20 controls->1600 ECards
50 controls->700 ECards

Package numbering

Every package is sequentially numbered from 1 and up since MTR is "cold-booted". The only way to "cold-boot" a MTR is to remove internal coin-cell battery for a few minutes.

Power-cycles.

Every time the MTR is started (a battery is inserted) and a ECard has been read, it starts a new "session" This means that the number of this first message received is stored and transmitted in the CurrentSessionStart# field of the status message. This number is moved to the Prev1SessStart# field at the next power cycle, and so on until it is "lost" after 7 power-cycles. Some or all data for a session may be still be lost even if the "session start number" is stored by a large amount of ECard data.

Networked MTR's

Note that future version of MTR's may be networked. The "MTR-id" field (=serial number) will at this time become more important as some messages received from a single MTR also may contain messages from the other MTR's that are networked to the "master-mtr" connected to the computer. This information is only here to make software implementors from implementing logic like:

"If first package number is X then 10 packages later we will recive package numer X+10."

COMMAND DESCRIPTION

The MTR will accept the following commands

/ST - Status

Will make the MTR to send a Status-message

/SA - Spool all data in MTR2. No Polling will be done!

/SBxxxx - Spool Binary. Spool all data from package# xxxx (LSB) and to on

/NS - New session

/GBxxxx - Get message binary.

Will send a single data-message from history. The MTR will continue "polling" for ECards during data sending, with short delay for receipt. Least significant byte first.

/SCymdhms - Set Clock

The 6 bytes are binary values for current time

y - year; values accepted are 90 to 99 (1990..1999) and 0 to 53 (2000..2053)

m - month; values accepted are 1 to 12

d - daynumber; values accepted are 1 to 31

h - hour; values accepted are 0 to 23

m - minute; values accepted are 0 59

s - second; values accepted are 0 59

/CL - Clear Ringbuffer. Will clear all history (and reset package counters!)

MESSAGE DESCRIPTION:

=====

MTR--datamessage

Fieldname	# bytes	
Preamble	4	FFFFFFFF(hex) (4 "FF"'s never occur "inside" a message). (Can be used to "resynchronize" logic if a connection is broken)
Package-size	1	number of bytes excluding preamble (=230)
Package-type	1	'M' as "MTR-datamessage".
MTR-id	2	Serial number of MTR2; Least significant byte first
Timestamp	6	Binary Year, Month, Day, Hour, Minute, Second
TS-milliseconds	2	Milliseconds NOT YET USED, WILL BE 0 IN THIS VERSION
Package#	4	Binary Counter, from 1 and up; Least sign byte first
Card-id	3	Binary, Least sign byte first
Producweek	1	0-53 ; 0 when package is retrived from "history"
Producyear	1	94-99,0-..X ; 0 when package is retrived from "history"
ECardHeadSum	1	Headchecksum from card; 0 when package is retrived from "history"

The following fields are repeated 50 times:

CodeN	1	ControlCode ; unused positions have 0
TimeN	2	Time binary seconds. Least sign. first, Most sign. last; unused:0
ASCII-string	56	Various info depending on ECard-type; 20h when retr. from "history" (See ASCII-string)
Checksum	1	Binary SUM (MOD 256) of all bytes including Preamble
NULL-Filler	1	Binary 0 (to avoid potential 5 FF's. Making it easier to haunt PREAMBLE

Size 234

Status-message

Fieldname	# bytes	
Preamble	4	FFFFFFFFHex (FFFFFFFF never occurs elsewhere within a frame).
Package-size	1	number of bytes excluding preamble (=55dec; 37Hex)
Package-type	1	'S' as "Status-message" (53Hex).
MTR-id	2	Serial number of MTR2.
CurrentTime	6	Binary Year, Month, Day, Hour, Minute, Second
CurrentMilliseconds	2	Milliseconds NOT YET USED, WILL BE 0 IN THIS VERSION
BatteryStatus	1	1 if battery low. 0 if battery OK.
RecentPackage#	4	if this is 0, then ALL following # should be ignored!
OldestPackage#	4	note: If RecentPack==0 then this is still 1!... meaning...: ...Number of packages in MTR is "RecentPackage#-OldestPackage#+1"
CurrentSessionStart#	4	Current session is from here to RecentPackage# (if NOT = 0)
Prev1SessStart#	4	Prev session was from Prev1SessStart# to CurrentSessionStart#-1
Prev2SessStart#	4	
Prev3SessStart#	4	
Prev4SessStart#	4	
Prev5SessStart#	4	
Prev6SessStart#	4	
Prev7SessStart#	4	
Checksum	1	Binary SUM (MOD 256) of all bytes including Preamble
NULL-Filler	1	Binary 0 (to avoid potential 5 FF's. Making it easier to haunt PREAMBLE

Size 59

ASCII-string

The 56 ASCII bytes sent from ECard will have the following info (only in on-line mode! Offline all blank!)

NOTE This ASCII string is reprogrammable, so no assumption should be made that the following data will remain correct for future versions of ECards.

New ECards (manufactured after summer 1998 with green/amber casing):

"EMIT EPT SYS VER 2 DISP-1 S0059P0136L0004 "

The S-field (pos 41-45) indicates the number of disturbances/noise that woke up the ECard but was not recognized the signal)

The P-field (pos 46-50) indicates the number of "tests/readings". A Test is when ECard is made put to sleep by MTR or 250-control within approx 4 minutes from being waken-up.

The L-field (pos 51-55) indicates the number of events when ECard was awake for more than approx 4 min.

Old ECards (yellow)

"REGNLY TRACK RECORDING SYSTEM DISP-1 DISP-2 DISP-3 "

Emit as

Protocol description

Type: Regnly EPT system

Data from 250 reader.

Communication settings: RS323, 9600, No parity, 8 bit, 2 stop bit.

Byte	Description	Bytes
1	Package start Identification FFH	1
2	do FFH	1
3	E-cards no LSB Binary code Max 999999 DEC	1
4	do	1
5	do	
6	not used	
7	Production week Binary 1-53	1
8	Production year Binary 94-xx	1
9	not used	
10	check byte e-card no. Addition of bytes 3-10= Bin 0 (Mod 256)	1
11-160	Control kodes and times. 50 x 1 byte binary control code 0-250 50 x 2 bytes bianry time 0-65534 sec.	150
161-168	Ascii sting Emit time system/ Runners name	8
169-174	do	8
177-184	do	8
185-192	do	8
193-200	Ascii string Disp 1	8
201-208	Ascii string Disp 2	8
209-216	Ascii string Disp 3	8
217	check byte. Addition of all bytes 1-217 = bin 0 (Mod 256)	1

	Sum	217

All info must be xor with OD before seperated

Disp 2-3 is now used for counters:

Disp 2:S0000P00	S0000 -> Numbers of disturbance
Disp3:00L00000	P0000 -> Numbers of tests
	L0000 -> Numbers of races

Emit as 11.2.94