MEMO Customization Guide

Introduction

Tablet allows to customize:

- Graphical Interface
- LoRaWAN Interface: Uplinks and Downlinks format

Customization is done via editing configuration files in YAML format.
(for detailed YAML format description here: YAML Ain't Markup Language (YAML™) revision 1.2.2)

Configuration Files

---

elements.yml - defines graphical elements which are displayed on the screen
texts.yml - defines all static texts which are used in graphical interface
styles.yml - defines style attributes of graphical element, e.g shadows, text alignment, etc
uplinks.yml - defines format of all UL messages and events which trigger uplink
dlinks.yml - defines format of all DL messages and events which are triggered when DL is received

How to apply new configuration?

---

1. Change configuration files according to your preferences
2. Upload configuration files to the tablet via USB
3. Reboot tablet using Reset button

Graphical Interface customization

Tablet allows to compose any interface from graphical elements.

Next graphical elements are available:

- Label - displays any text. E.g. “Room humidity is 65%“
- Panel - rectangular filled with color
- Image - display picture
- Button - touch button

How to create Graphical Interface?

1. Create elements.yml
2. Specify all graphical elements in the file
3. Specify attributes and position of the each element
4. Upload this file to the Tablet

5. Reboot Tablet

Example of elements.yml file

It includes configurations for 4 different graphical elements (label, image, panel, button)

```yaml
---
Elements:

# Product Name label
-
    id: lab_productName
    type: label
    style_id: blackLab_stl
    font: chivo40.bin
    x: 0
    y: 135
    width: 1024
    visible: 1
    text_id: cust1_txt

# Company logo black image
-
    id: img_logoBlack
    type: image
    x: 681
    y: 50
    image: LOGO_B.bin
    visible: 1

# Bottom panel
-
    id: pnl_bottom
    type: panel
    style_id: btnPanel_stl
    x: 0
    y: 586
    width: 1024
    height: 179
    visible: 1

# Some button
-
    id: btn_custScr1_1
    type: button
    style_id: bookNowBtn_stl
    label_style: whiteLab_stl
    font: chivo40.bin
    x: 5
    y: 588
    width: 1014
    height: 164
    text_id: cust2_txt
    visible: 1
```

Allowed Symbols in elements.yml :

IMPORTANT: Make sure you do not use not use symbols “#”, “!” and “@” which are special symbols in YAML (for example “#” is a symbol of comment line in YAML). If it is really needed you may use
“!” and “@” symbols (but not “#” symbol) for string values inside double quotes.
For example:

> wrong usage:
> `label_style: cool_style_@1`

> right usage:
> `label_style: "cool_style_@1"`

File Structure of elements.yml :

```
---
Elements:
-
    <screen element 1 configurations>

-
    <screen element 2 configurations>
...
-
    <screen element N configurations>
```

- <screen element 1 configurations>, <screen element 2 configurations> and <screen element N configurations> – sets of configurations for screen elements 1, 2 and N respectively.
- document starts with “---”.
- “Elements:” is on the next line.
- The configurations of the screen elements are placed after “Elements:” line.
- The configurations of every screen element start with “-” on separated line.

Concept of Screens:

Screen is a container for graphical elements.
It represents all the area of physical tablet screen, which can be used to display elements.

Initial Screen

Tablet always starts from the initial screen.
Initial screen is displayed until tablet receives LoRa downlink with command to switch to another screen. (pls see Custom Screens section).

Initial Screen displays tablet related data: firmware versions, logo, etc.

Graphical elements which are displayed on Initial Screen has next reserved ids:

lab_productName - product name
lab_fwVer - tablet firmware version
lab_tsFwVersion - touch screen controller virmware version
lab_bootloadVer - bootloader version
lab_apploadVer - application loader version
lab_iteFwVer - display controller firmware version
lab_loraStatus - statuses of connection to LoRa network (possible values: WAITING TO JOIN (before joining to the network) and JOINED SUCCESSFULLY (after it))
img_logoBlack - Tektelic logo
pnl_bottom - panel with LoRa statuses

This elements can be altered or removed from Initial Screen if needed.

Custom Screens

Aside of Initial Screen it is possible to create up to 4 custom screens.
Switch to particular Custom Screen is done via LoRa downlink.
Each custom screen can contain:

- up to 10 labels
- up to 6 buttons
- up to 5 panels
- up to 5 images

  NOTE:
  elements.yml contains all graphical elements for all Customs Screens and Initial Screen.

Description of Graphical Element Fields

Mandatory Fields for ALL graphical elements

- **id** - is used to identify each graphical element.

  id format MUST follow naming convention:
  <type>_custScr<number of the screen>_<number of the element with this type on the screen>

  - type - first 3 symbols which contain graphical element type:
    - “lab” for labels
    - “img” for images
    - “pnl“ for panels

    - "btn" for buttons

  - number of the screen - number from 1 to 4
    It is possible to have up to 4 Custom Screens (pls see Concept Of Screens section)

  - number of the element with this type on the screen - uniquely identifies each element on the Screen. Next values are allowed:

    - from 1 to 10 for labels
    - 1 – 6 for buttons
    - 1 – 5 for panels
    - 1 – 5 for image

  - **id** examples:
    - "*lab_custScr1_1*" - label placed on 1st Custom Screen with identifier 1
    - "*lab_custScr1_2*" - label placed on 1st Custom Screen with identifier 2
    - "*img_custScr3_4*" - image placed on 3rd Custom Screen with identifier 4

- **type** - specify graphical element type.
  Use:
  - "label" for labels
  - "image" for images
  - "panel" for panels
  - "button" for buttons

- **x** - horizontal coordinate of the left bottom corner of the graphical element (coordinates start from the left top edge of the screen). The value of this field is limited by the resolution of the screen (value should be in 0 – 1024 range).
- **y** - vertical coordinate of the left bottom corner of the graphical element (coordinates start from the left top edge of the screen). The value of this field is limited by the resolution of the screen (value should be in 0 – 768 range).

- **visible** – screen element visibility state. Possible values for this field are:
  - 0 – screen element is invisible.
  - 1 – screen element is visible.

Fields for the Label

- **font** - name of the custom font to be used.
  The length of the name of the file has to be not more than 8 symbols and the length of the file extension has to be not more than 3 symbols. Pls refer to "How to create custom fonts? " section for details. NOTE: Default font is font with height size equal to 20. It is support Basic Latin, Latin-1
  Supplement, Cyrillic, Arabic, Arabic Presentation Forms-A and Arabic Presentation Forms-B symbols.

- **width** – the width of the label in pixels. This field is optional. If this field is not set then width of the label will set automatically equal to the width of content (text).

- **text_id** – identifier of the static text string which is displayed on the graphical element.
  All static text strings are specified in text.yml file. text_id should correspond to the particular id in that file.

- **style_id** - defines style which is applied to the graphical element.
  Style can specify colors, shadows, lines weight, etc.
  All styles are configured in styles.yml file and style_id should correspond to the particular id in that file

- **calc_id** – used to display system information.
  If **calc_id** and **text_id** both are specified for the same label - **calc_id** field is ignored.

  Possible values for **calc_id** field are following (other values will be ignored):
  - *regStr_calc* – firmware version with regional belongings string.
  - *bootloadVer_calc* – bootloader version
  - *apploadVer_calc* – application loader version
  - *iteVer_calc* – ITE firmware version
  - *loraStatus_calc* – the state of the connection to LoRa network
  - *tsFwVer_calc* – touch screen firmware version

Fields for the Panel

- **style_id** - defines style which is applied to the graphical element.
  Style can specify colors, shadows, lines weight, etc.
  All styles are configured in styles.yml file and style_id should correspond to the particular id in that file.

- **width** – the width of the panel in pixels

- **height** – the height of the panel in pixels

Fields for the Image

- **image** – name of the image file that is the source of image. The length of the name of the file has to be not more than 8 symbols and the length of the file extension has to be not more than 3 symbols. This file has to be in binary format. (pls see How to prepare Images? section)

- NOTE: image dimensions are based on actual image size in pixels. Tablet cannot rescale images.

Fields for the Button

- **style_id** - defines style which is applied to the button
  Style can specify colors, shadows, lines weight, etc.
  All styles are configured in styles.yml file and style_id should correspond to the particular id in that file

- **label_style** – defines style which is applied to label which is placed on the button
  All styles are configured in styles.yml file and label_style should correspond to the particular id in that file

- **font** – the name of the font file that is the source of font for label viewed inside of the button.
  The length of the name of the file has to be not more than 8 symbols and the length of the file extension has to be not more than 3 symbols. Pls refer to ”How to create custom fonts? ” section for details. NOTE: Default font is font with height size equal to 20. It is support Basic Latin, Latin-1
  Supplement, Cyrillic, Arabic, Arabic Presentation Forms-A and Arabic Presentation Forms-B symbols.

- **text_id** – identifier of the static text string which is displayed on the button label .
  All static text strings are specified in text.yml file. text_id should correspond to the particular id in that file.

- **width** – the width of the button in pixels.

- **height** – the height of the button in pixels.

How to add static texts?

1. Create texts.yml
2. Specify all static texts in the file (use only predefined ids for texts)
3. Link static text with graphic element you want to use this text using id
4. Upload texts.yml and elements.yml files to the Tablet
5. Reboot Tablet

texts.yml configuration file example:

```
---
Texts:
# Some custom text
-
    id: cust1_txt
    value: "my text"

# Another custom text
-
    id: cust2_txt
    value: "my another text"
```

NOTE: Up to 30 texts can be created.

Each text object contains 2 parameters:

- **id** - unique string which identifies particular text object.
  - **id** format is following cust<number>_txt
  - number should be from 1 to 30
  - examples: cust1_txt, cust21_txt
  - **id** will be linked with **text_id** field in elements.yml file:
    text_id of the particular element in elements.yml will be the same as **id** in texts.yml
    So particular element will display that text string.

- **value** - text itself. NOTE: brackets “” must be used

Additional Customization of Graphical Interface

Custom styles and fonts can be applied if needed.
E.g shadows, font attributes, etc.

How to create custom styles?

1. Create styles.yml
2. Specify all styles
3. Apply styles to the graphical elements which are specified in elements.yml
4. Upload to the Tablet styles.yml and elements.yml files
5. Reboot Tablet

styles.yml file example

```
---
Styles:
# Dark grey label style
-
    id: dGreySpaceLab_stl
    text_color: dark_grey
    text_align: center
    letter_space: 1

# Generic button style
-
    id: genericBtn_stl
    bg_color: white
    radius: 5
    pad_hor: 0
    pad_ver: 0
    pad_gap: 0
    border_color: dark_grey
    border_width: 10
    border_opa: opa_cover
    border_side: full
    shadow_color: dark_grey
    shadow_width: 5
```

This example file includes configurations for 2 styles (with names *dGreySpaceLab_stl* and *genericBtn_stl* that are set in *id*).

Mandatory Field for styles

- **id** – unique identifier of the style. It should be literal name that may consist of Latin letters, "_" and digits (first symbol has to be a letter) and its length should not exceed 20 symbols. It is desirable to add "_stl" at the end of the *id* of the style (do not forget to include these 4 symbols in *id* length limit).

Optional Fields for styles

- **text_color** – a color of the text of label or label in button. Default value is *white*. Possible values of **text_color** field are (other value will be ignored by Tablet):
  - *white*
  - *light_grey*
  - *dark_grey*
  - *black*
- **text_align** – the alignment of the text in label. Text alignment is used by label only if the width of the label is more than the width of label content text (see width property for Default value is *auto*. Possible values of **text_align** field are (other value will be ignored by Tablet):
  - *auto* – text align according to symbols that it contents (for example left alignment for Latin and right alignment for Arabic).
  - *left*
  - *center*
  - *right*
- **letter_space** – the space in pixels between the nearest letters in the same word. The range of this field is 0 – 10 (0 means that this option is disabled). Default value is 0 pixels.
- **bg_color** – a color of the background of the screen element. Possible values of **bg_color** field are the same as for **text_color**. Default value is *white*.
- **border_color** – a color of the borders of the screen element. Possible values of **border_color** field are the same as for **text_color**. Default value is *black*.
- **border_opa** – the opacity of the borders of the screen element. Default value is *opa_cover*. Possible values of **border_opa** field are (other value will be ignored by Tablet):
  - *opa_transp* – transparent border
  - *opa_20* – 20% opacity
  - *opa_40* – 40% opacity
  - *opa_60* – 60% opacity
  - *opa_80* – 80% opacity
  - *opa_100* – 100% opacity
  - *opa_cover* – full opacity (the same as *opa_100*)
- **border_side** – the side of the screen element where border should be placed. Default value is *none*. Possible values of **border_side** field are (other value will be ignored by Tablet):
  - *none*
  - *bottom*
  - *top*
  - *left*

    - *right*
    - *full*
- **border_width** – the width of the border in pixels. The range of this field is 0 – 50 (0 means that this option is disabled). Default value is 0.
- **shadow_color** – a color of the shadows of the screen element. Possible values of **shadow_color** field are the same as for **text_color**. Default value is *white*.
- **shadow_width** – the width of the shadow in pixels. The range of this field is 0 – 50 (0 means that this option is disabled). Default value is 0.
- **pad_hor** – horizontal padding in pixels. The range of this field is 0 – 20 (0 means that this option is disabled). Default value is 0.
- **pad_ver** – vertical padding in pixels. The range of this field is 0 – 20 (0 means that this option is disabled). Default value is 0.
- **pad_gap** – gap padding in pixels. The range of this field is 0 – 20 (0 means that this option is disabled). Default value is 0.
- **radius** – the radius of every corner in pixels. The range of this field is 0 – 20 (0 means that this option is disabled). Default value is 0.

NOTE: The number of configured styles in styles.yml file should not exceed 50.

How to create custom fonts?

1. Convert font from TTF/WOFF to bin with Font Converter
2. Link new font bin-file with elements you want to use this font in elements.yml
3. Upload new font bin-file and new elements.yml file to the Tablet
4. Reboot Tablet

Example

Fonts may be set up in elements.yml file:

```
# Product Name label
-
    id: lab_productName
    type: label
    style_id: blackLab_stl
    font: chivo40.bin
    x: 0
    y: 135
    width: 1024
    visible: 1
    text_id: meetingRoomDisplayTablet_txt
```

This example configuration for Product Name label (with *lab_productName* as the value of *id* field) shows that this label will use chivo40.bin font file (because the value of *font* field is equal to *chivo40.bin*). If chivo40.bin was not uploaded to the Tablet or the size of this file is more then permitted (total size of uploaded to the Tablet images, fonts and touchscreen firmware files should not exceed 220 kB) than the Tablet will use default font instead of chivo40.bin.

How to prepare Images?

1. Convert image from PNG/JPG/BMP to bin with Image Converter
2. Link new image bin-file with elements you want to use this image in elements.yml
3. Upload new image bin-file and new elements.yml file to the Tablet

4. Reboot Tablet

Example

Images may be set up in elements.yml file:

```
#TEK logo black image
-
    id: img_logoBlack
    type: image
    x: 650
    y: 20
    image: LOGO_B.bin
    visible: 1
```

This example configuration for TEK logo black image (with *img_logoBlack* as the value of *id* field) shows that this image will use LOGO_B.bin file (because the value of *image* field is equal to *LOGO_B.bin*) as source of picture. If LOGO_B.bin was not uploaded to the Tablet or the size of this file is more then permitted (total size of uploaded to the Tablet images, fonts and touchscreen firmware files should not exceed 220 kB) than the Tablet will not show this picture.

Image Converter

Online image converter is a free tool that is accessible through the link: https://lvgl.io/tools/imageconverter

It is possible to create binary files from images (.png, .jpg and .bmp).

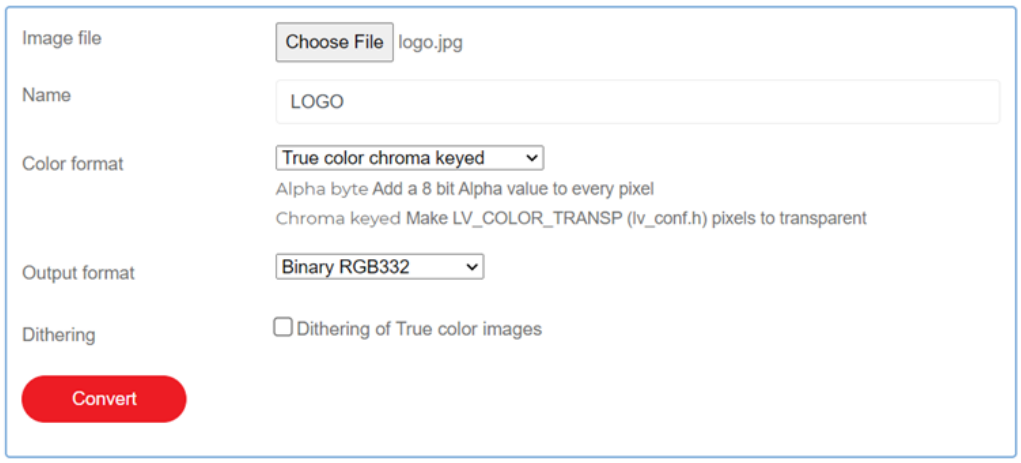

Figure 1: Online Image Converter

To convert a picture to binary format it is necessary to:

1. Choose an Image (png, jpg, or bmp) (see Figure **1**).
2. Give a Name to the output file. E.g. "LOGO"
3. Specify the Color format. It is recommended to use “True color chroma keyed” format. Some format may convert your picture wrong.
4. Set the output format to Binary RGB332, Binary RGB565 or Binary RGB888. Do not use “C array”. It is recommended to use “RGB332” format to decrease image file size.
5. Click the Convert button and to download the converted image.

Short explanations about image converter and about color formats can be find on the same page.

Font Converter

Offline font converter is a free tool that is require node.js 10+ (node.js may be downloaded from Download | Node.js).

To install globally Font Converter (“lv_font_conv”) use your command prompt and these commands:

```
# install release from npm registry
npm i lv_font_conv -g
# install from github's repo, master branch
npm i lvgl/lv_font_conv -g
```

It is possible to use lv_font_conv command in command prompt after that.

The command with -h key shows help message and the meanings of the keys that is possible to use with lv_font_conv:

```
> lv_font_conv -h
usage: lv_font_conv.js [-h] [-v] --size PIXELS [-o <path>] --bpp {1,2,3,4,8} [--lcd | --lcd-v] [--use-color-info]
                       --format {dump,bin,lvgl} --font <path> [-r RANGE] [--symbols SYMBOLS] [--autohint-off]
                       [--autohint-strong] [--force-fast-kern-format] [--no-compress] [--no-prefilter] [--no-kerning]
                       [--lv-include <path>] [--full-info]

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         show program's version number and exit
  --size PIXELS         Output font size, pixels.
  -o <path>, --output <path> Output path.
  --bpp {1,2,3,4,8}     Bits per pixel, for antialiasing.
  --lcd                 Enable subpixel rendering (horizontal pixel layout).
  --lcd-v               Enable subpixel rendering (vertical pixel layout).
  --use-color-info      Try to use glyph color info from font to create grayscale icons. Since gray tones are emulated via transparency, result will be good on contrast background only.
  --format {dump,bin,lvgl}          Output format.
  --font <path>         Source font path. Can be used multiple times to merge glyphs from different fonts.
  -r RANGE, --range RANGE
                        Range of glyphs to copy. Can be used multiple times, belongs to previously declared "--font".
                        Examples:
                        -r 0x1F450
                        -r 0x20-0x7F
                        -r 32-127
                        -r 32-127,0x1F450
                        -r '0x1F450=>0xF005'
                        -r '0x1F450-0x1F470=>0xF005'
  --symbols SYMBOLS     List of characters to copy, belongs to previously declared "--font". Examples:
                        --symbols ,.0123456789
                        --symbols abcdefghigklmnopqrstuvwxyz
  --autohint-off        Disable autohinting for previously declared "--font"
  --autohint-strong     Use more strong autohinting for previously declared "--font" (will break kerning)
```

```
  --force-fast-kern-format
                        Always use kern classes instead of pairs (might be larger but faster).
  --no-compress         Disable built-in RLE compression.
  --no-prefilter        Disable bitmap lines filter (XOR), used to improve compression ratio.
  --no-kerning          Drop kerning info to reduce size (not recommended).
  --lv-include <path>   Set alternate "lvgl.h" path (for --format lvgl).
  --full-info           Don't shorten "font_info.json" (include pixels data).
```

For example, if it is necessary to create binary bold_font_30.bin font file with Latin symbols from Chivo-Bold.ttf file and with Arabic symbols from DejaVuSans-Bold.ttf file with font height equal to 30 pixels than you may use this command:

```
> lv_font_conv --no-compress --no-prefilter --bpp 1 --size 30 --font Chivo-Bold.ttf -r 0x20-0x7F --font
DejaVuSans-Bold.ttf -r 0x600-0x6FF -r 0xFB50-0xFDFF -r 0xFE70-0xFEFF --format bin -o bold_font_30.bin --
force-fast-kern-format
```

More examples and details about Offline Font Converter see at: GitHub - lvgl/lv_font_conv: Converts TTF/WOFF fonts to compact bitmap format .

LoRaWAN Interface customization

How to switch Screen with Downlink?

1. Add to dlinks.yml file new downlink with defined *screen_id* field
2. Add to uplinks.yml file new uplink (if it is not exist) which will be send on periodic basis to flush downlink queue delivery from Network Server
3. Upload these files to the Tablet
4. Reboot Tablet
5. Send downlink that was added in step 1

Downlinks may be set up in dlinks.yml file:

```yaml
#Switch to Custom Screen 1 (0x50)
-
    header_id: 0x50
    port: 110
    screen_id: custom1_scr
```

And uplinks may be set up in uplinks.yml file:

```yaml
# Periodic Uplink to Manage Downlink Queue (0x33)
-
    header_id: 51
    port: 10
    size: 1
    source: timer
    period: 2
```

It is necessary to send downlink 0x50 to port 110 to switch screen to Custom Screen 1 in this example. 0x33 periodical uplink will be used here to flush downlink queue from Network Server.

How to send and display new data to tablet?

1. Add to dlinks.yml file new downlink with *updateElement_hdl* or *updateSeveralElements_hdl* handler
2. Add to uplinks.yml file with the uplink which will be send on periodic basis to trigger downlink delivery from Network Server
3. Upload these file to the Tablet
4. Reboot Tablet
5. Send downlink with new text

Note: New text in downlink should be in Unicode. Make sure that the label with new text was linked with font that includes all symbols to show this text.

Downlinks may be set up in dlinks.yml file:

```
# Update element (0x45)
-
    header_id: 0x45
    port: 102
    handler_id: updateElement_hdl
```

And uplinks may be set up in uplinks.yml file:

```
# Periodic Uplink to Manage Downlink Queue (0x33)
-
    header_id: 51
    port: 10
    size: 1
    source: timer
    period: 2
```

This example configuration for Update element downlink (with *0x45* as the value of *header_id* field) shows that this downlink payload will be interpreted by the Tablet as command to update screen element (change text in label (for example the value of CO_2 level) or visibility of screen element).

This downlink will be handled as command to update element only if it is sent to port 102 (because *102* is the value of *port* field) and with ACK bit equal to 1. ACK bit is the highest in header. In this example downlink should be with header 0xC5 (because 0x45 in hex is 0b**0**1000101 in binary and it will be changed to 0xC5 which is 0b**1**1000101 with 1 in highest bit) to be interpreted by the Tablet as command to update screen element.

Example:

You have 3 labels on Custom Screen 1:

1. Label with static text “Temperature”.
2. Label with temperature value.
3. Label with static text “°C“.

These labels defined in elements.yml file:

```
# Temp label
-
    id: lab_custScr1_2
    type: label
    style_id: blackLab_stl
    font: chivo40.bin
    x: 50
    y: 340
```

```
    visible: 1
    text_id: cust2_txt

# Temp value label
-
    id: lab_custScr1_3
    type: label
    style_id: blackLab_stl
    font: chivo40.bin
    x: 345
    y: 340
    visible: 1
    text_id: cust3_txt

# Degree label
-
    id: lab_custScr1_4
    type: label
    style_id: blackLab_stl
    font: chivo40.bin
    x: 400
    y: 340
    visible: 1
    text_id: cust4_txt
```

Static texts for them are defined in texts.yml file:

```
# Temperature
-
    id: cust2_txt
    value: Temperature

# Temperature value
-
    id: cust3_txt
    value: "24"

# Celsius degree
-
    id: cust4_txt
    value: "°C"
```

And you want to update temperature value to 47. To do that it is necessary to send such downlink on port 102 to Memo (it is necessary to add aforementioned downlink to dlinks.yml and uplink to uplinks.yml):

0xC5 01 3B 02 34 37

The following the DL frame formats for update screen element message (0x45 downlink in our case).

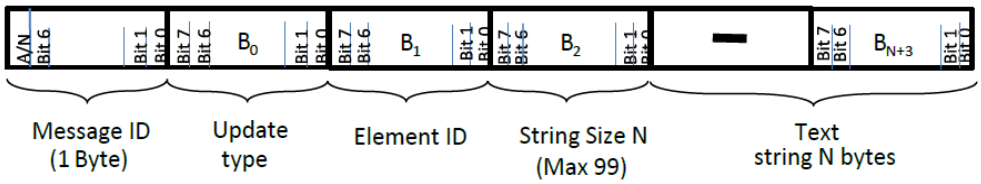

Figure 2: The format of a DL update screen element with text update message block

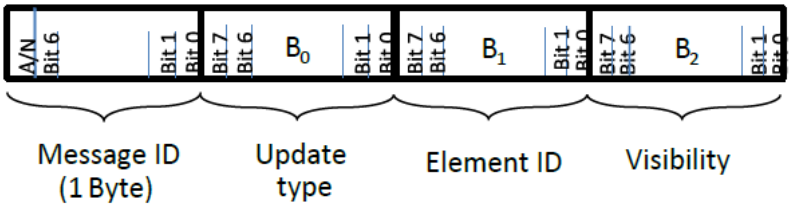

Figure 3: The format of a DL update screen element with visibility update message block.

Message ID bit 7 (A/N) of the message ID determines whether message is ack or nacked. Nacked message is ignored.

B_0 is determine screen element update type. Possible values for B_0 are: 1 – label text update, 2 - screen element visibility update (message with any other value in B_0 is ignoring by tablet).

B_1 is determine screen element ID number (identifier of the element of the screen that necessary to update). Possible values for B_1 are corresponding with *id* field in elements.yml screen element configuration file: 0 is for *lab_productName*, 1 – *lab_fwVer*, 2 – *lab_bootloadVer*, 3 – *lab_apploadVer*, 4 – *lab_iteFwVer*, 5 – *lab_tsFwVersion*, 6 – *lab_loraStatus*, 7 – *img_logoBlack*, 8 – *pnl_bottom*, 57 – *lab_custScr1_1*, 58 – *lab_custScr1_2*, 59 – *lab_custScr1_3*, 60 – *lab_custScr1_4*, 61 – *lab_custScr1_5*, 62 – *lab_custScr1_6*, 63 – *lab_custScr1_7*, 64 – *lab_custScr1_8*, 65 – *lab_custScr1_9*, 66 – *lab_custScr1_10*, 67 – *but_custScr1_1*, 68 – *but_custScr1_2*, 69 – *but_custScr1_3*, 70 – *but_custScr1_4*, 71 – *but_custScr1_5*, 72 – *but_custScr1_6*, 73 – *pnl_custScr1_1*, 74 – *pnl_custScr1_2*, 75 – *pnl_custScr1_3*, 76 – *pnl_custScr1_4*, 77 – *pnl_custScr1_5*, 78 – *img_custScr1_1*, 79 – *img_custScr1_2*, 80 – *img_custScr1_3*, 81 – *img_custScr1_4*, 82 – *img_custScr1_5*, 83 – *lab_custScr2_1*, 84 – *lab_custScr2_2*, 85 – *lab_custScr2_3*, 86 – *lab_custScr2_4*, 87 – *lab_custScr2_5*, 88 – *lab_custScr2_6*, 89 – *lab_custScr2_7*, 90 – *lab_custScr2_8*, 91 – *lab_custScr2_9*, 92 – *lab_custScr2_10*, 93 – *but_custScr2_1*, 94 – *but_custScr2_2*, 95 – *but_custScr2_3*, 96 – *but_custScr2_4*, 97 – *but_custScr2_5*, 98 – *but_custScr2_6*, 99 – *pnl_custScr2_1*, 100 – *pnl_custScr2_2*, 101 – *pnl_custScr2_3*, 102 – *pnl_custScr2_4*, 103 – *pnl_custScr2_5*, 104 – *img_custScr2_1*, 105 – *img_custScr2_2*, 106 – *img_custScr2_3*, 107 – *img_custScr2_4*, 108 – *img_custScr2_5*, 109 – *lab_custScr3_1*, 110 – *lab_custScr3_2*, 111 – *lab_custScr3_3*, 112 – *lab_custScr3_4*, 113 – *lab_custScr3_5*, 114 – *lab_custScr3_6*, 115 – *lab_custScr3_7*, 116 – *lab_custScr3_8*, 117 – *lab_custScr3_9*, 118 – *lab_custScr3_10*, 119 – *but_custScr3_1*, 120 – *but_custScr3_2*, 121 – *but_custScr3_3*, 122 – *but_custScr3_4*, 123 – *but_custScr3_5*, 124 – *but_custScr3_6*, 125 – *pnl_custScr3_1*, 126 – *pnl_custScr3_2*, 127 – *pnl_custScr3_3*, 128 – *pnl_custScr3_4*, 129 – *pnl_custScr3_5*, 130 – *img_custScr3_1*, 131 – *img_custScr3_2*, 132 – *img_custScr3_3*, 133 – *img_custScr3_4*, 134 – *img_custScr3_5*, 135 – *lab_custScr4_1*, 136 – *lab_custScr4_2*, 137 – *lab_custScr4_3*, 138 – *lab_custScr4_4*, 139 – *lab_custScr4_5*, 140 – *lab_custScr4_6*, 141 – *lab_custScr4_7*, 142 – *lab_custScr4_8*, 143 – *lab_custScr4_9*, 144 – *lab_custScr4_10*, 145 – *but_custScr4_1*, 146 – *but_custScr4_2*, 147 – *but_custScr4_3*, 148 – *but_custScr4_4*, 149 – *but_custScr4_5*, 150 – *but_custScr4_6*, 151 – *pnl_custScr4_1*, 152 – *pnl_custScr4_2*, 153 – *pnl_custScr4_3*, 154 – *pnl_custScr4_4*, 155 – *pnl_custScr4_5*, 156 – *img_custScr4_1*, 157 – *img_custScr4_2*, 158 – *img_custScr4_3*, 159 – *img_custScr4_4*, 160 – *img_custScr4_5* (update screen element message with any other value for B_1 is ignoring by tablet).

Bytes starting from B_2 are specific for different update types.

In update label text message (B_0 is set to 1) Byte 2 (B_2) is size of the new text string N bytes for the label (Figure **2**). The size (B_2) is permitted to have values:

- 0x00 – label change its value with space symbol.
- 0x01 – 0x63 – size of the text string for label (maximum size is 99 bytes; if value is more then 99 (0x63 in Heximal) and not 0xFF tablet ignore string after 99th byte).
- 0xFF – label text reset to default value that set in elements.yml configuration file.

Subsequent bytes in update label text message are new text string of N bytes for label (Fig. **2**).

In update screen element visibility message (B_0 is set to 2) Byte 2 (B_2) is new visibility state (any other values are ignored by tablet):

- 0x00 – screen element switches to invisible state.
- 0x01 – screen element switches to visible state.

The following table describes the update screen element downlink acknowledgment.

Table: DL Frame Update screen element acknowledgement.

Information Type	**Example Message ID Field Value**	**Maximum DL Frame size (Bytes)**	**Information**
Update screen element message (example *header_id* is 0x45)	0xC5	103	ACK
	0x45	1	NACK
Update several screen elements message (example *header_id* is 0x47)	0xC7	Depends on DR	ACK
	0x47	1	NACK

Example update screen element messages:

- 0x C5 01 3B 09 54 65 73 74 20 74 65 78 74 – message ID with ACK bit (this message is mandatory for tablet); update type B_0 is 0x01 (update label text); element ID B_1 is 0x3B (0x3B that is 59 in Decimal is *lab_custScr1_3* label); string size B_2 is 0x09 (0x09 is 9 in Decimal that means that new text string is consist of 9 symbols); string B_3 – B_{11} are ASCII codes for string “Test text”.
- 0x C5 02 3B 00 – message ID with ACK bit (this message is mandatory for tablet); update type B_0 is 0x02 (update screen element visibility); element ID B_1 is 0x3B (0x3B that is 59 in Decimal is *lab_custScr1_3* label); visibility state B_2 is 0x00 (set screen element invisible).
- 0x C5 01 3B FF – message ID with ACK bit (this message is mandatory for tablet); update type B_0 is 0x01 (update label text); element ID B_1 is 0x3B (0x3B that is 59 in Decimal is *lab_custScr1_3* label); string size B_2 is 0xFF (set default text that is determined in elements.yml configuration file for label).
- 0x C7 02 01 3B 09 54 65 73 74 20 74 65 78 74 02 3A 00 – message ID with ACK bit (this message is mandatory for tablet); number of elements B_0 is 2 (it is necessary to update 2 elements); update type of the first screen element B_1 is 0x01 (update label text); element ID B_2 is 0x3B (0x3B that is 59 in Decimal is *lab_custScr1_3* label); string size B_3 is 0x09 (0x09 is 9 in Decimal that means that new text string is consist of 9 symbols); string B_4 – B_{12} are ASCII codes for string “Test text”; update type of the second screen element B_{13} is 0x02 (update screen element visibility); element ID B_{14} is 0x3A (0x3A that is 58 in Decimal is *lab_custScr1_2* label); visibility state B_{15} is 0x00 (set screen element invisible).

Mandatory Field for downlinks

- **header_id** – 1 byte message ID. It is allowed to set the value of this field in Decimal (should be set as usual number) and in Heximal (should be set with 0x or 0X prefix).
- **port** – LoRa port number.

Optional Field for downlinks

- **handler_id** – identifier of the handler that processes downlink payload. Possible values are:
  - *updateElement_hdl* – handler for Update element downlink.
  - *updateSeveralElements_hdl* – handler for Update several elements downlink.
  - *eodSleep_hdl* – handler for End-of-day sleep downlink.
- **screen_id** – identifier of the screen that tablet should switch to after receiving configured downlink. Possible values are:
  - *init_scr* – initialization screen that consist of 6 labels, 1 panel and 1 image. This screen is designed to view base Custom Display Tablet version information.
  - *custom1_scr*, *custom2_scr*, *custom3_scr*, *custom4_scr* – screens that is designed to have possibility to configure the custom screen. They consist of 10 labels, 6 buttons, 5 panels and 5 images on each screen.

  NOTE: All custom screens do not have cross-screen connections between labels, panels and images.
- **ack_screen_id** – identifier of the screen that tablet should switch to after receiving acked configured downlink. This field is ignored if **screen_id** is present in downlink configurations. This field is used as **screen_id** configuration if **nack_screen_id** and **screen_id** fields are absent in downlink configurations. Possible values are the same as for **screen_id**.
- **nack_screen_id** – identifier of the screen that tablet should switch to after receiving nacked configured downlink. This field is ignored if **screen_id** is present in downlink configurations. This field is used as **screen_id** configuration if **ack_screen_id** and **screen_id** fields are absent in downlink configurations. Possible values are the same as for **screen_id**.

Downlink configuration file example is:

```
---
Downlinks:

# Get Main Screen response (0x58)
-
    header_id: 0x58
    port: 103
    screen_id: custom1_scr

# Get Temperature response (0x53)
-
    header_id: 83
    port: 102
    handler_id: updateElement_hdl
    ack_screen_id: custom2_scr
    nack_screen_id: custom3_scr

# Deep Sleep Management response (0x59)
-
    header_id: 89
    port: 104
    handler_id: eodSleep_hdl
```

This example file includes configurations for 3 downlinks (with message IDs *0x58*, *83* (that is 0x53) and *89* (that is 0x59) that are set in *header_id*).

All these downlinks are headed with commentary started with # sign. These commentaries give short brief about the downlink that is configured below and they are optional.

Get Main Screen response downlink (with *header_id 0x58*) is the first downlink in this configuration file. It is configured to port *103*. This mean that optional configurations (*screen_id* in this case) is used only if tablet received downlink on port that is set in *port* field and with

message ID that is specified in field *header_id* (*port 103* and *header_id 0x58* in this case). There is *screen_id* with *custom1_scr* as its value in configurations of this downlink. This mean that tablet switches its view to first custom screen after receiving of this downlink.

Get Temperature response is configured as downlink with message ID *83* (or 0x53 in Heximal) that is set in *header_id* field and is waiting on port *102* that is set in *port* field. The payload of this downlink is processes as Update Element downlink that is set in *handler_id* by value *updateElement_hdl*. The view of the tablet switches to the second custom screen after receiving this acked downlink (that is set in field *ack_screen_id* by value *custom2_scr*) and to the third custom screen after receiving this nacked downlink (that is set in field *nack_screen_id* by value *custom3_scr*).

Deep Sleep Management response is configured as downlink with message ID *89* (or 0x59 in Heximal) that is set in *header_id* field and is waiting on port *104* that is set in *port* field. The payload of this downlink is processes as End-of-day sleep downlink that is set in *handler_id* by value *eodSleep_hdl*. The view of the tablet does not switch to any screen because there no *screen_id*, *ack_screen_id* or *nack_screen_id* in configurations for this downlink.

How to send button press to NS?

1. Add to uplinks.yml file new uplink with *element* as value of *source* field
2. Set for this uplink up id of button (from elements.yml file) as value of *element_id* field
3. Upload this file to the Tablet
4. Reboot Tablet
5. Wait till the Tablet connect to LoRa network
6. Switch the screen to one where the button was placed (with downlink)
7. Tap the button

Example:

You have a button on Custom Screen 1. It is defined in elements.yml file:

```
# Request cleaning button
-
    id: but_custScr1_1
    type: button
    style_id: genericBtn_stl
    label_style: whiteLab_stl
    font: chivo24.bin
    screen_id: custom2_scr
    x: 725
    y: 510
    width: 240
    height: 100
    text_id: cust8_txt
    visible: 1
```

You want Memo to send uplink 0x34 without payload after tapping this button. To do that it is necessary to add uplink to uplinks.yml and this button as source of this uplink:

```
# Request cleaning button event (0x34)
-
   header_id: 0x34
   port: 10
   size: 1
   source: element
   element_id: but_custScr1_1
```

```
    retry: 10
    response_dl: 0x50
```

Mandatory Field for uplinks

- **header_id** – 1 byte message ID. It is allowed to set the value of this field in Decimal (should be set as usual number) and in Heximal (should be set with 0x or 0X prefix). It is not possible to set **header_id** equal to values that are message IDs for predefined uplinks (from 0x00 to 0x13, from 0x24 to 0x27, from 0x40 to 0x42, 0x70 and 0x72): tablet ignore uplink with such predefined message ID.
- **port** – LoRa port number. It is not possible to set **port** 20 (if **port** is set to 20 then uplink with such configuration will be ignored by tablet).
- **size** – payload size (with message ID). If **size** is set to 0, tablet will increase it to 1 (message ID size) internally.
- **source** – "something" that cause sending the uplink. Possible values (any other value will be ignored) for this field are:
  - *timer* – uplink will be sent periodically. If **source** is set as *timer*, *period* is also mandatory field.
  - *element* – uplink will be sent after pressing certain screen button. If **source** is set as *element*, **element_id** is also mandatory field.
  - *event* – uplink will be sent after some event that can happen during tablet runtime. If **source** is set as *element*, **event_id** is also mandatory field.

Optional Field for uplinks

- **period** – number of core ticks between two periodical uplinks. This field will be ignored if **source** is not set as *timer*.
- **element_id** – identifier of the button that cause sending the uplink. All button identifiers could be found in elements.yml file that configurate screen elements (any other value will be ignored). This field will be ignored if **source** is not set as *element*.
- **event_id** – identifier of the event that cause sending the uplink. Possible values (any other value will be ignored) for this field are:
  - *join_evt* – uplink will send right after tablet join to LoRa network.
- **retry** – this field is show if uplink should be resent in case of absence of response downlink to it (if **retry** value set equal to 1, uplink should be resent infinite number of times if there were no response to it; if **retry** value is more then 1 and less then 255, uplink should be resent defined number of times if there were no response to it; if **retry** value set equal to 0 or this field is absent in uplink configurations, uplink do not require response). If **retry** value is set to value that is more then 0, repeat uplink will be sent every 30 sec in case of absence of response downlink (dummy uplinks will also be emitted by tablet every 10 sec in this case). This field will be ignored if **response_dl** field value is set equal to 0 or if there are no **response_dl** in uplink configurations.
- **response_dl** – message ID of response downlink. The downlink with such message ID (any downlink with such message ID) stops uplink retry cycle and marks uplink request as responded. This field will be ignored if **retry** field value is set equal to 0 or if there are no **retry** in uplink configurations. It is allowed to set the value of this field in Decimal (should be set as usual number) and in Heximal (should be set with 0x or 0X prefix).
- **payload** – 1 byte payload value. It is allowed to set the value of this field only in Heximal (should be set with 0x or 0X prefix). This field is proceeding only if **size** is value is not less than 2. It is possible to use different payloads for uplinks with the same header.

NOTE: The number of configured uplinks in uplinks.yml file should not exceed 254.

Uplink configuration file example is:

```
---
Uplinks:

# Get Main Screen request (0x58)
-
```

```
    header_id: 0x58
    port: 10
    size: 1
    source: event
    event_id: join_evt
    retry: 1
    response_dl: 0x58

# Get Temperature request (0x53)
-
    header_id: 83
    port: 10
    size: 1
    source: timer
    period: 2
    retry: 1
    response_dl: 0x53

# Switch Screen request (0x54)
-
    header_id: 0X54
    port: 10
    size: 2
    source: element
    element_id: btn_custScr2_3
    payload: 0x05
    retry: 1
    response_dl: 0x54
```

This example file includes configurations for 3 uplinks (with message IDs *0x58*, *83* (that is 0x53) and *0x54* that are set in *header_id*).

All these uplinks are headed with commentary started with # sign (string after # is interpreted as commentary in YAML). These commentaries give short brief about the uplink that is configured below and they are optional.

All these uplinks are configured to be sent on port *10*.

Get Main Screen request with *header_id 0x58* and Get Temperature request with *header_id 83* have *size 1* that means that these requests are consist of 1 byte message ID only. Switch Screen request with *header_id 0X54* has *size* equal to *2* that means that this uplink consists of 1 byte message ID and 1 byte payload. This payload is defined by *payload* (*0x05* in this case).

All these uplinks configured to be resendable (*retry* field value is *1*) till getting the response (response downlink message ID is defined with *response_dl*).

Get Main Screen request has *event* as its *source*. That means that Get Main Screen request uplink is emitted by tablet after special event in tablet. This event is defined in *event_id* field (*join_evt* in this case that mean that this request is emitted by tablet after successful joining to LoRa network).

Get Temperature request has *timer* as its *source*. That means that Get Temperature request uplink is emitted periodical by tablet every *2* core ticks that is set in *period*.

Switch Screen request has *element* as *source*. That means that Switch Screen request uplink is emitted by tablet after pressing the screen button that is set in *element_id* field (*btn_custScr2_3* in this case).