**Thermitrack programming guide** Check www.thermitrack.com for latest info
Update history :
V2.11 18/2/09 Initial Release version
V2.15 25/2/09 added W option to R command, made Y after Z command not case-sensitive, added F option to V command, changed default netid to 1.
Firmware V2.20 6 Mar 09 added more baudrates

## 1 Terminology

Host : The thing that the camera is connected to (PC, control system, Arduino etc.). Note that if in autonomous mode, the host never actually needs to send anything to the camera, but it's still called the host nonetheless.
Packet : Sequence of characters from camera containing a specific type of data. Packet type is identified by the first character and most packets are terminated by a <CR> character.
ASCII mode : data comprising only of printable ASCII characters
Binary mode : data comprising any byte value 0 to 255
Frame Interval : the number of frames between autonomous packet transmissions (1= every frame, 2 = every 2nd frame etc.). A frame is approximately 32 milliseconds, but this can increase to up to around 60ms when several targets are being tracked ( even if tracking packets are not enabled)
<CR> : ASCII carriage-return character, 13 decimal, 0x0d hex.
[n] : optional paremeter
0xnn : Hexadecimal value, e.g. 0x12 = 18 decimal

## 2 Introduction

Thermitrack captures realtime 16x16 pixel thermal images of targets in its field of view which are at a different temperature to the background (typically the body heat of people). From these images, it uses shape-fitting algorithms to extract high-resolution co-ordinates of those targets. The internal framerate is nominally 30 frames per second, although this can reduce when tracking multiple targets.

The thermitrack camera interfaces via an RS422 serial interface. Adapters are available to connect to RS232 and USB, the latter also providing power to the camera. Default baudrate is 115.2kbaud, 8N1.

The camera can be configured to output various packet types containing status, tracking and image data in various formats, at specified intervals. Each packet type has its own timer, so intervals can be different for each packet types.

Depending on application requirements, Thermitrack can be configured to output data autonomously at regular intervals, or only in response to requests from the host. A mixture of these two modes is also possible, e.g. outputting tracking data autonomously and image data only when requested. When configured for autonomous mode, it can be regarded as a simple output-only device which requires no action from the host.
Thermitrack can be easily configured using any standard terminal software (e.g.Hyperterminal) using simple text commands. The configuration is held in nonvolatile memory and will take effect whenever power is applied.
The factory default configuration is to autonomously send status packets only, once every two seconds.

When configuring using terminal software, if the camera has been previously configured to send autonomous packets, these will appear on the screen making it difficult to see what's going on. Typing **T<enter>** will switch off all autonomous packets (without affecting the nonvolatile configuration).

## 3 Operating modes

### 3.1 Startup mode

For the first 14 seconds after powerup, the camera is in startup mode.
For the first 2 seconds of startup, baudrate is always 115.2Kbaud, to allow for reconfiguration if a different baudrate has been selected. If a '**B<CR>**' command is received during the first 2 seconds of startup, the baudrate will be locked to 115K2 until the next restart or power-cycle, regardless of the nonvolatile baudrate setting ( which will be unchanged).
During startup, all autonomous packets except status are disabled. Status packets will be output if enabled, but only after the initial 2 second period. This prevents packets being sent at 115K2 baud which would cause errors if running at other baudrates.

Startup mode will only last for 2 seconds on a soft restart (R command) ( unless restarted within 14 seconds of power-up, in which case the remaining part of the 14 second period will elapse)

### 3.2 Warmup mode

After the 14-second startup delay, the camera enters warmup mode, during which time the camera sensor is stabilising its internal temperature, and the 2 red LEDs flash alternately. **Warmup mode takes typically 2 minutes.**
No real image or tracking data is available during this time but autonomous packets are enabled.
Image packets will contain an hourglass icon image ( this can be disabled via the ImgFlags parameter), tracking packets will show no targets.

Warmup mode will not re-occur on a soft restart (unless the restart occurred during warmup, in which case any remaining part of the warmup period will continue as before) . A soft restart can force a new warmup if required – this is typically useful to test that the host system deals correctly with warmup states without having to disconnect power.

### 3.3 Active mode
All packet types are available.

### 3.4 Fault mode
This occurs if no data is received from the sensor for 15 seconds. This should never be seen in normal use – one possible cause is the power supply voltage being too low.

## 4 Commands (Host to camera)
Commands comprise a single ASCII letter, optional parameters, and a terminating carriage return (<CR>) character. Comamnds are not echoed – when using terminal software, you may want to turn on local echo so you can see what is being typed. Terminal software should also be configured to add a linefeed after a received carriage return. Commands are not case-sensitive.

Commands have a 3 second timeout – if more than 3 seconds elapse between characters, or between the command and terminating <CR> the command buffer will be cleared and a "=TO" response sent.. This ensures correct framing if garbage characters are received. **This should be remembered when configuring using terminal software – slow typists beware!**

Note that if any packet types are set to automously transmit, it is possible that a packet will have started transmission at the same time that a command is sent, so you may receive an autonomous packet before the reply to the command, however packets will always be complete, i.e. one packet type cannot interrupt another..

Coammands are fully queued, so it is possible to issue a second command without waiting for the response to the first – commands will be executed in sequence, but remember that autonomous packets may appear inbetween the responses to multiple commands..
For example to set two parameters, you can send **`P0=1<CR>P1=1<CR>`**, which will return **`=OK<CR>=OK<CR>`**

Responses to commands are prefixed by the '=' character, except the G and W commands, which just return the requested packet types. The '=' allows command responses to be distringuished from data packets, as autonomous data packets may arrive between sending a command and getting the response.

**Table 4.1 : Commands from host to camera : configuration commands**

| Command | Format & examples | Reply | Description |
|---|---|---|---|
| **V**<br>Version | **V[F]<CR>**<br><br>e.g.<br>**V<CR>**<br>returns all info<br><br>**VF<CR>**<br>returns firmware version only as =x.yy<CR><br>e.g. =2.15<CR> | | Shows firmware version and current configuration in user-readable form.<br>For compatibility with other commands and multi-camera modes, each line is prefixed with "=" and terminated by <CR><br>Adding F displays only the firmware version in fixed x.yy<cr> form to allow host to identify version, e.g. to check compatibility.<br><br>Sample output below.<br>First line is firmware version info<br>Second line is label programmed via L command<br>Third line is packet type/frame intervals for each packet type. The figure before the '/' is the current setting (set by T command or from default setting at powerup), the second figure is the default setting which will apply after the next powerup or restart.<br>Fourth line shows paramater settings<br>Fifth line shows baudrate, supply voltage and buffer usage. See section on networking for explanation of buffer figures.<br>Supply voltage allows checking of voltage drop over long cables.If this drops below 7.5, or 2V lower than the supply from the host end, then the cabling is inadequate. |
| | =Thermitrack V2.16 (C) 2007-9 White Wing Logic www.thermitrack.com<br>=Label : four<br>=Pkts (Now/Default): S10/0 O0/0 U0/0 M1/0 N0/0 R0/0 I1/0 A0/0<br>= P0 (TrkMode ) =0, P1 (ImgFilt ) =0, P2 (ImgFlags) =0, P3 (ImgMin ) =112, P4 (ImgMax ) =144, P5 (NetID ) =4,<br>=Baud : 11 (500000) 8N1 Vin=10.010 V. TX Buffer= 628/1024, RX Buffer= 322/1024 bytes | | |
| **D**<br>**Default packets** | **D[<pt>,<fr>[<pt>,<fr>]]<CR>**<br><br>e.g.<br>**DI,1<CR>**<br>outputs binary image every frame<br><br>**D<CR>**<br>Stops all autonomous packets<br><br>**DM,1,I,10<CR>**<br>Outputs M packets every frame and I packets every 10 frames | =OK<CR><br>or<br>=ER<CR> | Set default packet type(s) to be sent autonomously. This sets the power-on values, as per T command.<br><br>This value stored in nonvolatile memory and does not take effect until power-cycle or restart command. This is so that units can be configured easily without packet data obscuring the screen |

<table>
<tr>
<td>B<br><u>B</u>audrate</td>
<td>B<n>[S}<CR><br>e.g.<br>B5<CR><br>Set 230400 baud on next restart.<br><br>B<CR><br>Lock baudrate to 115K2 (only accepted during first 2 secs of startup). This will always return a =ER reply<br><br>B8S<CR><br>set baudrate to 250kbaud with 2 stop bits</td>
<td>=OK<CR><br>or<br>=ER<CR></td>
<td>Set baudrate. See table below for rates.This value stored in nonvolatile memory, but the new baudrate will not take effect until restart or next powerup<br><br>Rates above 115K2 are only guaranteed for RS422 & USB hardware interfaces.<br>At rates lower than 115K2 there is a risk of buffer overflow, especially in image modes, unless frame rate is low enough that the (1 Kbyte) transmit buffer will not overflow.<br><br>During the first 2 seconds of startup mode, baudrate will always be 115K2, 1 stopbit, to allow recovery of mis-configured units.<br>If a B<CR> or B<n><CR> command is received (at 115K2) during this 2 second period, the baudrate will stay at 115K2 until restart or power-cycle to allow reconfiguration.<br><br>If ‘S’ follows the rate number, 2 stop bits will be used instead of the normal 1.<br>This is primarily intended for commmunication to devices with one serial port (e.g. Arduino,AVR,PIC etc.) being used for sending DMX data, so that the same UART can be used to send DMX and receive data from the camera. (DMX uses 250kbaud with 2 stop bits)<br>It is often not possible to use rates above 115K2 on a PC when using a real COM port or COM port emulation on a USB interface due to driver limitations at the PC end, however the Thermitrack USB interface allows some extra baudrates – see section 7.<br><br>
<table>
<tr><td><n></td><td>baud</td></tr>
<tr><td>0</td><td>9,600</td></tr>
<tr><td>1</td><td>19,200</td></tr>
<tr><td>2</td><td>38,400</td></tr>
<tr><td>3</td><td>57,600</td></tr>
<tr><td>4</td><td>115,200 (default)</td></tr>
<tr><td>5</td><td>230,400</td></tr>
<tr><td>6</td><td>460,800</td></tr>
<tr><td>7</td><td>921,600</td></tr>
<tr><td>8</td><td>50,000</td></tr>
<tr><td>9</td><td>100,000</td></tr>
<tr><td>10</td><td>125,000</td></tr>
<tr><td>11</td><td>250,000</td></tr>
<tr><td>12</td><td>500,000</td></tr>
<tr><td>13</td><td>1,000,000</td></tr>
</table>
</td>
</tr>
<tr>
<td>Z<br>Factory reset</td>
<td>ZY<CR></td>
<td>=OK<CR><br>or<br>=ER<CR></td>
<td>Reset nonvolatile memory to factory default. Y is required as confirmation and ER will be returned if not present.<br>The default setting is<br>DS64 – send status once every 2 seconds<br>Baud 115K2, 8N1<br>Parameters as per table below.<br><br>Note this does not restart, so new values will only be used on the next restart or powerup</td>
</tr>
<tr>
<td>L<br><u>L</u>abel</td>
<td>L<label><CR><br>e.g.</td>
<td>=OK<CR></td>
<td>Stores up to 32 bytes of user-readable ASCII text in nonvolatile memory for</td>
</tr>
</table>

| | | | |
|---|---|---|---|
| | **LCamera 2, (North)<CR>** | | display by V command. e.g. for naming camera locations etc.. |
| **P**<br>**Set**<br>**Parameters** | **P<p>=<v><CR>**<br>e.g.<br>**P0=1<CR>**<br>**P3=10<CR>**<br><br>**P0=1<CR>P3=10<CR>**<br>as above, but can be sent without waiting for response from each, and will return<br>**=OK<CR>=OK<CR>** | =OK<CR><br>or<br>=ER<CR> | Set miscellaneous parameters.<br><p>=parameter number, <v>=value.<br><br>Values are nonvolatile and also take effect immediately.<br>See table below for parameter values & defaults.<br>=ER will be returned for invalid parameter numbers or out of range values |

| **Table 4.2 : Commands from host to camera : general-purpose commands** | | | |
|---|---|---|---|
| Command | Format | Reply | Description |
| **T**<br>**Packet Type** | **T[<pt>,<fr>[<pt>,<fr>]]<CR>**<br>e.g.<br>**TI,1<CR>**<br>outputs binary image every frame<br><br>**T<CR>**<br>Stops all autonomous packets<br><br>**TM,1,I,10<CR>**<br>Outputs M packets every frame and I packets every 10 frames<br><br>**TM,2,S,10,I,1<CR>**<br>Outputs M packets every 2 frames, S packets every 10 frames and I packets every frame | =OK<CR><br>or<br>=ER<CR> | Select packet type(s) to send autonomously, and sending frequency.<br><pt> is packet identifier as per packet type list above (case insensitive)<br><br><fr> is frame interval. 0=never, 1= every frame, 2 =every second frame etc.<br><br>Maximum value = 255 (approx 8 secs). Multiple packet types may be configured with one command. All packet types not explicitly set will be set to zero, i.e. disabled.<br><br>This value overrides the configured default ( set with D command), and will remain in force until explicitly changed or a restart/powerdown occurs.<br><br>ER will be returned and all types disabled if unknown types are requested, <fr> is greater than 255, or the syntax is incorrect<br><br>Packets will be transmitted in the order they are received from the image sensor ( which might occasionally vary) , not the order specified in this command.<br><br>This command resets the packet timers, so for example if a packet is requested to be sent every 10 frames, the first packet will arrive 10 frames later. |
| **R**<br>**Restart** | **R[W]<CR>**<br>e.g.<br>**R<cr>**<br>Resets, does not restart warmup<br>**RW<CR>**<br>Resets, restarting warmup | none. | Restart. Packet type/ interval values are set to nonvolatile defaults.<br>Depending on state of the image sensor, status will go to Startup, Warmup or active. The warmup time will not restart, so once warmed up, status will go to nomal after 2 seconds.<br>Adding W after the command will also reset the image sensor, rstarting the 14 second startup and 2 minute warmup periods. |
| **G**<br>**Get packet(s)** | **G<pt>[<pt>[<pt>]]<CR>**<br><br>e.g.<br>**GI<CR>**<br>Gets one image packet from last received frame | requested packet(s) | Get packet of type <pt>. Multiple packets may be requested, which will be returned immediately, in the order requested. The data will be that from the last frame - the command will not wait for a new frame |

| | | | |
|---|---|---|---|
| | **GIMS<CR>**<br>gets Image,Multi and Status packets<br><br>**GA<CR>**<br>gets one ASCII Image. | | If any autonomous packets are enabled, these will not appear between packets requested with this command, but may appear before the first requested packet as they could have initiated before this command was received.<br><br>Invalid packet type letters are ignored |
| **W**<br>**Wait for packets** | **W<pt>[<pt>[<pt>]]<CR>**<br><br>e.g.<br>**WI<CR>** gets one image packet from next available frame<br><br>**WIMS<CR>** gets Image,Multi and Status packets when next available.<br><br>**WA<CR>**<br>gets next ASCII Image. | requested packet(s) | As above, but waits until next frame arrives from sensor. Note that due to the relative timing of this comand and the sensor, if multiple types are requested, the packets may not always arrive in the same order, and there will often be delays between packets.<br><br>This command can also be used to force early transmission of a packet type configured for autonomous transmission. e.g. if image packets are configured to be sent every 10 frames, sending this command will return an image packet from the next frame, and reset the autonomous packet timer so the next image frame will be sent 10 frames later.<br><br>Unrecognised packet types will be ignored. |
| **X**<br>**Flush TX buffer** | **X<CR>** | none | Flush transmit buffer.<br>This immediately aborts any transmission (possibly mid-packet)<br>Can be used to help resynchronise host if packet framing is lost.<br>Also resets frame-interval timers to give host some time to get itself together before next autonomous packets arrive.<br>Bear in mind that due to buffering at the host (PC) end, you may still get some data after sending this, especially if using a USB-serial converter.<br>This command may be useful as part of the hoists error-recovery procedure, but should generally not be needed in most applications. |

**Table 4.3 : Parameters**

Parameters are set using the P command, and control various aspects of the camera's operation as described. parameters will typically only need to be configured once for a particular application.

| Number | Name | Range | Factory Default | Description |
|---|---|---|---|---|
| 0 | TrkMode | 0..1 | 0 | Set bit 0 =1 to suppress all types of tracking packets while no targets are in view. |
| 1 | ImgFilt | 0..255 | 0 | Image filtering. This controls how fast each pixel of image data is allowed to change, higher values = slower.<br>This is useful to reduce the amount of noise in the image data. Higher values will have the effect of making fast-moving targets less visible.<br>Values around 128 to 200 will mostly just reduce noise. Values above about 250 are only useful if very 'smooth' output is required. At the highest end, the filtering will fight with the sensor's fade-out characteristic to produce not much image at all! |
| 2 | ImgMode | 0..7 | 0 | Bit 0 = 1 to flip image horizontally<br>Bit 1 = 1 to flip image vertically<br>Bit 2 = 1 to suppress the hourglass icon display during warmup ( no image packets will be sent during warmup) |

| | | | | |
|---|---|---|---|---|
| | | | | (Bits 0..7 = 1,2,4,8,16,32,64,128)<br>e.g. to suppress the houglass and flip horizontally, set P2=5 |
| 3 | ImgMin | 0..255 | 112 | Image black level. Set to 128 to suppress cold targets |
| 4 | ImgMax | 0..255 | 144 | Image white level Set max<min for negative image. max=min will use factory default values.<br>Nominal image range is centred on 128, representing the background temperature. ImgMin/ImgMax determine image values returned for 0 and 255 respectively. Values further from 128 decrease sensitivity – these can be used to detect unusually hot targets, e.g. cigarettes.<br><br><u>Examples.</u><br>ImgMin=0, ImgMax=255 is the least sensitive, only exteremely hot targets (e.g. flames) will produce output pixel values approaching 255<br><br>Imgmin=128, Imgmax=144 will suppress cold targets and the 'cold-shadow' that typically follows moving targets.<br><br>Imgmin=144, ImgMax=112 will invert the image so hot targets produce lower pixel values.<br><br>These settings do not affect tracking sensitivity. |
| 5 | NetID | 0..9 | 1 | Network ID for multi-camera setup. 0 = none.<br>If nonzero, Packets will be prefixed by <n>, where <n> is '1' to '9'<br>See section on networking for mode details. |

**5 Packet types (Camera to host)**

The camera can output various types of data as specified by the configuration and/or commands from the host.

Each packet is identified by a single initial byte value, comprising an ASCII character.
Packets are prefixed by a single digit network ID, which is 1 by default for a single-camera system. This can be disabled if required by setting the NetID parameter to 0, but this is not recommended as it will make the output incompatible with networked configurations.
ASCII mode packets are terminated by a <CR> character. Binary mode packets are of fixed length or a length indicated by a byte within the packet
Binary mode packets are intended for use with hosts with limited processing power ( e.g. Arduino etc.) or where communication bandwidth must be minimised. It is reccommended that ASCII mode packets are used where possible, as these are compatible with multi-camera networked configurations.

The format of ASCII mode packets is chosen such that the packet ID byte and terminating <CR> never appear within the packet, so packet framing can be done solely by looking for specific characters in the incoming data stream. Further validation can be done by checking the packet length. A partial exeption to this is image mode, where the binary image data may contain packet ID characters, however the packet type character I and the <CR> character are filtered out of the binary data so these will never be seen within the packet.

In binary mode packets, the packet ID values and <CR> may appear in the data value, so care should be taken to ensure correct packet framing, based on known packet lengths. It is also reccommended that where possible, timeouts should also be used in binary mode to ensure correct framing. There will generally be no significant inter-byte delay between bytes within a packet, except possibly at the very highest rates, so a byte timeout of a few milliseconds will ensure correct framing. Note that framing by timing will generally not be reliable when running through USB-serial converters, as these typically buffer data internally and send in larger bursts, so timing gaps may get lost or change.

**Table 5.1 : packet types**

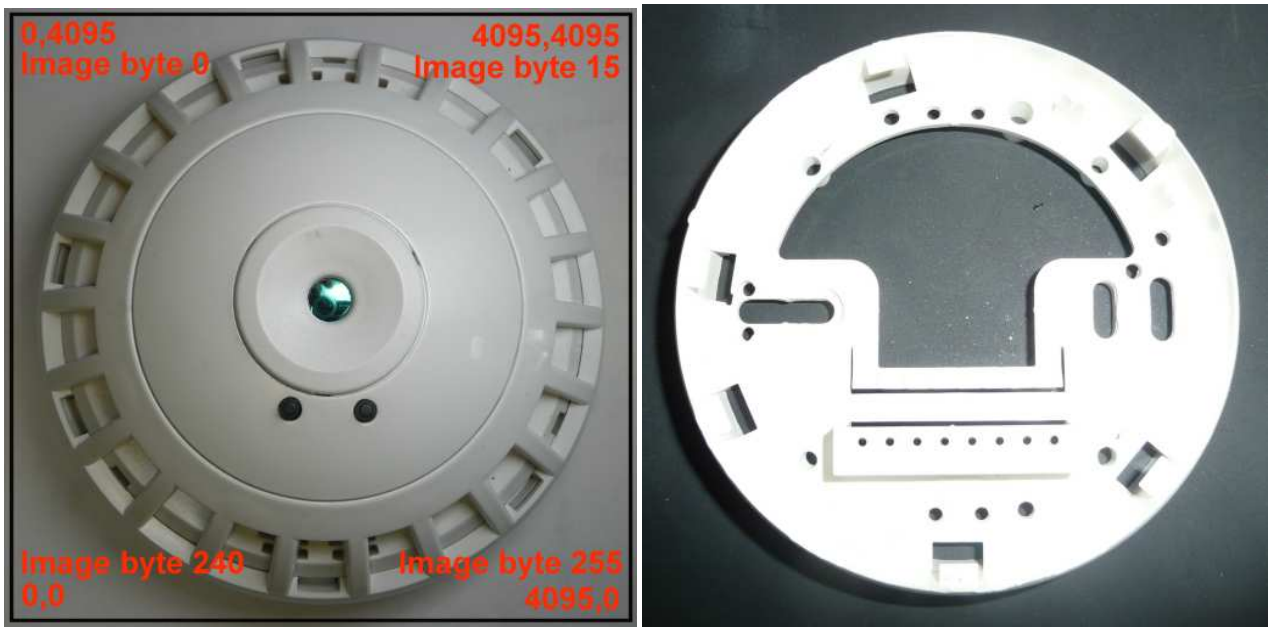
| Packet identifier ASCII (hex) Description | Format, examples Binary mode examples use the same values as preceding ASCII mode ones. | Length (bytes), including network ID digit and<CR> if applicable. length will be 1 less if netID disabled (=0) | |
|---|---|---|---|
| **S (0x53) Status (ASCII)** | **<id>S<nn>,<pp>,<c><CR>**<br>**e.g.**<br>**1S99,00,0<CR>**<br>**1S02,50,0<CR>** | 10 | <s> is one of :<br>99 : Startup mode<br>88: warmup mode<br>77: fault<br>or two ASCII digits indicating the number of targets currently being tracked from 00 to 15.<br><br><pp> is a 2 digit value showing the number of packets of all types (including this one) which have been sent since the last status packet. This value will be 99 if more than 98 packets have been sent.<br>This can be used to detect lost packets or other comms problems.<br><br><c> indicates communications state<br>0 : OK<br>bit 0 set : a transmit buffer overrun has occurred since the last "S" packet. This occurs if the output data rate in the present configuration exceeds the bandwidth of the selected baudrate. The state is cleared once the status packet is sent.<br>Bit 1 set : a receive buffer overrun has occurred since the last S packet. This is typically due to commands ( or forwarded packets in networked setups) being sent too quickly, or malformed commands with missing <CR> terminators |

| | | | |
|---|---|---|---|
| **O (0x4f) One-target tracking (ASCII)** | **<id>O<n>,<x>,<y><CR>**<br>e.g.<br>**1O6,3037,3130<CR>**<br>**1O7,2597,2181<CR>**<br>**1O0,0000,0000<CR>** | 14 | This mode tracks only the first target to enter the field of view. Subsequent targets are ignored until the first one disappears.<br>When a target disappears, if another target is in view, tracking will switch to that target and the tag will increment. If multiple targets are in view, the one chosen will be effectively random.<br><br>If <n> is "0" (0x30), no targets are in view and the X,Y co-ordinates will be set to 0000.<br>Otherwise <n> is a target tag (one ASCII digit 1..9), which increments each time that tracking changes to a new target, wrapping from 9 to 1.<br><br><x>, <y> are target co-ordinates, 4 ASCII digits 0000..4095<br><br>If the autonomous frame interval has been set to more than 1 (every frame), the tag value may be seen to increment by more than one between packets – this indicates that more than one target has disappeared and reappeared during the skipped frames. |
| **U (0x55) One-target tracking (Binary)** | **<id>U<n><x><y>**<br>e.g.<br>**31 55 06 0B DD 0C 3A**<br>**31 55 07 0A 25 08 85**<br>**31 55 00 00 00 00 00** | 7 | As above but binary output format<br><n> is 1 byte as above, i.e. 0x00..0x09<br>Co-ordinates are 2 bytes, MSB first, range 0..4095 (0.. 0x0FFF).<br><br>Note : This packet type is not supported in networked mode. |
| **M (0x4D) multi-tracking, (ASCII)** | **<id>M<n>[,<t1>,<x1>,<y1>]..[,<tn>,<xn>,<yn>]<CR>**<br><br>e.g.<br>**1M00<CR>**<br><br>**1M01,36,1864,1942<CR>**<br><br>**1M03,36,1881,2058,37,2501,1735,38,1272,2763<CR>** | Variable 5+ntargets*13 (ntargets is the number of targets currently being tracked) | <n> is number of targets in list, 00..15<br><t> is target tag, 00..63.<br>The tag remains the same for the life of the target. New target tags are allocated in numerical order, 00 to 31 for hot targets, 32 to 63 for cold ones, wrapping to 00 or 32 respectively.<br><x>,<y> are x/y co-ordinates, 0000..4095<br>As targets may disappear at any time and new targets may reappear at any position in the list, no meaning should be inferred from the order of the list – only the tag should be used to keep track of the life of individual targets. |
| **N (0x4E) multi-tracking (binary, variable length)** | **<id>N<l><n><t1><x1H><x1L><y1H><y1L>…..<tn><xnH><xnL><ynH><ynL>**<br><br>e.g.<br>**31 4E 01 24 07 48 07 96**<br><br>**31 4E 03 24 07 59 08 0A 25 09 C5 06 C7 26 04 F8 0A CB** | Variable 3+ntargets*5 (ntargets is the number of targets currently being tracked) | As above, binary output,<br><l> is the number of target records to follow, which equals the number of targets being tracked<br><br>Each Target record is 5 bytes :<br><t> bits 0..5 are tag – 00..0x3f<br><t> bit 5 is set for cold targets, clear for hot.<br><br>Note : This packet type is not supported in networked mode. |

| | | | |
|---|---|---|---|
| **I(0x49) Image (binary, CR-filtered)** | **<id>I<d0>..<d255><CR>** | 259 | Returns image data as 256 bytes of intensity data. To simplify framing, values 0x0D (<CR>) and 0x49 (I) will be suppressed in the data and set to the next highest code. The image data is generally noisy enough that this will not be noticeable.<br><br>Image brightness range is 0..0xFF (cold-hot). Background temperature is approximately in the middle of this range, 128. There will generally be a few counts of noise around this figure.<br>The ImgMin and ImgMax parameters may be used to control the range of the raw image data that is mapped to the 0..255 output range.<br>The ImgMode parameters allows control of the pixel order of the data.<br>A lowpass filter may be applied to the image data via the ImgFilt parameter.to reduce noise. |
| **A(0x40) Image(ASCII)** | **<id>A<0x0c><cr><br><id>A<sp><sp><br><x0y0>..<x15y0><CR><br><id>A<sp><sp><br><x0y1>..<x15y1><CR><br>..<br><id>A<sp><sp><br><x0y15>..<x15y15><CR>** | 341 | Displays a crude ASCII-Art representation of the image – this is intended for quick testing using terminal software. Terminal software must be set to add <LF> to incoming <CR> characters.<br>A clear-screen (0x0C) character is sent at the sart of each frame.<br>The ImgMin,ImgMax and flip x/y parameters do not affect ASCII image mode. |
| = | **=OK<CR><br>=ER<CR><br>=TO<CR>** | 4 (or variable for some commands) | Response to commands from host.<br>TO is RX timeout, which occurs if >2 secs elapse between charachers of a command, or a command and <CR> |

**Fig. 5.2 Co-ordinate and image data orientation**

Viewed on front of camera, with mounting bracket in same orientation.



## 6 Multi-camera networking

Up to nine cameras may be interconnected to allow them to use a single serial port. In this mode, the data signals from the cameras are connected in a ring configuration.(It is possible to use a chain configuration of pre-configured cameras, but where possible, a ring is recommended as this allows for configuration and firmware updates from the host.)
In networked mode, cameras forward incoming packets from upstream cameras, and incoming commands from the host, to the next camera in the ring. See the installation guide for wiring information.
In networked mode, Commands from the host to the camera chain must be prefixed by '*' plus one ASCII digit to indicate the camera for which they are intended. For example to read the version and configuration of the third camera, the command would be ***3V<CR>** As host commands are passed down the camera chain, the digit is decremented by each camera, until it reaches 1, at which point the receiving camera process the command. Responses to commands do not have any ID added, as it is assumed that the host knows which camera a command was sent to, so these are passed transparently through any downstream cameras.

To allow the host to identify which camera a packet originated from, a network ID is programmed into each camera. When this ID is programmed to a nonzero value, all outgoing packets are prefixed by a corresponding ASCII digit from 1 to 9. The network ID is only used to provide this packet prefix – all other addressing is done using the position in the chain. This allows network IDs to be assigned after installation. By default the ID is 1, so all packets will have this prefix.

Binary packet types U and N are not supported in networked mode and must not be enabled or requested in networked configurations.

### 6.1 Example setup

To set up a ring of 3 cameras :
Disable autononmous transmission for all cameras :

```
*1T
*2T
*3T
```

Set up network Ids :

```
*1P5=1
*2P5=2
*3P5=3
```

Set packet types, e.g. M packets every 10 frames

```
*1DM,10
*2DM,10
*3DM,10
```

Restart

```
*1R
*2R
*3R
```

You will then see the M packets coming from each camera, prefixed with their net IDs
e.g.

```
1M00
2M01,01,1234,1000
3M02,09,1000,2000,10,0100,0200
```

Note that packets will not always appear in the same order, as the timings of each camera may drift over time, and will vary slightly depending on the number of targets being tracked.

It is also possible to set baudrates on networked cameras, but the order in which they are set up needs to be considered, as if you end up with cameras with different rates, the forwarding will no longer work.
e.g. to set all cameras in a 3 camera network to 500Kbaud :
Set rates (new rates will not be used til restart)

```
*1B11
*2B11
*3B11
```

Restart cameras. Note this is done in reverse order, because after restart, each camera will no longer see commands issued at the previous baudrate, so you need to work back along the chain.

```
*3R
*2R
*1R
```

Then change host baudrate to new rate.

## 6.2 Additional Issues to be aware of when using a networked configuration.

Framing

In standalone operation, there will usually be some idle time interval between each packet, however in networked mode, it is very likely that sometimes two packets will arrive with no gap at all between them due to the unsynchronised relative timings of the cameras. This means that the host must be able to deal with the continuous stream of data and separate out the packets regardless of their relative timing.

Bandwidth & baudrate

At the default 115k2 baudrate, there is only enough bandwidth to deal with image packets from one camera at a time. The host must therefore take care to ensure that image data is never enabled from more than one camera.
Even at higher baudrates, or if image frames are sent less often than every frame, there is potential for the image packets to overflow the cameras' receive and/or transmit buffers if many (>6) cameras are sending image data. Setting the frame interval is not a guaranteed solution, as the cameras will not stay in sync with each other, so there will be occasions when all cameras went to send image packets simultaneously.
It is possible to evaluate how close a network is to the buffer-size limit using the figures shown in the V command, looking at the last camera in the chain ( e.g. *4V for a 4-camera network).
These figures show the largest number of bytes that have been in the TX and RX buffers since the last V command.
e.g. TX Buffer= 495/1024, RX Buffer= 269/1024 bytes
Shows that the worst-case TX buffer usage is 495 of 1024 bytes, and the RX usage is 269/1024. Provided these figures never approach. The 1024 byte buffer size is shown as it may change in future firmware versions.
Bear in mind that the peak data rate will vary with the number of objects in view, and will vary due to the relative synchronisation of cameras, so to establish whether a particular setup will reliably work without buffer overflows, tests should be run over time, with a lot of objects in view.

To determine the minimum required baudrate, add up the bandwidths required for the number of cameras and packet types used, and select the next higher rate. The nominal frame rate is 30 frames/second, so for example for two cameras using M and I packets :
I packets 259 bytes * 30fps * 2 cameras = 15540 bytes/sec
M packets 200 bytes (max) * 30fps * 2 cameras = 12000 bytes/sec
Total 15540+12000 = 27540 bytes/sec. baud = bytes/sec * 10, so baudrate needs to be >275400 – use 468K080 or higher

Note that the M packet rate above will be an overestimate as the famerate slows down when tracking large numers of targets.

Of course you should also ensure that the host can deal with the high data rates from multiple-camera networks.
Another issue is that RS232 interfaces are not likely to work well at rates above 115K2, especially with any significant cable length, so if you can't use USB, then a purpose-made RS422 or RS485 interface should be used.

## 7 USB-Serial adapters

### 7.1 The Tremitrack USB –RS422 interface

The driver for Thermitrack dedicated USB-Serial interface has been pre-configured with optimised settings, giving performance which can in many cases be better then 'real' serial ports. The USB interface has so far been successfully tested on a 4 camera network receiving I,M,S and O packet types from all cameras at 500K and 1Mbit data rates.
The drivers have also been configured to allow use of baudrates not normally available via the Windows COM port API, or via programs only offering the 'standard' baudrate set. When one of these rates is selected, the actual rate is set as shown below.

| Selected baudrate | Actual baudrate |
|---|---|
| 300 | 230,400 |
| 600 | 250,000 |
| 1200 | 460,800 |
| 2400 | 500,000 |
| 4800 | 1,000,000 |

### 7.2 Driver Installation - Windows

Download the USB Drivers from the resources page at www.thermitrack.com. Unzip them to a folder on your hard disk.
Plug the USB cable in – Windows should then pop up a Driver installation screen. The exact wording varies between windows versions so instructions here are generic. Do not select any 'search the internet' type options. Select the option to load the driver from a specified location, and point it to the folder where you put the drivers. It will normally go through the 'install drivers' process twice, once for the USB driver and again from the virtual COM port

Once installed, you need to check which COM port number has been assigned – you can do this vis Settings-> control panel-> system-> Hardware -> Device Manager -> Ports (COM and LPT). This should show the Thermitrack USB interface with its port number.
In some cases, a rather high COM port number (over COM20 is not uncommon) may have been assigned, due to presence of things like bluetooth or other virtual COM port type devices in the PC. Some application software may not be able to deal with high COM port numbers, so you may need to reassign it to a lower one. The COM port number can be changed by double-clicking on the Thermitrack interface line, selecting Port settings-> Advanced, and selecting a different port number. The warning about a port being already in use can generally be ignored provided the port number is not a physical COM port or actively being used by another device.

For information on installing USB drivers for other operating systems please see the resources page at www.thermitrack.com.

Power issues

Due to the power draw, it should be connected to a USB port on the PC or via a powered hub - an unpowered hub is unlikely to be able to supply sufficient power.
The USB interface will only power one camera from the USB supply. When using multiple cameras, an external power supply must be used.

### 7.3 Other USB-RS232 or RS422 interfaces

USB-to-serial (RS232) interfaces have some differences to 'real' physical COM ports due to the fundamentally different nature of the USB and serial hardware. Due to the overhead of sending small amounts of data over USB, these interfaces typically buffer characters until either some number of characters has been received , or an inactivity timeout has expired, before sending the data to the host. This can sometimes cause problems for applications like Thermitrack where a high data throughput and/or low latency is important, especially when handling image data

When using other USB-serial interfaces, it may be necessary to adjust settings in their driver to get acceptable performance. These settings (where controllable) are typically found in Settings- > System -> Hardware -> Device Manager -> Ports (COM & LPT) > [the USB port number] ->port settings -> advanced. Look for things like 'latency' or 'timeout' and set to minimum/fastest.

Usb-serial interfaces based on the FTDI (www.ftdichip.com) FT232R chip are highly recommended, in particular the US232R-100 and USB232R-10, (which have pretty blue LEDs on the side to see when data is present). This is the chip used in the Thermitrack USB interface.
When using the FTDI adapter, it is **ABSOLUTELY ESSENTIAL** to configure it for minimum latency, to ensure the maximum data throughput, especially in image mode. This can be done via Settings->control panel->system->Hardware->Device Manager. Select the USB COM port under Ports (COM & LPT) , right-click and do properties->port settings->advanced and set the latency timer to 2.
Also ensure that you are using the current version of the FTDI CDM drivers, available at www.ftdichip.com (the above options may not be available in earlier versions).

The FTDI chip also allows for nonstandard baudrates, accessible via their D2XX driver API. This API also allows a special character to be defined to trigger the send of data over the USB bus. Setting this to the <CR> character (13 decimal) will help decrease latency, especially in tracking modes which use short packets. See the D2xx programmer's guide and FT232R datasheet at www.ftdichip.com for more information. Other baudrates are also accessible via the virtual COM port interface with a little fiddling – see the FTDI appnote AN232B-05 Part II "aliasing baud rates" for more information.

## 8 Development guidelines – recommendations when writing drivers, applications etc. for Thermitrack

Accommodate the use of networked cameras where possible. As the factory default NetID is 1, it is acceptable to not handle packets without the ID (NetID set to 0) as it is unlilely that this will be needed for most applications.
Even if you ignire its value, assume the Net ID byte will be there.
Issue commands with the *<n> prefix, even if <n> is always 1, but again it is recommended that networking is supported.

Check packet lengths of incoming packets where possible and check that the <CR> is in the expected place. Reset your framing if not. Also reset framing if an unexpected packet type is seen - this typically indicates bad framing.
Any new packet types defined in future will be by default disabled, so you should never see 'new' packet types unless you explicitly ask for them.
Remember to deal with the '=' command-result packets, even if you ignore them.
Wherever possible, use the S packet type to give the user a positive indication that the camera is warming up – otherwise 'no output' for the 2 minute warmip time is likely to cause unnecessary confusion.
Do not assume there will be any time gap between packets, especially us using USB, as the RS232 to USB conversion means that multiple camera packets may end up in the same USB packet and appear to arrive simultaneously.
Avoid using (or relying on the use of) binary packet types unless there is a good reason to ( e.g. bandwidth constraints), as these are not compatible with networked mode.
Do not make assumptions about the possible range of COM port numbers. With things like Bluetooth, Irda, cellular modems and all the other stuff on typical modern PCs, COM port numbers can easily get well into the twenties.

## 9 Possible future additions/enhancements

Other information that is potentially available from the sensor – extra packet types to support these may be added in future if demand is sufficient.. Please email info@thermitrack.com with any comments, suggestions etc.
Target life : how long a target has been in view.
Target size (width/height)
Polar co-ordinates - bearing/distance of targets from a specified point
Filtering of tracking data for smoother motion.
Motion vectors. (maybe also predictive vectors based on previous motion)
Traffic rates ( new targets per sec/min etc.), avarage speeds etc.
Special formats tailored for particular software packages (e.g. vvvv)

Direct DMX output mode for custom applications to allow direct connection to lighting equipment for simple standalone applications.