Thinkify

HSP 3.0 Guide

Table of Contents

Note Regarding RF Exposure.

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance of 20cm between the radiator (antenna) and your body. This transmitter must not be co-located or operating in conjunction with any other antenna or transmitter.

FCC Notice and Cautions

Any changes or modifications to this device not expressly approved by Thinkify, LLC could void the user's authority to operate the equipment.

Compliance FCC Part 90

Thinkify RFID readers that have been configured by the manufacturer for the FCC Part 90 regulatory region. Pursuant to FCC Part 90.205, the Thinkify RFID reader's radiated power is limited to +20dBm (100 mWatts) ERP (Effective Radiated Power). Under all conditions, the Thinkify RFID reader will operate will below the maximum limit of Part 90 devices.

Acceptable Frequencies

917 to 923.25 MHz

Operation outside of these limits in Japan is not authorized!

Three preferred frequencies are:

917 MHz

920 MHz

923.25 MHz

Migrating from Thinkify HSP2_0 to HSP3_0 firmware

The 3.0 firmware command interface is similar, but not backwards compatible with the interface used in the 2.0 system. Changes were made to add functionality, and improve the usability and consistency of the command structure.
At the end of this document is a list of commands supported by the 3.0 system. Further details of these commands will be described in additional documents.

Overall command entry -

In the 2.0 system several commands were redundantly provided both as PI level commands and TR65 commands, but were actually just pass through commands to the TR65 module. Then later on in the development of the 2.0 system it was determined there was just too many commands to support this way of doing things, and the concept of sending a command preceded by a '!' would result in a command directly sent to the TR65.
The 3.0 system no longer supports this. For any command sent, the command interface parser in the 3.0 will first determine if this command is meant to be processed by the PI. If it is not recognized as a valid PI command it is then sent to the TR65. If the TR65 recognizes it as a valid command it executes it, otherwise reports UNKNOWN COMMAND.
Therefore the whole concept of preceding the commands intended for the TR65 with a '!' is eliminated. and the PI does not support any commands that are really direct TR65 control commands, but instead just directly sends these on to the TR65. Some examples of the commands this applies to are setting the attenuation and radio parameters and setting the trigger and delay parameters.

Data entry of programming data -

The development process in the 2.0 evolved as time went on. In the beginning only 96 bit EPC programming was supported along with passwords. As development proceeded it became apparent that further functionality was desired - examples being the size of the EPC memory data field and the ability to program USER memory data. As these were attempted to be added to the 2.0 command interface, the control structure got cumbersome and difficult to use. There were multiple commands and setup procedures depending on the desired actions to the tags currently being run which were unable to be changed easily etc. The 3.0 system attempts to correct this.

The concept approach in the 3.0 system is **WYEIWYP** (What You Enter Is What You Program).. A structure of 65536 (0x10000) programming database entries is maintained as a circular FIFO buffer. Each of these database entries can hold varying length data fields for the EPC and USER fields, data for the password fields, and data for the PC word. When a system trigger occurs to initiate a programming sequence, the next available entry is popped from the FIFO, and whatever data is included in that particular database entry is used in programming operations to the tag. Any programming operations for data fields of NULL length are simply not executed.
In this way all operational control on a tag that has triggered the system, for any operations, depends solely on the data entered for that particular database entry - there is no need for modes and setup commands etc.

In the initial release of the 3.0 system, this programming database can be created in two ways - through a series of WRITEITEM commands, or the system can automatically create a programming sequence list through a GENERATE command. Further control commands of this database include GETBUFFERCOUNT, READBUFFER, RESETBUFFER, and AUTOCLEAR.
One further operation beyond the data programming which can optionally be performed during a triggered loop is LOCKING the tag.
A separate document describes details of these database control commands, along with program flow and the LOCK operation control details.

System Type -

Three different system types are supported in the 3.0 system. Any PI/TR65 combination can be defined as a VERIFIER, an ENCODER, or STANDALONE system. The specification of the system type is best done in the config.ini file, but a command also exists to specify the type during runtime.
The exact differences in operations for these three types is described in a separate document.

Config.ini File -

A file will exist in the **/home/pi/HSP_3_0** folder of the Pi Linux file system. This file will specify parameters the program will initialize at reset for general system setup. It will allow setting the TCP port numbers used, the USB com port the TR65 is attached to, the overall system type, and the initial settings of the AUTOCLEAR mode and the TRIGGER type. The file is a simple text file and can be edited with any text editor. It is expected that parameters set in this file rarely change after initial system setup.

The STATUS message -

As the set of operations performed in response to a trigger is concluded, a status message is sent indicating the success/failure of all operations performed during the interaction with this tag. This is similar to the HSPDONE message in the 2.0 system, but since a different set of operations is now available to be performed the syntax of this response is changed somewhat. It is also modified in an attempt to be a more informative.
A separate document will describe the format and status codes used in this message.

System Commands -

The DISENAGE and ENGAGE commands are used to enable the acceptance of triggers by the system. The behavior of the programming database during these commands is controlled by the AUTOCLEAR setting.
Two further commands are provided - one to do a firmware update to the attached TR65, and one to easily modify the STARTUP MACRO settings in a TR65 to facilitate system setup.
A separate document will describe details of these.

List of supported commands as of version 3.0.4 5-25-2018

Programmer3.0 PI Internal Commands
QUIT - Exit the program
SYSTEMTYPE(= TYPE) - either GET the current setting of the system type, or set it to VER,ENC,or STA
GETVERSIONS - PI and TR65 firmware version
GETBUFFERCOUNT - Current number of PROG DATA STRUCT loaded
READBUFFER - Display all PROG DATA STRUCT loaded
RESETBUFFER - Clear PROG DATA STRUCT buffer
WRITEITEM=<DATA LIST> - enter 1 item into PROG DATA STRUCT buffer
GENERATE=<DATA LIST>,NUM(count) - create arbitrary number of items in PROG DATA STRUCT buffer
AUTOCLEAR=<ON or OFF> - control clearing of PROG DATA STRUCT buffer on ENGAGE
ENGAGE - START the trigger acceptance system
DISENGAGE - STOP the trigger acceptance system
TR65LOADER=<filename> - updates the firmware in the attached TR65
TR65MACROLOAD - loads the text file in the HSP_3_0 folder named TR65MacroList.txt as a STARTUP MACRO into the TR65

Programmer3.0 Specific TR65 Commands

Any Normal TR65 command may also be sent (radio control etc)
Send command with no parameters to GET the current value
HC<2 ASCIIHEX> -- CORRECT Minimum loop time control - units are milliseconds
HH<2 ASCIIHEX> -- HOLDOFF inhibit Trigger time control - units are milliseconds
HI<ACTION RETRY COUNT (1 ASCIIHEX)><OPEN RETRY COUNT (1 nibble)> -- ITERATE Controls looping retry paramters
HL<OFF> -- LOCK Inhibit LOCK command in ENCODE loop
HL<LOCK PAYLOAD (5 ASCIIHEX)> -- LOCK Set the LOCK command payload
HP<2 ASCIIHEX> -- PAUSE Delay after Trigger time control - units are milliseconds
HQ<1 ASCIIHEX> -- QUIET Delay after OPEN - units are milliseconds
HT<OFF or R or F> -- TRIGGER Control TRIGGER INPUT - off or edge type RISING FALLING
HW<1 ASCIIHEX> -- WAIT Override GEN2 ASYNC WAIT - units are milliseconds

Ethernet Communication between HOST and THINKIFY HSP3.0

The communication between a HOST system and the Thinkify HSP3.0 system is performed over 3 Ethernet PORT connections. These port connections are simple ASCII protocol Telnet type connections.

The three ports are used for command and control (called the CMD port), status of the operations inside the triggered programming loops (called the DATA port) and a diagnostic stream indicating timing and occurrence of various events in the HSP operation (called the DIAG port).

It is very useful for the HOST to log all messages on the DIAG port - especially in cases of system setup and in the troubleshooting of problems. If the system is functioning properly there is no need to have the DIAG port open, but it is generally advised to do so.

The ports are configured as SERVERS and expect the host to establish a CLIENT connection to initiate communication. The CMD port is full duplex - commands can be sent from the host and responses to commands received. The DATA and DIAG port are simplex - data is only sent from the system to the host - the only system reception on these ports is to perform the necessary TCP protocol handshaking and port termination.

The server configuration will allow up to 5 simultaneous client connections on each port - however - it is NEVER necessary to have more than one client connection to each. Code execution overhead is increased with each client connection so it is advised to only have one connection open during operation. There is NEVER any need during the program flow to disconnect and reconnect port. Connections should be done ONCE at the beginning of the operation and left connected until it is desired to completely exit the system.

However if more than one client connection is opened for some reason - the CMD port will only send responses to the CMD port which sent the command being responded to, the DATA and DIAG ports will send identical data to all open ports.

There are also some diagnostic messaging on the Linux console port - mainly concerning initialization operations before the ports are connected and status of the port connections. It can be useful to have access to this console port either through the HDMI port of the PI or through a SSH connection into the Linux system.

The system default for the port numbers are 5000 for the CMD port, 5001 for the DATA port and 5003 for the DIAG port, but these can be changed to any desired available port number in the config.ini file provided.

Syntax of the messaging formats is described in other documents but in general all messaging is simple ASCII with CRLF delimiting.

Output and Status Messaging in the Thinkify HSP3.0 system

As each triggered programming loop terminates one or more messages are sent out to inform the host of the data and status of the event in that loop. The type and format of the messages varies dependant on the SYSTEMTYPE.

All three system types allow the option of defining extra memory reads through the setup of the four READ DESCRIPTORS in the TR65. If enabled these reads will occur at the very end of the programming loop. If any of these are enabled, the first set of messages will be the results of these extra reads

Possible messages will be (one for each descriptor enabled)

P3READ<descriptor number>=<data read> or <FAIL> or <errorcode from tag>

If the SYSTEMTYPE is verifier

If the OPEN of the tag was successful the first message sent will be the EPC data read during the ACK command.

Format --
ENTRYEPC=<ASCII HEX><CRLF>

If the READ of the TID was successful (of course this only occurs if the OPEN was successful) the next message will be the data read from this memory area.

Format --
TID=<ASCIIHEX><CRLF>
or
TID= READFAIL<CRLF>

The Final message is the overall status codes of the operations performed

Format --
VERDONE=ALL:<statusbyte>,OPN:< statusbyte >,TID:< statusbyte ><CRLF>

The values and meanings of the statusbyte fields is given at the end of this document.

If the SYSTEMTYPE is ENCODER

Only one termination message is given indicating status

ENCDONE=ALL<statusbyte>,OPN<statusbyte>,TID<statusbyte>,EPC<statusbyte>,KIL: <statusbyte>,ACC: <statusbyte>,USR: <statusbyte>,PCW<statusbyte>,LCK: <statusbyte><CRLF>

The values and meanings of the statusbyte fields is given at the end of this document.

If the SYSTEMTYPE is STANDALONE

If the OPEN of the tag was successful the first message sent will be the EPC data read during the ACK command.

Format --
ENTRYEPC=<ASCII HEX><CRLF>

If the READ of the TID was successful (of course this only occurs if the OPEN was successful) the next message will be the data read from this memory area.

Format --
TID=<ASCIIHEX><CRLF>
or
TID= READFAIL<CRLF>

If all the programming operation enabled by the current programming database structure are successful, the EPC memory will be read after all programming operations complete and the status of this read will be reported.

Format --
EXITEPC=<ASCIIHEX><CRLF>
or
EXITEPC = READFAIL<CRLF>

Finally the termination status message will be sent

STADONE=ALL<statusbyte>,OPN<statusbyte>,TID<statusbyte>,EPC<statusbyte>,KIL: <statusbyte>,ACC: <statusbyte>,USR: <statusbyte>,PCW<statusbyte>,LCK: <statusbyte><CRLF>

Status message byte meanings and values

In the final status message the fields are

ALL: - this is the code for the overall success/fail of all enabled actions the entire loop
OPN: - this is the success/fail of the OPEN sequence to bring the tag to OPEN/SECURE
TID: - success/fail of the read of the TID memory area
EPC: - success/fail of the write to the EPC memory area
KIL: - success/fail of the write to the KILL password memory area
ACC: - success/fail of the wrie to the ACCESS password memory area
USR - success/fail of the write to the USER memory area
PCW: - success/fail of the write to the PC word
LCK: - success/fail of the LOCK operation

The bytewide codes for each field are similar in function to the HSP2.0 system but have different values to avoid confusion between the two systems

0x00 - SUCCESS
0x90 - indeterminate - no valid response was processed for this operation
0x9F - action for this field was not enabled
0x9E - action for this field was enabled but not executed (probably due to previous error)
0xCe - the action was performed but the tag returned an ERROR response e

IO and Timing control in the Thinkify HSP3.0 System

The TR65 module in the Thinkify HSP3.0 has two logic level inputs and 2 logic level outputs.
In the HSP3.0 system INPUT0 is used to as the source of the trigger input signal that initiates a programming loop for any specific tag. The two outputs provide signals indicating the programming loop is in progress, and at the end of the loop the success or failure of operation in the loop.

Trigger input

The INPUT0 is a TTL level driven signal. It is not optically isolated, and it is recommended that any triggering device employ external opto-isolation. Electrical specification of this port can be found in the normal TR65 documentation.
Minimum timing requirement for the trigger pulse is 3ms duration. The acceptance of a trigger is edge triggered , and can be command defined as either using a rising or a falling edge.
The TR65 module supports several commands that allow modifying some aspects of the loop timing to achieve best performance for a tag - this allows some degree of adjustment for optimization for different sizes and shapes of tags as they are passed through the system.

Commands to adjust trigger timing

HT[<OFF> or <R (rising)> or <F (falling)>] controls the trigger state
examples
HT GETS the current setting for TRIGGER
HTOFF turns off triggering
HTR enables a rising edge trigger and HTF enables a falling edge trigger
No triggers are recognized when the system is in DISENGAGE state
HP[<delay (2 ASCII HEX)>] controls delay of programming loop after a trigger is received - this allows time for the tag to be in the optimum relation to the RF antenna for the RF operations. Units are in milliseconds from 0x00 to 0xFF
examples
HP GETS the current setting for trigger PAUSE
HP0A delay 10 milliseconds after trigger before entering programming loop
HH[<delay (2 ASCII HEX)>] controls minimum timing between triggers - this can be used to prevent spurious triggers. Units are milliseconds from 0x00 to 0xFF.
examples
HH GETS the current setting for trigger HOLDOFF
HH32 inhibits triggers for 50ms after a trigger is received

Commands to further adjust timing within the programming loop

HC[<delay (2 ASCII HEX)>] controls the minimum time of the programming loop. The elapsed time a programming loop - defined as the time between the trigger and the final output messages indicating completion - can vary widely dependant on what operations are performed and the success or failure of these operations. For example if a tag fails to OPEN the loop can complete in just a few milliseconds compared with 100 or more if all operations in the loop succeeds. This variation in timing can cause problems with host systems. By setting the minimum time for the loop these variations can be minimized. Normal setting would be somewhat less than the normal elapsed time for the current set of programming actions being performed if they all succeed.
examples
HC GETS the current setting for loop time CORRECTION
HC64 sets the minimum time between trigger and output messages to 100ms
HQ[<delay (1 ASCII HEX)>] allows the insertion of a small delay to be inserted after the tag OPEN sequence completes and the READS/WRITES defined in the current programming loop occur. The purpose of this delay is to allow host systems a bit more time the input the desired programming data. In most cases this can be left at 0. Units are milliseconds from 0x0 to 0xF
HW[<delay (1 ASCII HEX)>] controls the ASYNC RESPONSE DELAY for write and lock operations. The GEN2 specification states the tag will respond with a status at any time within 20ms from the end of any command that changes internal EE memory of the tag. In normal operation an interrogator must always wait until either a response is received or this timeout occurs. However in reality most tags return their status response much more quickly than this - many within just a few milliseconds. This command can be used to shorten the time the TR65 will wait looking for a response from the tag in any write or lock operation. This is to minimize the time added to the loop in the case of a missed response. However user must be aware this setting does cause operational violation of the specification the tag is designed to - care must be taken to ensure that for the particular tag IC being used, the wait time is long enough for any normal response to be acquired
examples
HW GETS the current setting for async response WAIT
HWA sets the maximum time for the async response WAIT to 10ms
HI<ACTION RETRY COUNT (1 ASCIIHEX)><OPEN RETRY COUNT (1 nibble)> controls RETRY iteration control of the actions within a programming loop. Since the programming loop consists of several RF modulated commands to tag, then reception of backscatter response from tag, it is possible random events can cause an operation to fail. If a failure occurs in any of these discrete RF interactions, the entire programming loop is deemed to have failed. This command makes it possible to retry any failed event N times. However user must be aware that each retry can cause elapsed time of loop to increase - if retry is enabled, the spacing between trigger events must be

adjusted for worst case timing of all retries occurring. This setting is a tradeoff off between throughput time and throughput success rate. This command is more useful in a STOP-AND-GO transport mechanism than in a continuous flow transport mechanism.
Two parameters are supported - the ACTION retry count, and the OPEN retry count. The ACTION retry will apply to any6 read/write actions enabled after the tag OPEN sequence is performed. The OPEN retry will apply to the actions need to move the tag to OPEN/SECURE. Units are N+1 where N can be a count value of 0x0 to 0xF
HI GETS the ITERATION counts
HI12 would iterate 1 time on all loop action failures after open, and iterate 2 times on OPEN failures

STATUS Outputs

Hardware signals are provide to allow user to monitor the timing of the programming loop, and receive an indication of the overall success of failure of the loop with any particular tag. if desired.
The signal on OUTPUT0 of the TR65 will go high on reception of a valid trigger and will go low after the loop terminates (the final status message is sent)
If the program loop was successful in all operations OUTPUT1 of the TR65 will go high 2ms before the falling edge of OUPUT0. In this way OUTPUT0 falling edge can be used to latch the status of the loop indicated on OUTPUT1.

System Type Functions and control in the Thinkify HSP3.0 system

The firmware in the PI and TR65 in the HSP3.0 system allows configuring for 3 specific purposes. These are VERIFIER, ENCODE, and STANDALONE. This document describes the control commands to specify the system type, and program flow differences between the three types.

The system type can be specified in the config.ini file to configure the system at power up reset - this is the expected way to configure a system, since it is unlikely to change once defined for the current tag transport mechanism. A runtime command is also provided to change it - syntax SYSTEMTYPE= <VER, ENC, or STA>. Issuing a SYSTEMTYPE command with no parameters will return the current setting. In the splash messages sent to console port on power up reset the current setting in the config.ini will be reported.

Historically the concept of the system having different types comes from the initial HSP2.0 transport hardware. Most if not all HSP2.0 systems have at least 2 verifiers and one encoder type system in the transport mechanism.

The first verifier reads the pre-programmed value of the EPC field and the TID field of the tag, then the encoder performs programming operations, then the second verifier again reads the EPC and TID fields to check the validity of the programming operation. Some transport systems provide a third verifier for various purposes.
To support backwards compatibility with these type of systems the HSP3.0 system will allow system setup options to offer this same functionality (along with all the new functionality provided).

Advantages of this approach are throughput speed since each system has fewer overall operations to perform and very robust data integrity verification. of at least the EPC data. Disadvantages are system cost, complexity and the need for the host controller to access and control multiple systems with very stringent timing control.

Initial HSP2.0 systems only allowed 96 bit EPC programming. As desire to add different EPC lengths, and USER bank programming was expressed the configuration and control of multiple systems becomes more cumbersome and complicated. Also even in earlier systems the programming functions of the PASSWORD fields was never re-checked by the verifier which decreased the robustness of the approach.

To overcome these disadvantages, the HSP3.0 system offers a third system type - the STANDALONE. If a system is configured this way it will do all of the different tag access operations internally, which allows an overall system to only have one PI/TR65 combination. This can make a large difference in overall system cost and host control complexity, But since many more RF-tag interactions are necessary it will affect the necessary time the tag is in the antenna rf field pattern - and depending on many factors may slow down the throughput speed somewhat. A user must weigh the advantages and disadvantages in determining the configuration to use with their tag transport mechanism and host software control system.

Program Flow of the System Types -

All systems start off in DISENAGE mode - in this mode the TRIGGER is disabled. All timing, trigger, and general RF control commands should be sent. A suggested and superior way to perform all of these setup tasks is through the use of a reader STARTUP MACRO (described in a different document)

To start operation of all system types an ENGAGE command must be sent. The system will then wait for a valid trigger to indicate interaction with a tag should begin.
All operations with a GEN2 tag must first INVENTORY the tag to get it into OPEN or SECURE state, all system types will perform the necessary RF operations for this process. See description at end of this document for details of these operations, and controls a user can issue. Most users can just use default operational settings. The INVENTORY will return the EPC data of length defined by the PC word, and the HANDLE to be used in all subsequent operations - see GEN2 specification.
If the INVENTORY operation fails there is no way to interact with this tag - the programming loop will terminate with no further actions. The status message will indicate this has occurred.

If a system type is configured as a VERIFIER
After tag reaches OPEN/SECURE, a read of 6 words starting at address 0 of the TID memory bank is performed.

If the TID READ fails, the loop terminates and a status message is sent indicating this failure. Since to get to this point the INVENTORY was by definition successful, the EPC data acquired will be reported along with a message indicating the TID read failure.
The HSP3.0 extends VERIFIER operation beyond the 2.0 system. If the TID read is successful, the four READ descriptors in the TR65 are checked. If any are enabled further reads of up to four memory areas defined in these read descriptors is performed. This allows memory areas such as passwords, or user memory, or EPC memory beyond the length defined by the PC word to be also be read during the triggered operation. Of course any additional RF operations enabled require extra time to execute and will impact the throughput rate of the verifier.

The setup of the read descriptors is beyond the scope of this document - refer to the normal TR65 user manual.

The system then issues a series of messages and returns to looping waiting for a trigger. These messages will include the EPC data from the INVENTORY (if successful), The TID data from the READ, any data from descriptor driven reads and finally a status message indicating the failure/success of all operations.

If a system type is configured as an ENCODER
After tag reaches OPEN/SECURE, the programming structure database is checked for having a valid entry. If it has one, it is read from the FIFO and any non-NULL length fields will drive write operations. The check for non-NULL length and RF operations will occur in the order

EPC
KILL PASSWORD
ACCESS PASSWORD
USER MEMORY
PCWORD
In the event any of these operations are enabled but fail (as indicated by either nonreception of the tag status message or reception of the tag status message with error indication) the ENCODE loop will terminate immediately and no subsequent programming operations will occur even if enabled by non-NULL length fields. The status message will indicate this failure and any operations enabled but not performed. If all programming operations are successful, the status of the LOCK enable is checked. If it is enabled a LOCK operation will occur. A separate document will describe enabling the lock and setting the lock payload.

If the LOCK operation is successful the 3.0 system again extends the functionality of the system by checking the TR65 4 READ descriptors as described above for the VERIFIER. The ENCODER will always terminate it's loop with the status message and then idle waiting for the next trigger to occur.

If a system type is configured as STANDALONE
After tag reaches OPEN/SECURE, the actions of both the verifier and encoder are performed.
The TID is read, the programming data base is accessed, all enabled programming actions occur, the lock occurs. A standalone system will then do final read of the EPC data, then do the check of the 4 read descriptors. upon success of all these operations the ENTRY EPC is reported, the TID read is reported, the EXIT EPC is reported, any READ descriptor results are reported and finally a status message is sent indicating all success and failures.

Appendix - description and control of the INVENTORY process

In most cases the user can just use the default settings for the inventory process. There are two aspects of this that can be controlled for fine tuning a system if desired. These include controlling the SELECT used and allowing an ACCESS operation to occur. Only fairly non-standard use cases would use these
TODOTODO - add details of these - TODOTODO for now just use the defaults
In general the default INVENTORY used consists of the RF operations
SELECT ALL,
QUERY, with Q of 0,
ACK,
REQRN
to bring the tag to OPEN/SECURE. See Gen2 spec for description of these and the inventory process.

Using the LOCK function in Thinkify HSP3.0

In any 3.0 system configured as an ENCODER or STANDALONE the GEN2 LOCK functionality can be added to any tag programmed. See GEN2 spec for the full description of the LOCK functionality.

If the LOCK is enabled it will be the last thing programmed to the tag after all other enabled memory areas are complete. Since the 3.0 programming aborts on any error this means only tags fully programmed with all desired data will be issued a LOCK command.

From a power up reset the LOCK functionality is disabled. If desired a TR65 STARTUP MACRO command could override this and enable the LOCK by setting a payload. Or at any time the host can issue a command to enable and specify the LOCK function.
The LOCK command is a TR65 command - HL[<payload>]. HL alone returns the current setting of the lock PAYLOAD. If the command is entered with a payload of OFF the lock is disabled. If the payload is 5 ASCII HEX characters the LOCK function is enabled and the payload entered defines the characteristic of the lock as defined in the GEN2 specification.

Any LOCK payload setting is applied to all subsequent tags until explicitly overwritten or a reset of the TR65 occurs.

Interest has been expressed in the possible addition of other command formats to minimize the need for the user to fully understand how a lock payload is formed. This may be added in future versions, but the direct payload entry will always be supported.
It is a characteristic of the GEN2 specification that there is no means to read back the value of the lock payload as set into the tag. If the tag returns a valid response of either OK or ERROR in response to a LOCK command then the status of the lock payload in the tag is deterministic - if the response is missed or unreadable then the status is undetermined. The status message at the end of the programming loop will indicate the result.

Controlling the programming list database in the Thinkify HSP 3.0 system

The data to be programmed in to a series of tags in the HSP3.0 system is controlled by a database of data strings for the various fields supported by the EPC gen2 specification. This database exists in systems that have been defined as ENCODERS or STANDALONE, Systems defined as VERIFIERS do not support this database.
This database is organized as a circular FIFO of 0x10000 entries, with each entry a structure possibly holding data to be written into the EPC memory bank, the USER memory bank, The ACCESS PASSWORD, the KILL PASSWORD, and the PC word.

The EPC and the USER data fields in each structure can be varying length - in unit of 16 bit words, from 1 word to 8 words. Further the base address of the first word in the field can be specified - then all other words will be written to consecutive addresses.
The Password fields must be exactly 2 16 bit words, and the PC field must be exactly 1 16 bit word.

Any field can also have NULL length if it is desired to not do operations on that memory area for this particular tag.

As tags are processed by the system in response to input triggers, the next available programming structure is popped from FIFO database, and the data of any field of non-NULL length is attempted to be written into the tag memory.

If the database UNDERFLOWS - i.e. there is no available entry when a valid trigger occurs an ERROR will be logged in the status message and diagnostic stream. No programming operations will occur.

If the database OVERFLOWS - i.e. more entries than the 0x10000 allowed are attempted to be entered, an error message will indicate this condition and the entry will be ignored.
In the sequence of operations on any particular tag - if any error occurs all subsequent operations will be aborted. The status message will show the results of all possible operations.

As an entry in the database is used for a programming sequence, all fields in that entry will be reset to NULL length so they cannot be reused. Some interest has been expressed in allowing the reuse of an entry in the case of a tag failing a programming operation, such that the next tag will then use the same database entry - this is being evaluated and may be added in future versions.

Two commands exist to enter data fields into the database structure entries.
WRITEITEM=<datalist>
and

GENERATE=<datalist>,NUM<number of entries to create>
The WRITEITEM command is used to enter the data for one database indexed structure. The GENERATE command can automatically create an arbitrary number of database structure entries - up to the full 0x10000 allowed with one command.

The GENERATE command was created in HSP3.0 because it was noted that a common tag programming scenario is to program a fixed set of data and one incrementing word into tags as they trigger the system. The GENERATE command allows filling the database with data that follows this pattern. Using it frees up the host system from having to enter data for every tag as it triggers the system - instead the database can be created for a whole field of tags with one command. This relaxes the communication timing requirements of the system considerably. The command would be sent before the transport mechanism is started, then as tags trigger the programming loop subsequent enteries would be programmed into each tag.

The WRITEITEM command can be sent at any time as tags are triggering the system - the only requirement is that at least database entry has been filled at the time the GEN2 OPEN sequence of RF commands is complete. If there is no database entry available at this time an UNDERFLOW will occur and the tag will not be programmed. The condition will be flagged in the STATUS message, and in the diagnostic stream.
Multiple memory area data can be entered as sub-fields into the <datalist> of either command. Each memory area sub-field is comma delimited in the <datalist>. To specify the memory area to be used for the data in the comma delimited sub-field the first three characters must be one of the following
EPC for data in bank1 of the GEN2 tag
USR for data in bank3 of the GEN2 tag
KIL for the kill password
ACC for the access password
PCW for the PC word

Following this 3 character memory area code, data should be entered as ASCII HEX. All data is WORD based so must be in multiples of 4 ASCII HEX characters. EPC and USR fields may have any number of words from 1 to 8 (4 to 32 characters). Passwords must have exactly two words (8 characters). PC word must be exactly 4 characters for one word.

These sub field specifiers may be sent in any order inside the <datalist> as long as they are comma delimited. Any sub-field not specified within a <datalist> will result in a NULL length field for that database entry - when this entry is used as a tag triggers the system, no programming action will occur for that memory area.
There is a data modifier for the EPC and the USR sub-fields which will specify the start

address within the bank. Multiple word in either bank will then be written at contiguous addresses from the start address. If no address modifier is given it is assumed the start address for the EPC is address2 of bank1, and the start address for the USR is address0 of bank3.

To specify a non-default start address surround 4 ASCII hex characters in parenthesis immediately after the 3 character memory area specifier and before the data field.
In the GENERATE command there is one further data modifier. If any 4character WORD is preceded by a ! (ASCII 0x21) this word will be incremented by one for each subsequent entry generated.

EXAMPLES of WRITEITEM commands----
To specify one database entry for programming 96 bits of EPC only
WRITEITEM=EPC111122223333444455556666
To specify one database entry for programming 128 bits of EPC only
WRITEITEM=EPC11112222333344445555666677778888
To specify one database entry for programming 96 bits of EPC and ACCESSPASSWORD of 0xDEADBEEF
WRITEITEM=EPC111122223333444455556666,ACCDEADBEEF
To specify the above but also with KILLPASSWORD of 0xF00DFACE
WRITEITEM=EPC111122223333444455556666,ACCDEADBEEF,KILFOODFACE
To specify the above and also program user memory from address 0 with 0x123456789ABC
WRITEITEM=EPC111122223333444455556666,ACCDEADBEEF,KILFOODFACE,USR123456789ABC
To program one word with value 0xB00B of EPC at address4 of EPC memory
WRITEITEM=EPC(0004)B00B

EXAMPLES of GENERATE commands----
The NUM field in the GENERATE command specifies how many database entries to create, and is in ASCII DECIMAL
To GENERATE 10000 database entries with 80 bits of 0x11112222333344445555 then an incrementing field from 0x0000 to 0x270F
GENERATE=EPC11112222333344445555!0000,NUM10000
To GENERATE 10000 database entries with 80 bits of 0x11112222333344445555 then an incrementing field from 0x0000 to 0x270F and also program the KILL PASSWORD with ABCDEF01
GENERATE=EPC11112222333344445555!0000,KILABCDEF01,NUM10000
To GENERATE 1000 entries programming only address8 of USR memory with incrementing values from 0x2000 to 0x23E7
GENERATE=USR(00080)!2000,NUM1000

As the data entry commands are executed they will be written to the database in ascending order in a circular addressing fashion from 0x0000 to 0xFFFF, and conversely when triggers occur entries will retrieved in a FIFO manner - the oldest unused database entry will be used first.
Four other commands will affect the database.

RESETBUFFER is somewhat self explanatory - all entries in the database will be reset to NULL length and the available buffer entry count will be set to zero.

READBUFFER - All data in database will be reported - user should be aware this could result in a very long communication stream if the generate command has been used to fill a large number of database structures

GETBUFFERCOUNT will return the number of currently filled but unused database structure entries

AUTOCLEAR=<ON or OFF> controls the behavior of the buffer when the system triggering is enabled/disabled through a ENGAGE or DISENGAGE. If it is ON any transition from ENGAGE to DISENAGE or vice versa will result in a RESETBUFFER being executed.