Application Note

AN_306

FT800 Jackpot Application

Version 1.1

Issue Date: 2018-01-04

This application note describes the operation of the Jackpot Demo Application running on the FT9XX MCU, Microsoft Visual C, Arduino and EVE Screen Editor platforms. The objective of the Demo Application is to enable users to become familiar with the usage of the FT8XX, the design flow, and display list used to create the desired user interface or visual effect.

Bridgetek Pte Ltd (BRTChip)
178 Paya Lebar Road, #07-03, Singapore 409030
Tel: +65 6547 4827 Fax: +65 6841 6071
Web Site: http://www.brtchip.com

Table of Contents

1 Introduction

This application is a custom version of a Slot Machine game and it demonstrates the usage of built-in FT8xx widgets and primitives. The steps used to construct the User Interface (UI) and the algorithms used are also discussed in this document.

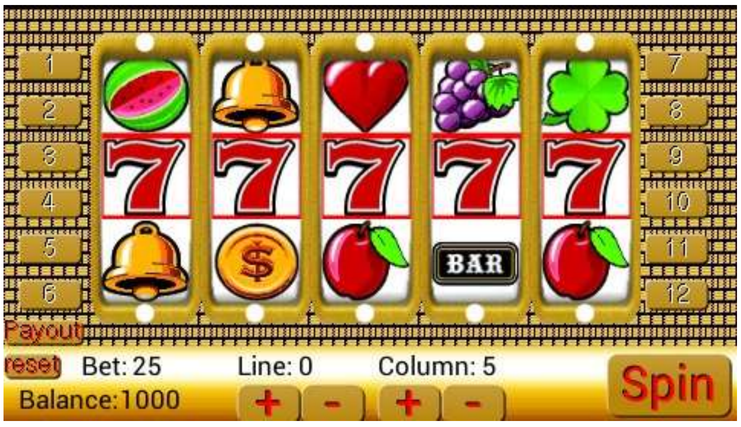

Figure 1-1 Slot Machine

Further sample code for the EVE family can be found on the EVE examples page: http://brtchip.com/SoftwareExamples-eve/.

1.1 Overview

This document presents a basic understanding of the FT8xx built-in feature sets.

The example code can be run in a variety of platforms as listed below but could be ported over to other platforms:

- PC running Visual Studio (C++) with FTDI USB-SPI interface (C232HM cable, VA800A-SPI adapter, VM800BU basic board with USB interface)
- FT900 MCU platform
- EVE Screen Editor
- Arduino

1.2 Scope

This document should be used by software programmers to develop GUI applications by using the FT8XX with any MCU via SPI.

Please refer to the sample code package provided at the following link:
http://brtchip.com/SoftwareExamples-eve/

Note: This document is intended to be used along with the source code project provided. This can be found at the link in above.

2 Application Flow

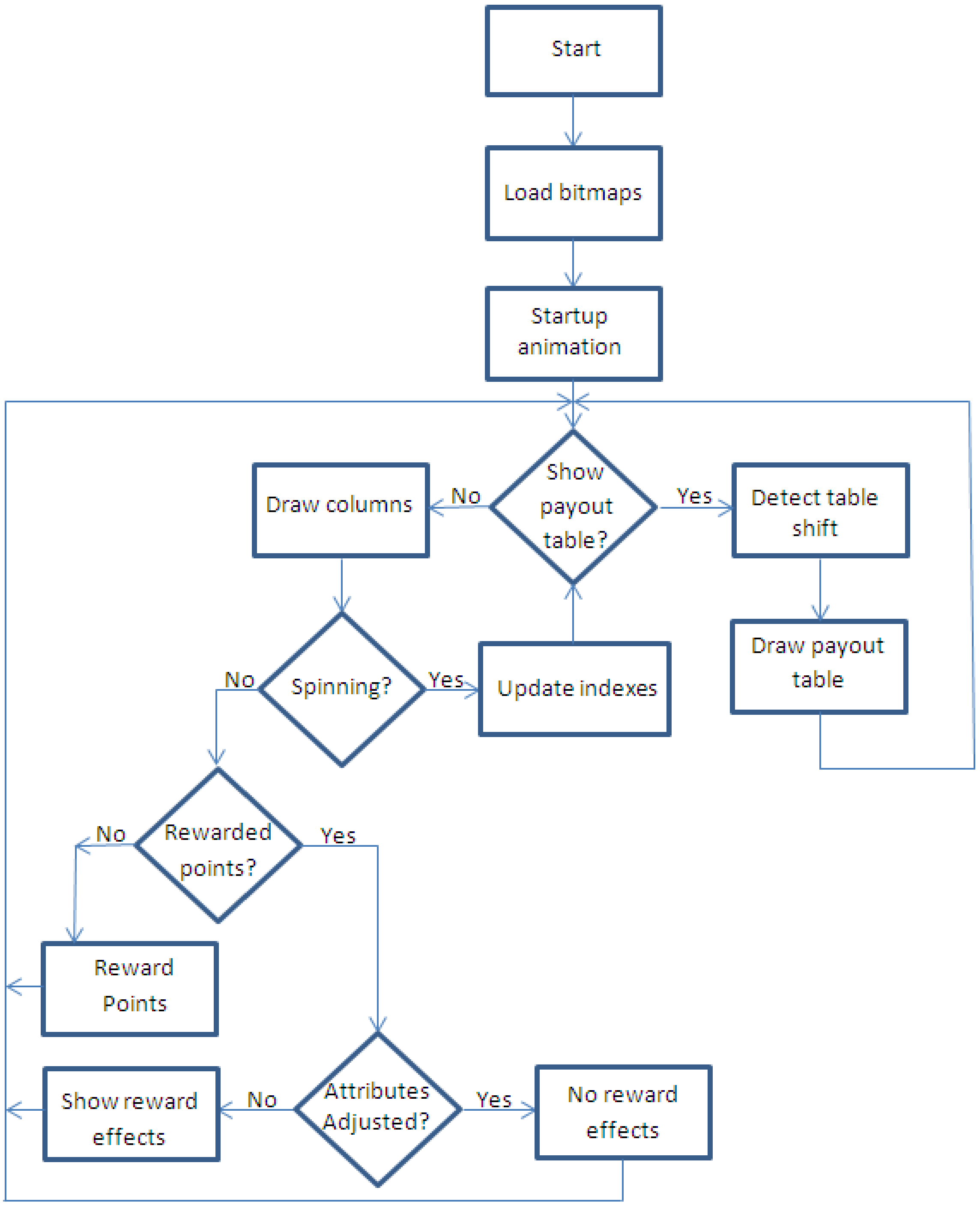

Figure 2-1 Application Flowchart

3 Description

Refer to AN 391 EVE Platform Guide for information pertaining to platform setup and the necessary development environment.

3.1 Initialization

3.1.1 Global Arrays

A series of arrays were used for the UI construction.

- spinning_column_t - Holds the information about each of the spinning columns.
    - curIndex – This variable holds the index in the iconArray which is the first icon to be displayed for the column.
    - velocity – The purpose of this variable is to randomize the icons. After the spin button has been released, velocity decreases at a fixed rate in each update loop. The column stops when the velocity becomes 0.
    - iconArray – This array holds bitmap handles in a random order.
- bet_line_t- This predefined table holds the coordinates for each bet line and the line color.
- payout table - A simple array which the bitmap handle corresponds to the index of the array and the value of the index is the payout amount.
- bitmap_header_t - It holds the bitmap attributes such as: width, height, stride and location in the memory.

3.1.2 Load Bitmap

This application uses bitmaps extensively. All bitmaps used in this application are in RGB565 format except the spinning column overlay which is in L8 format. Bitmaps are loaded into the ram before the game loop.

This application also demonstrates the usage of various types of supported bitmap formats.

- .jpg - Many photo-editing software packages support this format so quick modifications are easy, but this format doesn't support transparency. The user can convert Jpeg images to one of the supported bitmap formats that retain transparency such as ARGB4. An image conversion tool is available at: http://brtchip.com/utilities/#evescreenconverters and more information regarding supported bitmap formats can be found in the FT8XX Series Programming Guide.
- .bin - This binary format is compressed by ZLIB algorithm.
- .raw - This binary format can be downloaded into FT8XX graphics memory directly without decoding and decompressing. Large bitmaps should be loaded into the ram directly to prevent hanging during the loading stage.

Note 1: Bitmaps with the same dimensions and format can be edited into a long vertical bitmap strip. The single bitmap would only takes up one bitmap handle and the individual bitmap can be accessed by the CELL parameter.

Note 2: When all the user definable bitmap handles have been exhausted, the bitmap drawing method of specifying the source, layout, and size is required.

3.2 Functionality

This application starts up with an animation of spinning columns and ends up on the highest reward combination of symbols. Other than the startup animation the status bar is fixed at the lower portion of the screen. It is constructed by repeating a single bitmap line across the screen width and buttons and status text are then drawn on top of it.

This demo application has two playing modes. One mode is the version where the middle icons of the selected columns are the only ones being considered for rewards. Touch of the + or – button under the *Column* field will increase or decrease the number of selected columns respectively. The other mode is betting on the predefined lines. Touching one of the numbered buttons on either side of the screen will automatically include all the other lower bet lines. Similarly, touch of the + or – button under the *Line* field will also increase or decrease the bet lines respectively. The winning combination is rewarded according to the payout table that appears when the *payout* button on the lower left side of the screen is pressed.

One of the main screens is the *spinning-columns* screen. This screen displays the spinning columns and it is the default screen for this demo application. The steps to construct the screen are as follows:

```
{
draw coin background;
update the current_icon index for each column and decrease velocity;
for(i=0;i<number of spin columns;i++){
        if(column velocity == 0)
                this column's drawn index = current_icon index + 1;
        for(j=0;j<3;j++){
                draw the index of (current_icon index + j);
                draw L8 formatted overlay bitmap;
                draw outer overlay;
                highlight selected spinning column with two dots;
                Y coordinate += ICON_HEIGHT;
        }
        X coordinate += (ICON_WIDTH + gap);
        Y coordinate  = starting Y coordinate;
}
draw line bet buttons on both sides of the spinning columns;
}
```

The background is a single 16*16 bitmap and it fills up the whole screen by using REPEAT as the value for both wrapx and wrapy parameters in the BITMAP_SIZE function. REPEAT should only be used if the corresponding axis dimension is power of two, otherwise the result is undefined.

While the spin button is pressed the *current_icon* index is randomly generated during each update loop. After the *spin* button is released, a random velocity is then generated for each column and the velocity decreases at a fixed rate till it reaches 0, which is when the column stops.

The purpose of the L8 formatted overlay bitmap is to provide a smooth color transition from the outer edges of the icons to the inner edges of the outer overlay bitmap by blending with the already drawn icons.

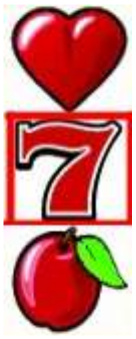
Figure 3-1 Plain icons

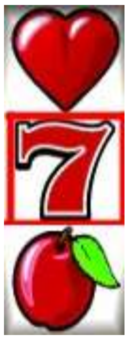
Figure 3-2 L8 overlay bitmap applied

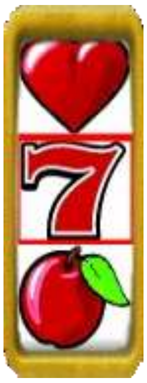
Figure 3-3 Outer overlay bitmap applied

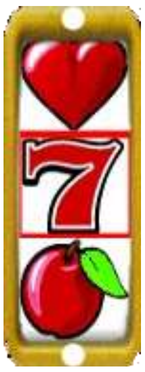
Figure 3-4 White dots as the selected-column indicator

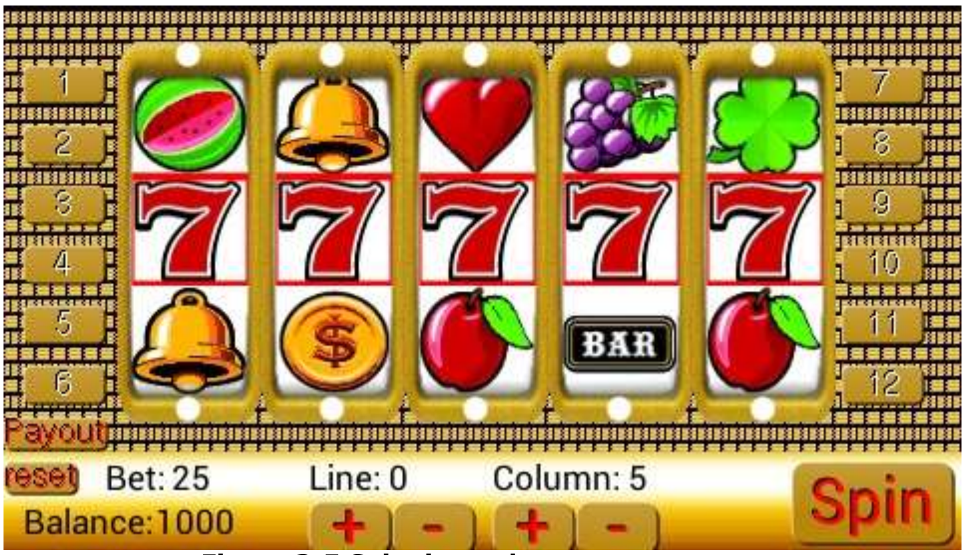

Figure 3-5 Spinning-column screen

The other screen is the payout screen which displays icons and their respective payout amount in a scrollable fashion. The application draws the payout table when the *payout* button is pressed.

The button text turns to "exit" whenever the payout table screen is shown and pressing the *exit* button returns to the *spinning-columns* screen. The payout table is displayed on a pixel basis as opposed to the spinning columns' full icon per update so the scrolling would be smooth. The payout table is portrayed as a single image strip with all the icons stacked on top of each other in the order of their bitmap handle. The topPoint variable keeps track of the current pixel position in the payout table during scrolling. The construction of the payout table is as follows:

```
int currentMultiplier;  /*how many multiples of the current drawing icon*/
int startingX=starting X coordinate of drawing pane;
int startingY;
int topIndex;
static int topPoint=0;  /*current pixel location of the table*/
static int movement_shift;  /*how much has scrolled*/
static int multiplier=5;  /*the multiplier of the current top icon*/


update movement_shift from scrolling;
add or subtract topPoint with movement_shift, depends on movement shift's signage, and then
        decrease movement shift;
update multiplier and topPoint according to topPoint's position; /* if(topPoint<0) then increase
        multiplier and topPoint=table height + topPoint; if(topPoint>table height) then decrease
        multiplier and topPoint%=table height;*/
topIndex=topPoint/ICON_HEIGHT;
startingY=(topPoint%ICON_HEIGHT)*(-1); /*hide part of an icon, if necessary, by drawing it in an
off-screen position. VERTEX2F command is needed to draw negative coordinates*/
currentMultiplier=multiplier;
construct drawing pane to confine the bitmaps;
for(i=0;i<number of icons that can appear in drawing pane;i++){
        cap and update topIndex and currentMultiplier;
        draw the topIndex bitmap handle at startingX and startingY;
        update startingX for the drawing of currentMultiplier and payout amount;
        update startingY for next icon;
        reset startingX for next icon;
        topIndex++;
}
```

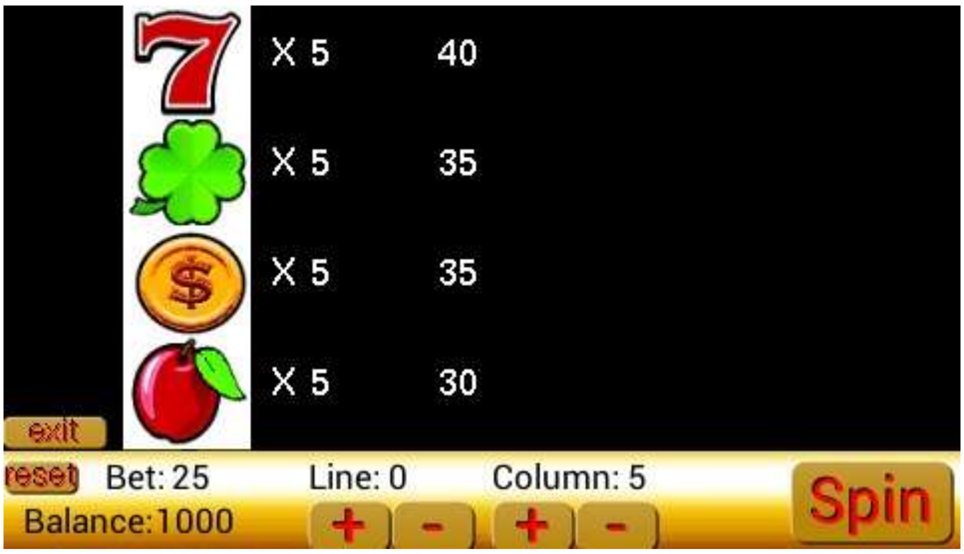

Figure 3-6 Payout Screen

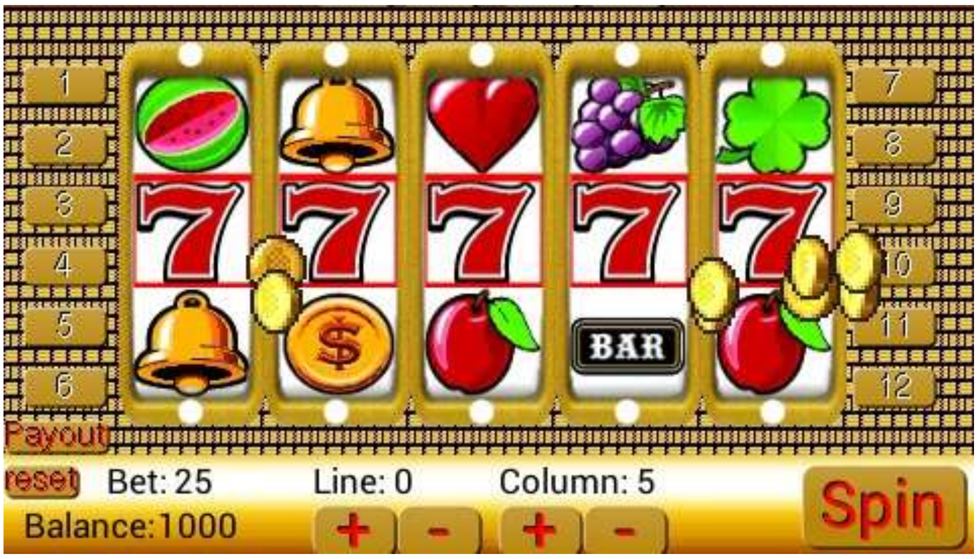

Figure 3-7 Bouncing Coins

After the spinning columns have stopped spinning, the appropriate amount of points is rewarded as shown in the payout table. Only the combination of the highest value is being considered for rewards, but multiple winning combinations are possible if they have the same amount of winning symbols. The winning symbols in the combination are highlighted and the total amount of points won is briefly displayed in the middle of the screen. Bouncing coins can also appear and the numbers of coins are proportionally to the points won. The reward algorithm is as follows:

```
{
int lookingFor=number of selected columns;  /*check for  multiples first*/
int tempIndex[total icons];  /*holds the drawn indexes for each column*/
int winningIndex[number of columns]; /*holds the winning bitmap handles in corresponding
        column*/
int rewardPoints;
for(i=0;i< number of selected spin columns;i++){
        tempIndex[bitmap handle of this column's drawn index]++;
}


for(i=0;i<total icons;){
        if(tempIndex[i]==lookingFor){
        for(j=0;j<number of selected spin columns;j++){
        if(bitmap handle for the current column's drawn index == i)
                queue bitmap handle in winningIndex;  //winning indexes have colored boxes on
top.
        }
        if(lookingFor>2)
                return points earned;


         (lookingFor==2)
                points+=points earned;
        }
        i++;
        if(i==total icons){
                lookingFor--;
                i=0;
                if(lookingFor<2){
                        if(points earned > 0)
                        return rewardPoints;
                        break;
                        }
                }
}


check for fruit icons;
return fruit icon points earned;
}
```

3.2.1 Fonts

All fonts used in this application are FT8XX built-in fonts. Various fonts are used for the status text on the status bar and in the payout table.

3.2.2 Audio

The sound effects used in this application are from the FT8XX's sound synthesizer. Sound effects are used during column spinning, reward coin collision, button pressed, and points rewarding.

3.2.3 Buttons

All buttons used in this application are FT8XX built-in widget buttons.

4 Contact Information

Head Quarters – Singapore

Bridgetek Pte Ltd
178 Paya Lebar Road, #07-03
Singapore 409030
Tel: +65 6547 4827
Fax: +65 6841 6071

E-mail (Sales)	sales.apac@brtchip.com
E-mail (Support)	support.apac@brtchip.com

Branch Office – Taipei, Taiwan

Bridgetek Pte Ltd, Taiwan Branch
2 Floor, No. 516, Sec. 1, Nei Hu Road, Nei Hu District
Taipei 114
Taiwan , R.O.C.
Tel: +886 (2) 8797 5691
Fax: +886 (2) 8751 9737

E-mail (Sales)	sales.apac@brtchip.com
E-mail (Support)	support.apac@brtchip.com

Branch Office - Glasgow, United Kingdom

Bridgetek Pte. Ltd.
Unit 1, 2 Seaward Place, Centurion Business Park
Glasgow G41 1HH
United Kingdom
Tel: +44 (0) 141 429 2777
Fax: +44 (0) 141 429 2758

E-mail (Sales)	sales.emea@brtichip.com
E-mail (Support)	support.emea@brtchip.com

Branch Office – Vietnam

Bridgetek VietNam Company Limited
Lutaco Tower Building, 5th Floor, 173A Nguyen Van Troi,
Ward 11, Phu Nhuan District,
Ho Chi Minh City, Vietnam
Tel : 08 38453222
Fax : 08 38455222

E-mail (Sales)	sales.apac@brtchip.com
E-mail (Support)	support.apac@brtchip.com

Web Site

http://brtchip.com/

Distributor and Sales Representatives

Please visit the Sales Network page of the Bridgetek Web site for the contact details of our distributor(s) and sales representative(s) in your country.

Appendix A – References

Document References

- FT800 datasheet
- Programming Guide covering EVE Command Language
- AN_391 EVE Platform Guide
- AN_240 FT800 from the Ground Up
- AN_245 VM800CB SampleApp PC Introduction – Covering detailed design flow with a PC and USB to SPI Bridge Cable
- AN_246 VM800CB SampleApp Arduino Introduction – Covering detailed design flow in an Arduino Platform
- AN_325 FT9XX Toolchain Installation Guide
- AN_281 FT800 Emulator Library User Guide – Covering API Interface for FT800 Emulator
- AN_252 FT800 Audio Primer
- http:/brtchip.com/utilities/ - EVE Screen Editor link
- https://www.ardunio.cc/ - Arduino IDE

Acronyms and Abbreviations

Terms	Description
EVE	Embedded Video Engine
MCU	Microcontroller
SPI	Serial Peripheral Interface
MSVC	Microsoft Visual C
FAT	File Allocation Table
SD	Secure Digital

Appendix B – List of Figures & Tables

List of Figures

List of Tables

NA

Appendix C – Revision History

Document Title:	AN_306 FT800 Jackpot Application
Document Reference No.:	BRT_000195
Clearance No.:	BRT#114
Product Page:	http://brtchip.com/product/
Document Feedback:	Send Feedback

Revision	Changes	Date
1.0	Initial Release	2014-03-25
1.1	Updated with FT900 platform; Document migrated from FTDI to BRT (Updated company logo; copyright info; contact information; hyperlinks)	2018-01-04