LED Scrolling Message Board

The messaging board interface allows a user to input and edit messages.

Internally the messaging board has been set up in such a way that an action is initiated by a command string. These command strings are represented by tags.

These are the tag combinations recognized by the messaging board's interpreter. These tags are embedded in the **playlist**(s) and **message** file(s).

##**nn** -: Start of a message - where **nn** is the message number
#. -: End of a message

##* -: Start of transmission [a message file group]
#. -: End of transmission [a message file group]

#?nn -: Start of a playlist - where **nn** is the playlist number
#: -: End of a playlist

##$ -: Start of transmission [a playlist file group]
#. -: End of transmission [a playlist file group]

#()**nsc**-: Start of a message stream - where **nsc** is as defined in **EFFECTS**
#()**ntt**-: Start of a message stream - where **ntt** is as defined in **EFFECTS**
#. -: End of message stream

\ -: Escape character

#. -: Stop/ end /terminate

#*nsc -: Effect tag (dynamic effects)

#*ntt -: Effect tag (static effects)

#@n -: Repeat tag - where n is repeat count

#$n -: Loop tag- where n is the loop count

#T -: display the time

#TA -: set/adjust the time

#Fn -: Select display font- where n is the font number

## -: User defined /programmed message tag

#& -: Preloaded message tag

<u>MESSAGE FILE</u>

Any message must have a tag to mark its **beginning** and another to mark its **end**.

##**nn** -: 'Start of a message' tag - where **nn** is the message number
#. -: 'End of a message' tag

E.g. '**this is a sample message.'**

If this is message '00', then the tags embedded/generated with the message will be

##00this is a sample message.#.

a) **##00** marks the start of message 0. ##**nn** :- where **nn** is the message number. Two digits to allow expansion (to 99 messages).
b) **#.** marks the end of the message.
Note: The **#** comes after the full stop (in case a full stop is included in the message) and **no spaces** are allowed between them. A space will be treated as being a valid part of the message.

##00**this is a sample message.**#.
##01**this is another sample message.**#.
##02**this is yet another sample message.**#.

##* -: Start of transmission [a message file group]
#. -: End of transmission [a message file group]

To be correctly received, the messages transmitted must have **transmission tags**. The display has a "gate keeper" task that looks out for these tags then routes the message to the correct destination.

Note: Besides the **transmission tags, the** tags that mark the beginning and end of messages must also be included.

The message files formatted for transmission should look like this

##*##00**this is a sample message.**#.##01**this is another sample message.**#.##02**this is yet another sample message.**#.#.

a) The **##*** before **##00** indicates the **start** of **transmitted message file group**.

b) The last **#.** at the end indicates the **end** of **transmitted message file group**.

Note: the messages can be transmitted individually or as a group(s) as long as **a start message transmission** (##*) and **a stop message transmission** (**.**#) are included.

##*##00**this is a sample message.**#.#. is valid and can be transmitted.

And so is this

##*##00**this is a sample message.**#.##01**this is another sample message.**#.#.

NOTE: No **effects, repeats** or **loops** are included/allowed in the message files.
SEE: EFFECTS, REPEATS & LOOPS

<u>ESCAPE CHARACTER</u>

\ -: Escape character (used only in message files and message streams)

Format: xxx\ where xxx is a predefined character combination.

Used immediately after a predefined character combination to "escape" from the effect of the combination.

Note: A message with " **#.**" will prematurely terminate if the " **#.**" were a valid part of the message.

E.g. the message " **I intend to rob #. the bank** " will display as " **I intend to rob** " then terminate. The reason for this is that the messaging board will interpret " **#.**" as the end of the message file and truncate the file.

To display correctly included an escape character " **** " to be inserted immediately after the predefined character combination **(no spaces).**

" **I intend to rob #.\ the bank** " then displays correctly as " **I intend to rob #. the bank** ".
NOTE: To implement correctly you have to scan a **message** that the user has entered then include the escape character whenever " **#.**" is encountered.

<u>PRE-LOADED MESSAGES</u>

#&nn -: Preloaded/canned message tag

Format: #&**nn** where **nn** is the message number

Messages preprogrammed into the messaging board are

#&00 Preloaded Message 0 - 'HAVE A NICE DAY.'
#&01 Preloaded Message 1 - 'WELCOME. WE ARE OPEN
#&02 Preloaded Message 2 - 'GREAT OFFERS TODAY.
#&03 Preloaded Message 3 - 'NO SMOKING PLEASE.
#&04 Preloaded Message 4 – 'THANK GOD ITS FRIDAY.
#&05 Preloaded Message 5 - 'HAPPY EASTER '
#&06 Preloaded Message 6 - 'MERRY CHRISTMAS.'
#&07 Preloaded Message 7 - 'HAPPY NEW YEAR'
#&08 Preloaded Message 8 - 'ITS SHOW TIME'

Preloaded messages are a part of the display's firmware and can be altered on request.

USER DEFINED MESSAGES

##   -: User defined /programmed message tag

Format: ##**nn** where **nn** is the message number

A user enters a message in the user interface and the relevant tags are attached to the message before being sent to the display board.
Details are as defined in **MESSAGE FILE.**

STOP

#.   -: Stop/ end / terminate

Format: #.
Marks the end of a **message file**.

It also marks the "beginning" of a **loop**.
Normally a "stop" marks the end. In the **loop** implementation, the logic is reversed.
[More details on this in **LOOPS**]

CONFIGURATION COMMANDS (EFFECTS AND FONT SELECTION)

EFFECTS

#* -: Effect tag

There are **dynamic effects** (motion is involved e.g. scroll left, scroll right, etc) and **static effects** (e.g. ‘hang’ a stationary message for time **xx** seconds).

Dynamic effects format: #*nsc

n (one byte) - effect number /identity
s (one byte) - effect speed (how fast the effect should run
. e.g. a left scroll can scroll slowly or the speed can be increased to make it scroll faster)
c (one byte) - effect count (number of motions the effect makes)
note: effect count has nothing to do with **repeats**

NOTE: There are 8 allowable speeds (from 0 to 7).
: Default effect is 1.
: Default speed is 4.
: Default count is 1 except for **effects 1 & 2 where it is the ASCII ’E**’ and **effect 0 where it does not apply**.

Static effects format: #*ntt

n (one byte) - effect number /identity
tt (two bytes) - effect timing (length of time the effect should run before it is timed-out)
– if the effect is to run for 25 seconds then **2** will be regarded as the **high byte** and **5** the **low byte**

--
E.g. 1 #*13E#&01#*25E#&03

Split it up for clarity **#*13E #&01 #*25E #&03**

On executing, will load **effect 1** running at **speed 3** (#*13E). The effect will be performed on preloaded message **1** (#&01).

Then **effect 2** running at **speed 5** will be performed on preloaded **message 3**(#&03)**,** (#*25E).

--
E.g. 2 #*14E#&01#*015#&03

Split up **#*14E #&01 #*015 #&03**

On executing, will load **effect 1** running at **speed 4** (#*14E). This will be performed on preloaded **message 1** (#&01)**.**

Then **effects 0** (this is a static effect) on **message 3**(#&03) will timeout after **15 seconds**.

Like the preloaded messages, a user can choose effects.

<u>A suggestion:</u> you could have a pull down menu where you can enter effect parameters. Or you could have columns where the appropriate action and parameters are entered. (Your choice)

<u>Effects available</u>

Effect 0 **(Static)** **c-** blink (message blinks while stationary for time xx seconds then exits by scrolling left)

Effect 1 **(Dynamic**) **s-** scroll left
Effect 2 **(Dynamic)** **s-** scroll right
Effect 3 **(Dynamic)** **c-** Left-Right bounce (oscillates between a scroll left and a scroll right)

Effect 4 **(Dynamic)** **c-** "left-shift-in"
Effect 5 **(Dynamic)** **c-** rotate/ scroll down
Effect 6 **(Dynamic)** **c-** wipe in – invert - wipe out

Effects can also be categorized as

1) **s** (simple) : is an effect that performs one basic motion or transformation.
2) **c** (compound): is an effects that is a combination of more than one simple effect.

On executing a **'simple effect + message'** command, the last **effect** command remains active. If the same **effect** is to be performed on the **next message**, the **message** can be called without explicitly mentioning the **effect**.

<u>Example</u>

#*14E #&01 #*14E #&02

Effect 1 is a simple effect (scroll left)

On executing, will **scroll left (Effect 1)** at **speed 4** (#*14E). This will be performed on preloaded **message 1** (#&01) and then the same will happen to **message 2** (#&02).

This combination **#*14E #&01 #&02** executes in exactly the same way as

#*14E #&01 #*14E #&02

NOTE: this only happens when you are dealing with **simple effects.**
The reason for this is that **a compound effect calls** one **simple effect/motion** after another. On **exit**ing the last **simple effect/motion** is left active. This last active effect acts on the next message unless another effect is **explicitly** called.

Example

#*342 #&01 #*342 #&02

Effect 3 is a compound effect (Left-Right bounce): - oscillates between a **scroll left** and a **scroll right**. On executing will run a **simple scroll left** then a **simple scroll right** and repeat based on the count**.** Given an odd count **Left-Right bounce** exits on a **simple scroll left** and given an even count it exits on a **simple scroll right.**

#*342 #&01 #*342 #&02

On executing, will load **effect 3** running at **speed 4** (#*342). This will be performed on preloaded **message 1** (#&01)**, twice** and then the same will happen to **message 2** (#&02).

This combination **#*342 #&01 #&02 does not** executes the same way as

#*342 #&01 #*342 #&02

When generating **'simple effect + message'** combinations you could use either formats (whichever makes coding easier).

--
sample effect tags

#*015 (Effect 0) message blinks while stationary for 15 seconds then exit
#*14E (Effect 1) scroll left at **speed 4** then exit
#*26E (Effect 2) scroll right at **speed 6** then exit
#*342 (Effect 3) oscillate between a scroll left and a scroll right at **speed 2** (**twice**).
#*442 (Effect 4) scroll down at **speed 4**, **2 rolls** then exit
#*541 (Effect 5) "left-shift-in" at **speed 4** then exit
#*672 (Effect 6) wipe in- invert - wipe out at **speed 7**, **2 wipes** then exit.

The number may increase later.

FONT SELECTION.

Format: #Fn -: Select display font- where n is the font number

Action: When the tag is encountered all messages after the tag will be converted to the selected font till the next font change tag is encountered.

You can drop it into the playlist before "effect details"+ " message details" combination.

Available fonts are

NORMAL_FONT_SIZE
SMALL_LOWERCASE
BLOCK_UPPERCASE
WIDE_BLOCK_UPPERCASE

The GUI replaces them in the background with the appropriate tags. (#F0, #F1, F2, #F3)

NORMAL_FONTSIZE	= #F0
SMALL_LOWERCASE	= #F1
BLOCK_UPPERCASE	= #F2
WIDE_BLOCK_UPPERCASE	= #F3

Default font is '0'

<u>**FLOW CONTROL COMMANDS**</u> (REPEATS AND LOOPS)

<u>**REPEAT**</u>

#@ -: Repeat tag

Format: #@**n**

Repeats the " **effect + last message " n** times.

<u>**Example 1**</u>

……..#*14E#&03…… **#*14E #&03**
Interpreted as **effect 1 at speed 4** on **prerecorded message 3.** This runs then moves on to the next operation.

<u>**Example 2**</u>

……..#*14E#&03#@5 …… **#*14E #&03 #@5**
Interpreted as run **effects 1 at speed 4** on **user-defined message 3** then **repeat 5 times** before moving to the next operation.

LOOP

#$n -: Loop tag

Format: #$n

A loop instruction (**#$n)** runs everything between it and the last **stop** (**#.**) encountered earlier.

On execution, a **loop** instruction causes the interpreter to reverse to the last **stop** (#.), and then acts on all **messages** and **effects** after the **stop**. If a **loop**ing were to be done between **'effects X_1 + messages'** and **'effects X_n +messages'** no **stop** (#.) should be included between them.

As seen earlier, any new effect and message can be called without 'STOP'ing the previous one. In this scenario the **previous effect** and **message** is enqueued. A '**LOOP**' instruction runs enqueued messages and effects.

The difference between a **loop** and a **repeat** is the "operand" range.

A repeat acts only on the last '**effect + message'**. Looping operand range extends till the last stop.

Nested loops are supported. (A loop inside a loop)

Example 1

……..#*14E#&03#.#*355#&03…… **#*14E #&03 #. #*355 #&03**

The stop between **#&03** and **#*355 terminates** and **resets** everything that had happened earlier. In this case **#*355 #&03** remains enqueued.

Example 2

……..#*14E#&3#*355#&3…… **#*14E #&03 #*355 #&03**

After running, **#*14E #&03 #*355 #&03** remain enqueued till the next stop is encountered.

Example 3

#*25E#&1#*25E#&2#.#*121#&6##4#&4##3#$2

Split up #*25E #&1 #*25E #&2 #. **#*121 #&6 ##4 #&4 ##3** #$2

Step 1: On executing, will load **effect 2** running at **speed 5** (#*253). This will perform the effect on preloaded **message 1** and on **message 2** then **stop (#.)**.

Step 2: Next will load **effect 1** running at **speed 2** (#*12). This will perform the effect on preloaded **message 6**,user defined **message4,** preloaded **message 4**, user-defined **message3**.

Step 3: (A loop instruction is encountered). The interpreter will reverse to the last stop #. and repeat everything after the stop (**Step 2**). All these [#***12 #&6 ## 4 #&4 ##3**]

Implementation.

1) Call a stop
2) Place all the stuff in the loop,
3) Call a loop (and number of times to loop)

……. #._all_the_stuff_in_the_loop_#$**n**……… no spaces.

(**A suggestion**: in your GUI there could be a provision to select a loop.
A user can **check/tick** where he wants the loop to start and where he wants it to end.
Your tag generator can then insert

1) A stop before a **check/tick** where loop starts
2) A loop tag after a **check/tick** where loop ends.

PLAYLIST FILE

A **playlist** is a list of **message files** (file tags represent the message files) and **commands** on what to do with the files. There are configuration commands **(effects, font selection, playlist timing and delimiters)** and flow control commands **(repeats** and **loops).**

The **playlist** is read by an **interpreter** that **tries** to do as the playlist says. Without flow control, it performs a simple playlist read from beginning to end. **Repeats** and/or **loops** alter read flow.

Before a message file is read, the reader has to be **configured** (either explicitly or as implied in the definitions). The **message reader** has to know how to read (from **right to left** or from **left to right** and the expected **character font**).

A **playlist** in its basic form is something like this

"Start of a **playlist** " "effect details"+ "message details" "effect details"+ "message details" "effect details"+ "message details" "End of a **playlist**"

#?nn -: Start of a **playlist** - where **nn** is the **playlist** number
#: -: End of a **playlist**

This is a sample playlist. #*132#&01#*13E#&02 . For clarity this will be represented as **' _ playlist _detail_' .**

If this is playlist '02', then the tags generated with the playlist will be

#?02CACACACA**_ playlist _detail_**#:

b) **#?02** marks the start of playlist '02'. #?**nn** - where **nn** is the playlist number. Two digits to allow future expansion (up to 99 playlist).

c) '**CACACACA'** is the playlist timing info. When a **start** time and **stop** time are not set by the user the default tag will be **CACACACA** (8 bytes).

For timing purposes, I use a 24HR format.
If start time is **09.30**AM and timeout is **10.45**AM then time tag will be **09301045** (8 bytes).
If start time is **02.30pm(14.30hrs)** and timeout is **04.15pm(16.15hrs)** then time tag will be **14301615** (8 bytes).

d) **#:** marks the end of playlist.

This is another sample playlist. #*253#&1#.#*123#&6#@2
It is set to start at **10:45am** and timeout at **11:30 am**

If this is playlist '05', then the tags generated with the playlist will be
#?0510451130**_ playlist _detail_#**: (for clarity) or

#?0510451130**#*253#&1#.#*123#&6#@2**#: .
.

a) **#?05** marks the start of playlist '2'. #?**nn** - where **nn** is the playlist number
b) **#:** marks the end of the playlist.
c) **10451130** is the playlist timing info.

Playlist examples

Playlist 0	#?00CACACACA**_ playlist _detail_#:**
Playlist 3	#?0310451130**_ playlist _detail_#:**
Playlist 5	#?0510001210**_ playlist _detail_#:**

##$ -: Start of transmission [a playlist file group]
#. -: End of transmission [a playlist file group]

Note 1: These tags are used during file transmission to mark the start of a transmitted playlist file
Note 2: Playlist can be transmitted individually or as a group(s) as long as **a start playlist transmission** (##$) and **a stop playlist transmission** (.#) are included.

Example 1

If playlist '00' is to be transmitted alone, then it will be formatted as shown below.

##$#?00CACACACA**_ playlist_detail_#:#.**

Example 2

Playlist '00' and Playlist '03' (On transmission)

##$#?00CACACACA**_ playlist_detail_**#:#?0310451130**_playlist_detail_**#:#.

Example 3

Playlist '00', Playlist '03' and Playlist '05' (On transmission)

##$#?00CACACACA**_ playlist _detail_**#:#?0310451130**_ playlist _detail_**#:#?0510001210**_ playlist _detail_**#:#.

All the files above are valid for transmission.
Note: no spaces between sequences or start/ terminating characters

<u>REMEMBER</u>: **'effect + message' tag combination cannot have anything between them.**
Repeats, loops, "show time" or "font selection" tags are place either before or after them.

#T -: display time

Format: #Tss

Displays the time for a period of **ss** seconds. The instruction can be included in a playlist so that as the playlist is running it displays the time when the tag comes up.

Time can also be displayed in streaming mode.

Format: #()**ntt**#Tss#. Note the time is read from the messaging board's clock. The format is as defined in **MESSAGE STREAM**

Format: #()**ntt**12:30#. Note the time value is generated elsewhere then sent as data to the messaging board. The format is as defined in **MESSAGE STREAM**

On the messaging board all playlist are sorted by starting time. In case a Playlist file doesn't have a time tag then it will be sorted by Playlist number.

<u>Note:</u>
In case two sequences have overlapping times then the one with an earlier starting time gets to run first. When the next sequence's starting time comes up, the running sequence will be terminated. **Starting time will always override stopping time.**

<u>Example 1</u>

Sequence 1	10.30 to 12.30
Sequence 2	10.32 to 12.30

Sequence 1 will run from **10.30** to **10.32**, then **sequence 2** from **10.32** to **12.30**

<u>Example 2</u>

Sequence 1	10:30 to 12:55
Sequence 2	12:00 to 12:05

In this scenario, **Sequence 1** runs from **10:30** to **12:00** then **sequence 2** runs from **12:00** to **12:05** .

<u>Example 3</u>

Sequence 1	10:30 to 12:00
Sequence 2	10:30 to 12:30

In the third scenario, **Sequence 1 never** gets to run**.**