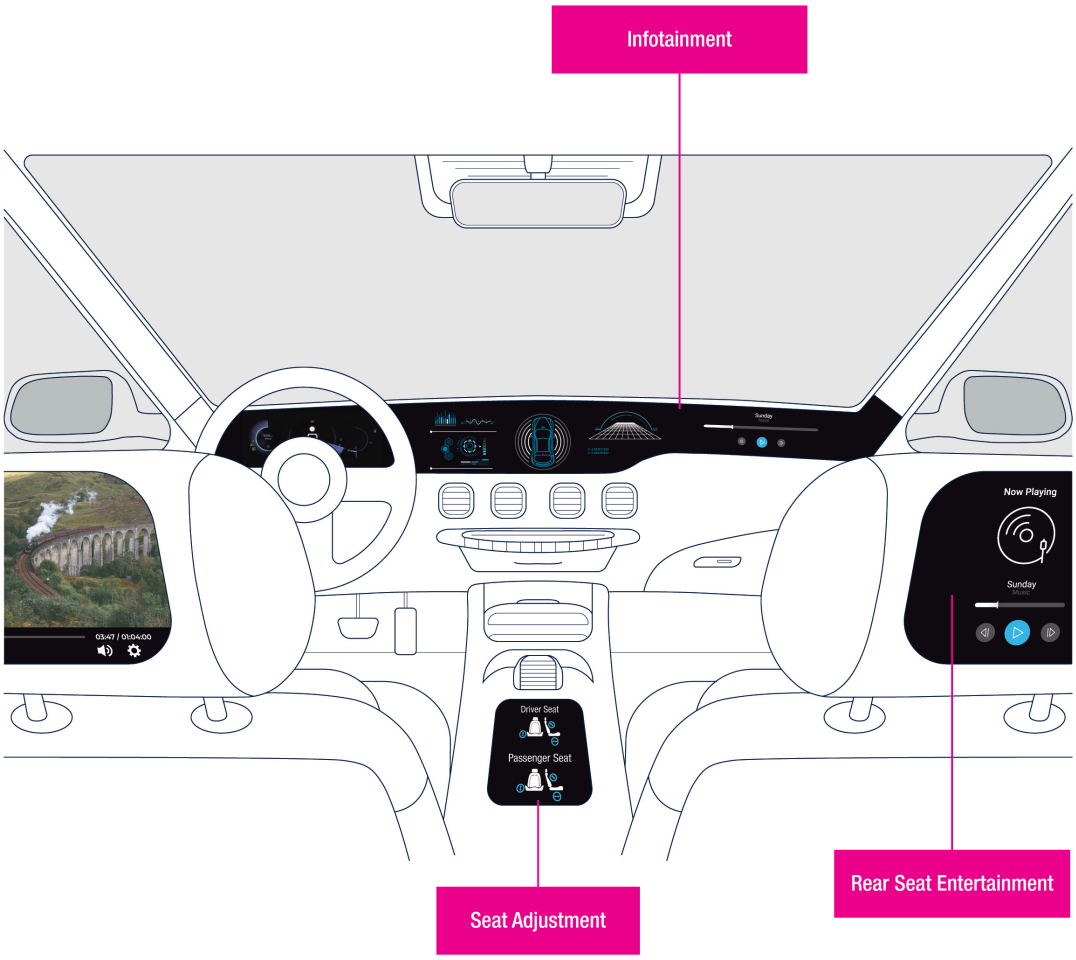
Getting started with the AEK-LCD-DT028V1 display expansion board with resistive touch for Chorus family

Introduction

The AEK-LCD-DT028V1 evaluation board hosts a 2.8” LCD display with resistive touch for a graphical user interface (GUI) which interacts with SPC58 MCU discovery boards. It aims at providing, in the prototyping phase, a simple and fast tool to display information or simple menus in automotive applications.

A new trend in modern vehicles calls for several auxiliary displays to complement the main infotainment system. The additional displays are employed for status feedback reporting, for extended infotainment features like in rear-seats and for basic control to replace standard buttons and levers like in seat adjustment, mirror positioning or headlight height adjustment.

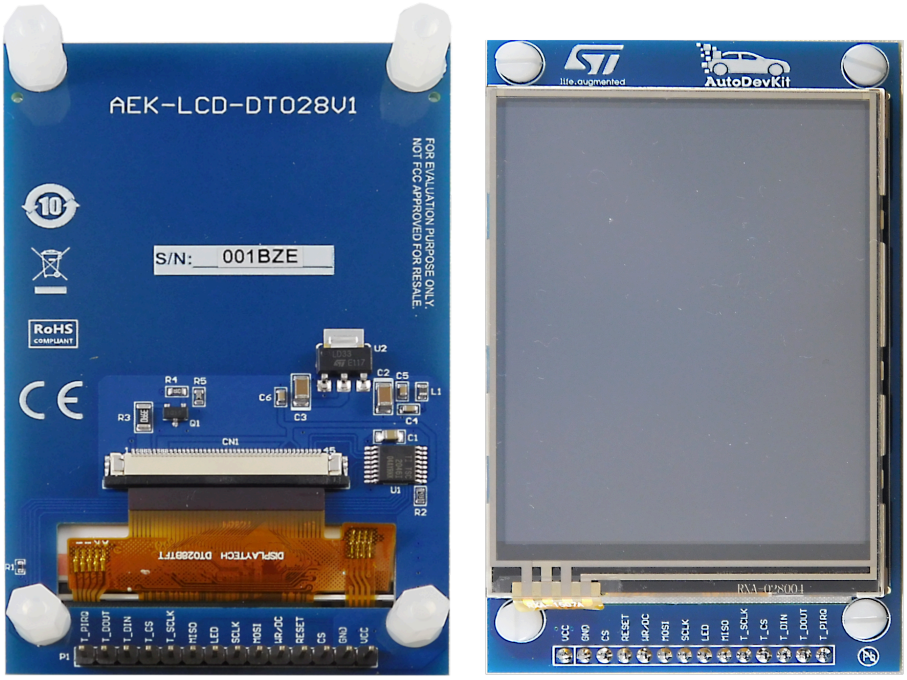
Figure 1. Extended infotainment system



The board can be easily evaluated thanks to a common platform consisting of SPC5-STUDIO and AutoDevKit (plug-in of SPC5-STUDIO based on the Eclipse platform). This tool-set facilitates the configuration of the board and the microcontroller peripherals used and provides automatic code generation capability exploiting a template-based engine.

The AEK-LCD-DT028V1 has a 240x320 pixels LCD resolution and interfaces with an ILI9341 IC driver, managing the display, and a TSC2046, managing the touch feedback, via SPI communication (4 wires).

Figure 2. **AEK-LCD-DT028V1 evaluation board (top and bottom views)**



Warning:

The AEK-LCD-DT028V1 evaluation board has not to be used in a vehicle as it is designed for R&D laboratory use only.

1 Getting started

1.1 Features

- 2.8 "(240x320 pixel) TFT SPI LCD with resistive touch managed by an SPI touch screen controller available on the board
- PCB header connector interfacing with SPC5 MCU discovery boards
- 3.3V LDO voltage regulator for I/O signals
- 53 mm x 87 mm
- WEEE and RoHS compliant

1.2 Hardware overview

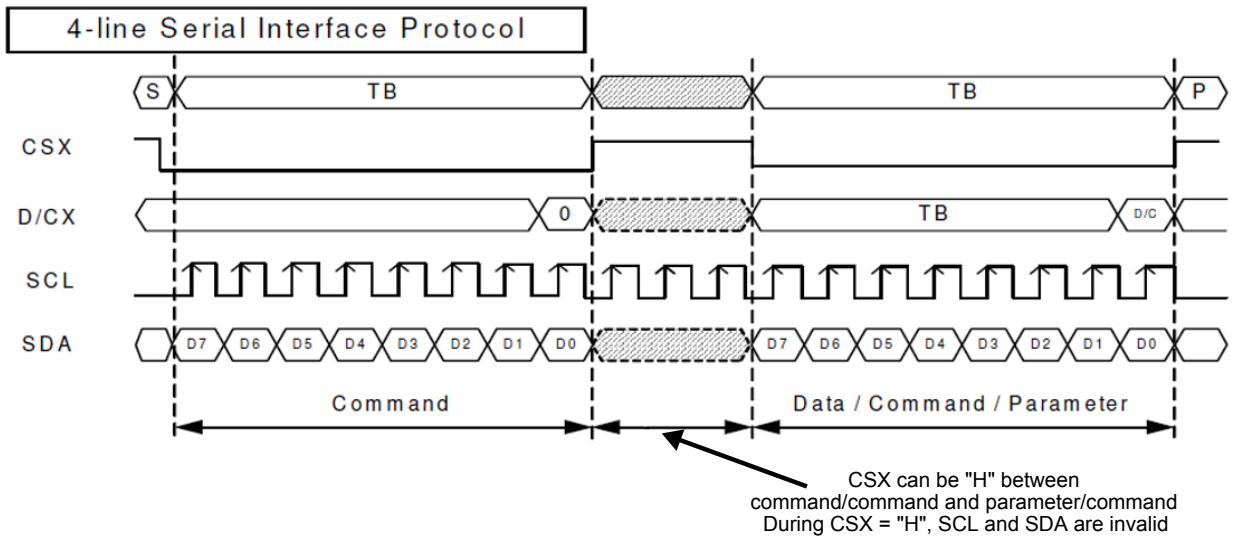
1.2.1 Display control circuit

The display control circuit is managed by an ILI9341 controller. It is a 262, 144-color single-chip SOC driver used for a TFT liquid crystal display with a resolution of 240RGBx320 dots, 172,800 bytes GRAM for graphic display data, and power supply circuit.

The ILI9341 supports different communication protocols such as parallel communication at 8-bit, 9-bit, 16-bit, 18-bit, and SPI communication via three or four wires. It also supports display at 65K, 262K RGB color, display rotation and scroll view.

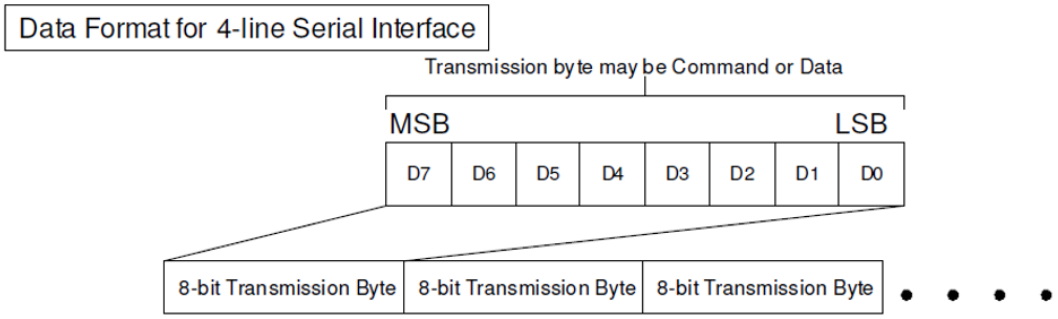
In the AEK-LCD-DT028V1 evaluation board, the communication between the ILI9341 and the external microcontroller is managed via 4-wire SPI communication. Moreover, the ILI9341 controller is configured to work in 16-bit (RGB565) mode in order to display up to 65K colors per pixel.

Figure 3. SPI communication protocol implemented by ILI9341



As we can see from the above image, only when the CSX (chip select) is low the communication is enabled and, depending on the value of D/CX data control pin, the data sent by SDA are interpreted as commands or as data: when D/CX is high, the SDA data are interpreted by the controller as commands, whereas when it is low, they are interpreted as data.

Figure 4. **Data Format**



1.2.2 Touch control circuit

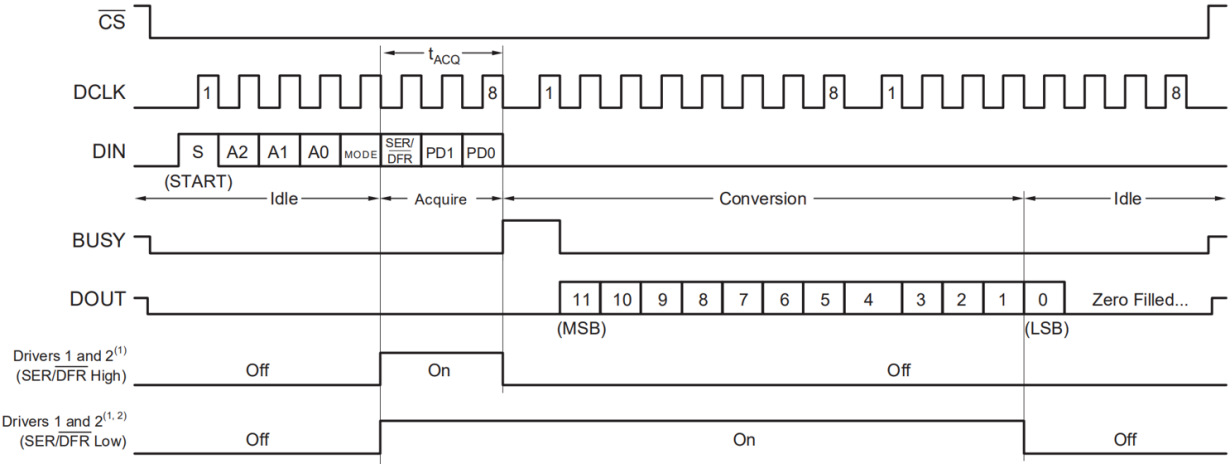
The touch control circuit is managed by the TSC2046. It is a next-generation version to the ADS7846 4-wire touch screen controller which supports a low-voltage I/O interface from 1.5 V to 5.25 V. The TSC2046 can detect the pressed screen location and measure touch screen pressure. The communication between the TSC2046 and the external microcontroller is performed via SPI.

On the AEK-LCD-DT028V1 evaluation board, every time the TSC2046 detects a touch on the display, it raises an external interrupt to inform the MCU that a touch event occurred. Then, the MCU sends a request via SPI communication to the TSC2046 to retrieve the X and Y coordinates related to the touch just detected. The external interrupt is disabled and low during the measurement cycle for the X and Y coordinates.

Figure 5. **SPI communication protocol implemented by TSC2046**

Note: (1) For Y position, driver 1 is on when X+ is selected and driver 2 is off. For X position, driver 1 is off, Y+ is selected and driver 2 is on. Y- turns on when power-down mode is entered and PD0 = 0.

Note: (2) Drivers remain on if PD0 = 1 (power-down mode not activated) until the selected input channel, reference mode or power-down mode is changed or CS is high.



In the figure above, the source of the digital signals is a microcontroller with a basic serial interface. The communication between the processor and the converter consists of eight clock cycles. One complete conversion can be accomplished with three-serial communication, for a total of 24 clock cycles on the DCLK input. The first eight clock cycles are used to provide the control byte via the DIN pin. When the converter has enough information, it enters acquisition mode.

The acquisition time is included in the first eight clock cycles while the next twelve clock cycles accomplish the actual analog to-digital conversion. In AEK-LCD-DT028V1, the TSC2046 is configured to complete the conversion in thirteen clock cycles. To complete the D_{OUT} replay message, three more clock cycles are needed but they will be ignored by the converter.

1.2.3 AEK-LCD-DT028V1 pin-out description

Table 1. AEK-LCD-DT028V1 pin-out

Pin	Symbol on PCB	Description
1	VCC	5 V power supply
2	GND	Ground
3	CS	Chip select signal
4	RESET	Reset control signal
5	WR/DC	Data selection control signal
6	MOSI	SPI bus master output slave input
7	SCLK	SPI clock signal
8	LED	LCD back light control signal
9	MISO	SPI bus master input slave output
10	T_SCLK	Touch screen SPI clock signal
11	T_CS	Touch screen chip select signal
12	T_DIN	Touch screen SPI bus master output slave input
13	T_DOUT	Touch screen SPI bus master input slave output
14	T_PIRQ	Touch screen interrupt

2 AutoDevKit software library for AEK-LCD-DT028V1 board

All the development of the drivers related to the LCD display based on AEK-LCD-DT028V1 are included in a component belonging to AutoDevKit software (STSW-AUTODEVKIT) version 1.5.0 (or higher).

The library is written in C and the target software is automatically generated according to the code generation and pin allocation paradigm included in AutoDevKit design flow.

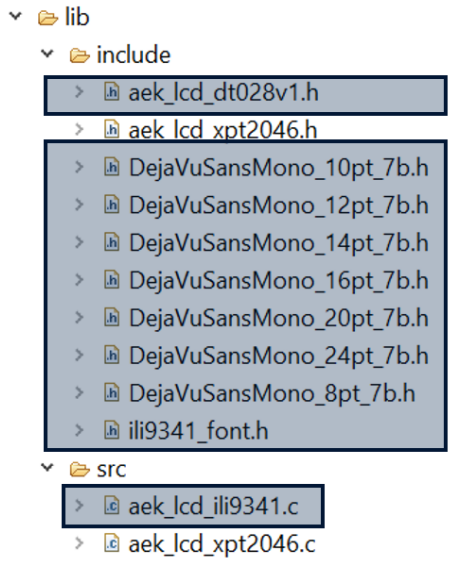
The library implemented to manage the AEK-LCD-DT028V1 board can be divided into:

- Graphic library
- Touch library

2.1 Graphic library

The graphic library implements screen orientation, basic geometric figures (e.g., rectangles, triangles, circles, lines) and the respective fill functions. It also includes functions dedicated to the visualization of characters and strings with different fonts and colors.

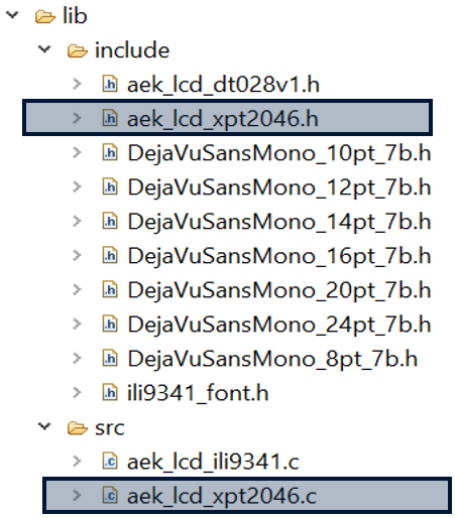
Figure 6. AEK-LCD-DT028V1 graphic library files



2.2 Touch library

The touch library implements the function to retrieve the X and Y coordinates (not in pixel) of the touched point on the display and the functions to manage the external interrupt when the touch is detected.

Figure 7. **AEK-LCD-DT028V1 touch library files**

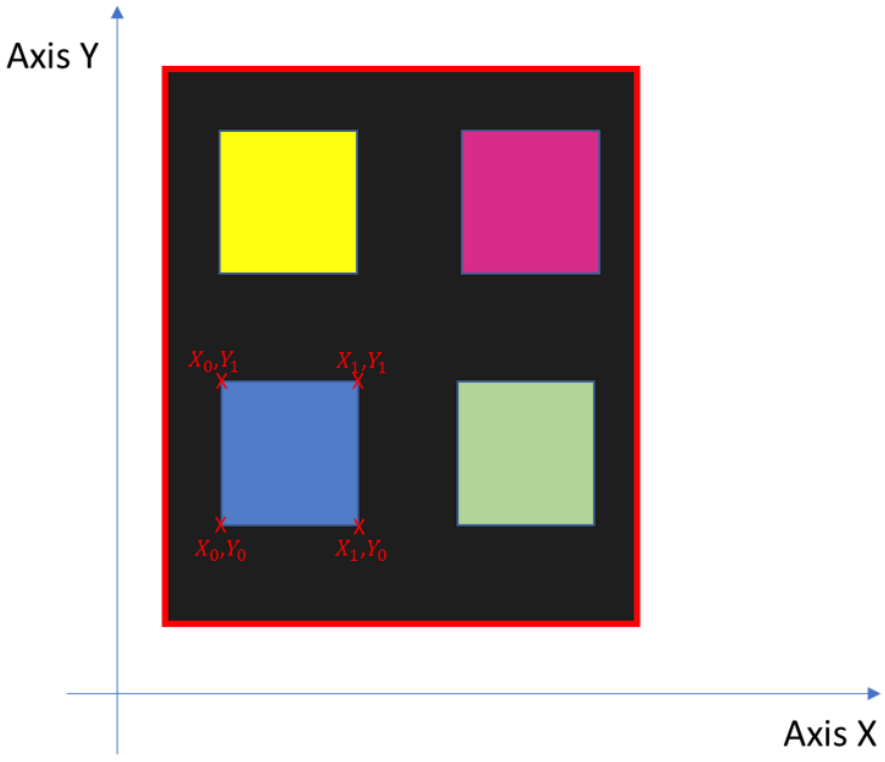


2.3 Usage of graphic and touch libraries example

This example is based on the assumption that we want to display a menu of four buttons on the AEK-LCD-DT028V1 LCD as showed in the image below and add an action each time a button is pressed.

Note: This square-shaped approximation could be used also to display more complex images. In this case, the square would represent the touchable area behind the image.

Figure 8. **AEK-LCD-DT028V1 square-shaped touch menu representation**



The following steps shows how to create the code required to implement a proper detection of the blue square (bottom left one).

Step 1. Use the graphic library to draw the four squares representing our buttons.

The required driver function is:

```
aek_ili9341_drawRect(AEK_LCD_DEV0, 10, 80, 105, 110, ILI9341_COLOR_BLUE);
```

and the input parameters are:

```
*@brief   Draw a Rectangle
*
*@param[in] dev  LCD_Component
*@param[in] x, y  top left coordinates
*@param[in] h, w   rectangle dimension
*@param[in] color  RGB565 color
```

Step 2. To retrieve the X, Y absolute coordinates of the blue square (bottom left one), touch the vertices of the blue square one by one in progression.

At every touch detection in the code, a driver function is invoked to read and store the returned position values. The procedure will be repeated four times to cover all square vertices.

```
i=0;
for (; ;) {
    if (aek_lcd_get_touchFeedback(AEK_LCD_DEV0) != 0U)   { // the screen has been
touched
            if (i<4) { // we are in calibration phase to identify the blue square
coordinates
            aek_lcd_read_touch_pos(AEK_LCD_DEV0, x_value[i], y_value[i]);//store in
the array each vertex position
            i++; }//move to next corner
            else {
                  // first time save the extreme corners of the square in the
xx_BlueButton variables
                  aek_lcd_read_touch_pos(AEK_LCD_DEV0, &x_value. &y_value] // we
are in normal touch detection phase
                  if ((X)_BlueBotton < x_value && x_value < X1_BlueButton) &&
(Y0_BlueButton< y_value && y_value <
                  Y1_BlueButton)) {   /*do something*/ }
                }
             // reset the state of LCD Touch in order to detect a new touch.
                aek_lcd_set_touchFeedback(AEK_LCD_DEV0)
   }
}
```

The input parameters of the `aek_lcd_read_touch_pos()` function are:

```
*@param[in] dev  LCD_Component
*@param[in] x  coordinate x read
*@param[in] y  coordinate y read
```

The blue square touching area is now identified.We Once received, we read the new position detected and we compare it with the *xx_BlueBotton* variables to verify if the touch is on the blue square.

Step 3. To test if the touches are inside or outside the blue square, store the extreme corner of the square in four `xx_BlueBotton` variables and wait for the next touch event ("else" case of the above code).

```
if((X0_BlueButton  < x_value  && x_value  < X1_BlueButton)  &&  (Y0_BlueButton  <
y_value  &&  y_value  <  Y1_BlueButton))
{
    //  do something

}
```

3 AutoDevKit ecosystem

The application development employing the AEK-LCD-DT028V1 takes full advantage of the AutoDevKit ecosystem, whose basic components are:

- SPC5-STUDIO integrated development environment (IDE)
- PLS UDE programmer and debugger
- AutoDevKit software library (STSW-AUTODEVKIT)
- AEK-LCD-DT028V1 driver

3.1 SPC5-STUDIO

SPC5-STUDIO is an integrated development environment (IDE) based on Eclipse designed to assist the development of embedded applications based on SPC5 Power Architecture 32-bit microcontrollers.

The package includes an application wizard to initiate projects with all the relevant components and key elements required to generate the final application source code. It also contains straightforward software examples for each MCU peripheral.

SPC5-STUDIO also features:

- the possibility of integrating other software products from the standard Eclipse marketplace
- free license GCC GNU C Compiler component
- support for industry-standard compilers
- support for multi-core microcontrollers
- PinMap editor to facilitate MCU pin configuration

Note: You need to download the SPC5-UDESTK-SW software to run and debug applications created with SPC5-STUDIO.

3.2 STSW-AUTODEVKIT

The STSW-AUTODEVKIT plug-in for Eclipse extends SPC5-STUDIO for automotive and transportation applications.

STSW-AUTODEVKIT features:

- integrated hardware and software components, component compatibility checking, and MCU and peripheral configuration tools
- the possibility of creating new system solutions from existing ones by adding or removing compatible function boards
- new code can be generated immediately for any compatible MCU
- high-level application APIs to control each functional component, including the ones for the AEK-LCD-DT028V1 board

The GUI helps configure interfaces, including SPI, and can automatically manage all relevant pin allocation and deallocation operations.

For more information, refer to UM2623 (in particular, Section 6 and Section 7) or watch the video tutorials.

3.3 AEK_LCD_DT028V1 component

The AEK-LCD-DT028V1 evaluation board drivers are provided with the STSW-AUTODEVKIT (from version 1.5.0 on) installation to facilitate the programming phase.

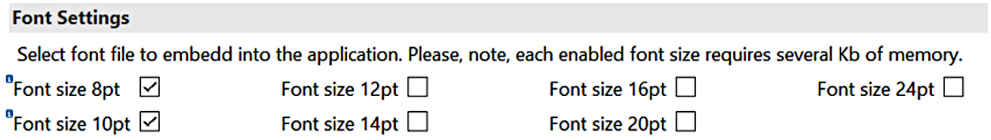
Update your AutoDevKit installation to get the latest version. Once properly installed, you will be able to select the component named *AEK_LCD_DT028V1 Component RLA*.

3.3.1 AEK_LCD_DT028V1 component configuration

To configure the component, follow the procedure below.

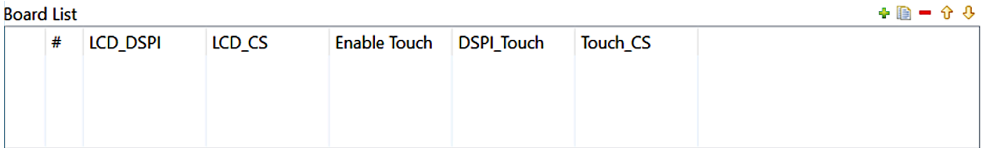
Step 1. Select the font file to be embedded into your application.

Figure 9. Selecting the font size file



Step 2. Add a row (by clicking on the + sign) in the board list for each LCD board that you would like to use.

Figure 10. Adding rows

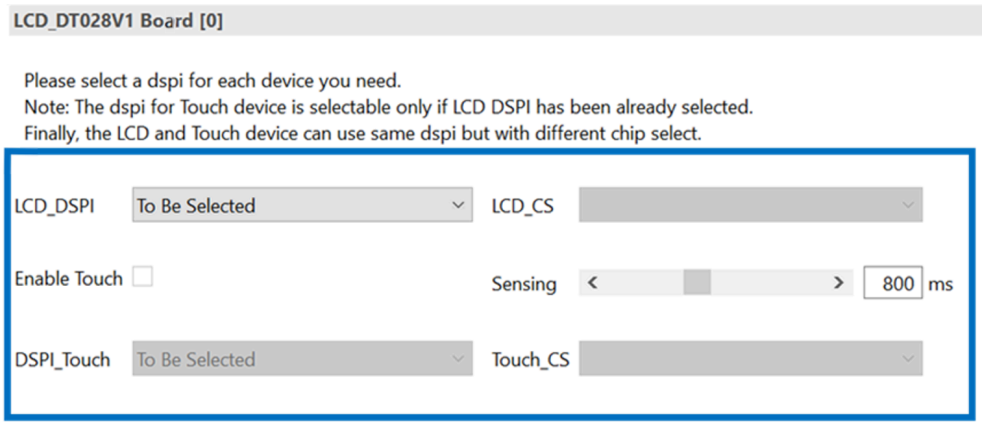


Step 3. Double click on the element added.

Step 4. Choose one among the following options:

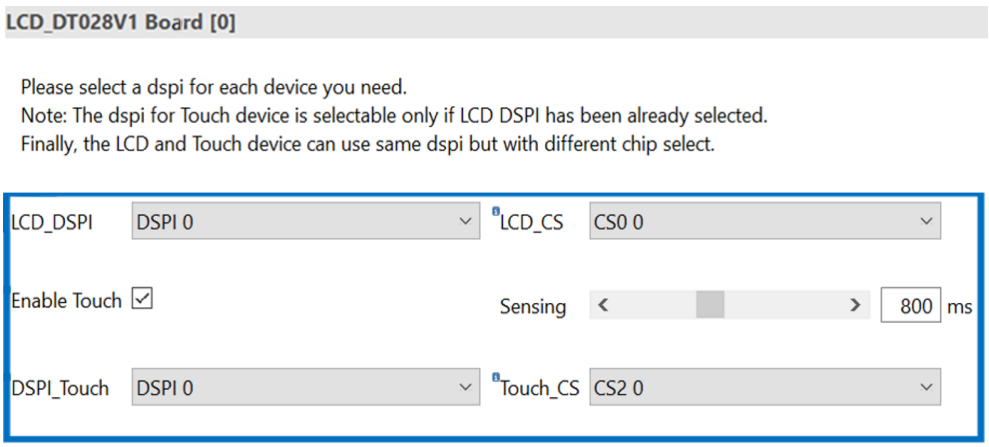
- **LCD_DSPI + LCD_CS** to use the display only, without touch feature
- **LCD_DSPI + LCD_CS + Enable_Touch and Sensing** to use the display features with the touch detection only. Selecting this configuration, you will not be able to retrieve the X, Y coordinates.
- ◦ **LCD_DSPI + Enable_Touch + DSPI_Touch + Sensing** to use all board features.

Figure 11. Configuration options



Note: To optimize the number of the allocated pins, it is possible that both LCD and Touch share the same DSPI peripheral with two different chip selects.

Figure 12. Adding rows



The configuration is complete. You can return to the main tab and proceed by pressing the automatic pin allocation button.

4 AEK-LCD-DT028V1 sample application

4.1 How to create a simple AEK-LCD-DT028V1 sample application

This example allows you to create an application that asks to tap the "X" character on the screen and returns the coordinates of the touched point.

Step 1. Create a new SPC5-STUDIO application for the SPC58EC series microcontroller and add the following components:

- SPC58ECxx Init Package Component RLA
- SPC58ECxx Low Level Drivers Component RLA

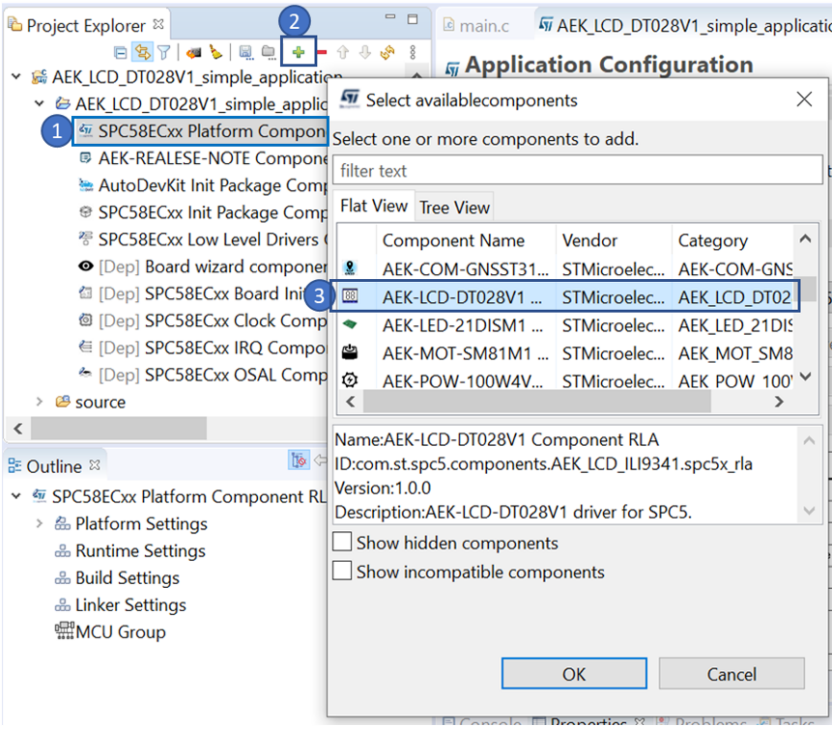
Note: If these components are not added, the other components are not visible.

Step 2. Add the following additional components:

- AutoDevKit Init Package Component
- SPC58ECxx Platform Component RLA
- AEK-LCD-DT028V1 Component RLA

Figure 13. SPC5-STUDIO - adding AEK-LCD-DT028V1 component

1. SPC58ECxx Platform Component RLA
2. Open available components
3. AEK-LCD-DT028V1 Component RLA

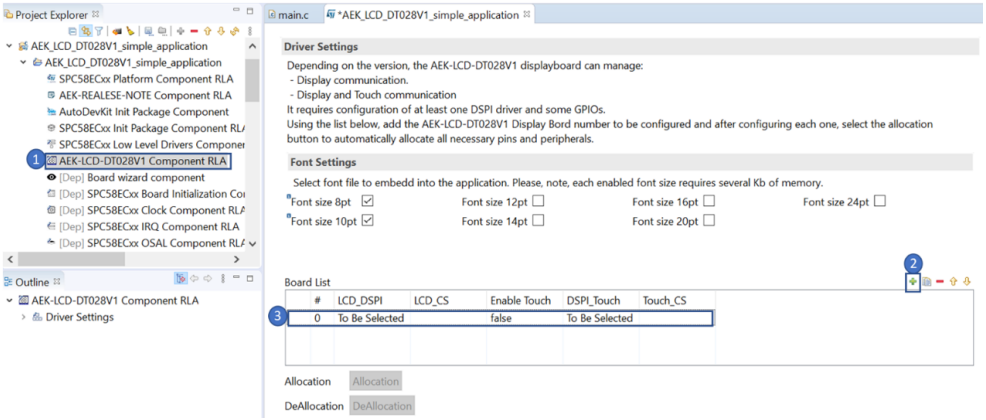


Step 3. Select [**AEK-LCD-DT028V1 Component RLA**] to open the [**Application Configuration**] window.

Step 4. Click on [**+**] to add a new element to the board list.

Figure 14. AEK-LCD-DT028V1 component configuration

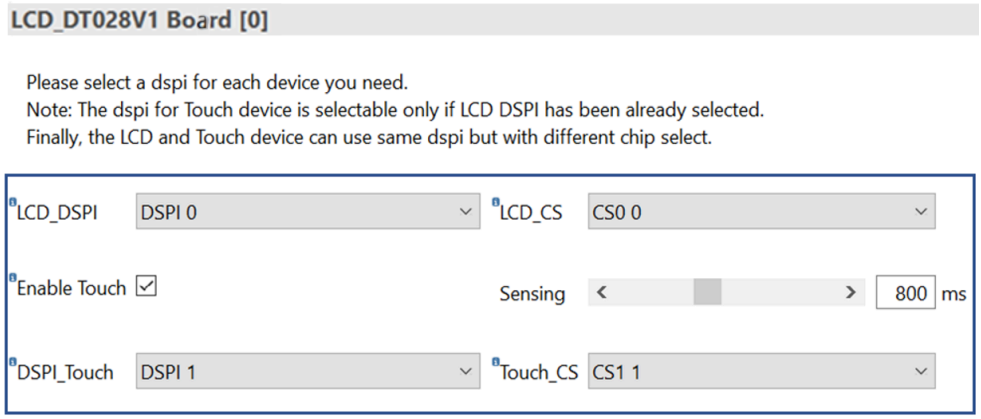
1. AEK-LCD-DT028V1 component
2. Add new element icon (+)
3. New entry



Step 5. Double click on the newly added element to configure the board.

Step 6. Configure the board as shown in the image below.

Figure 15. AEK-LCD-DT028V1 sensor configuration

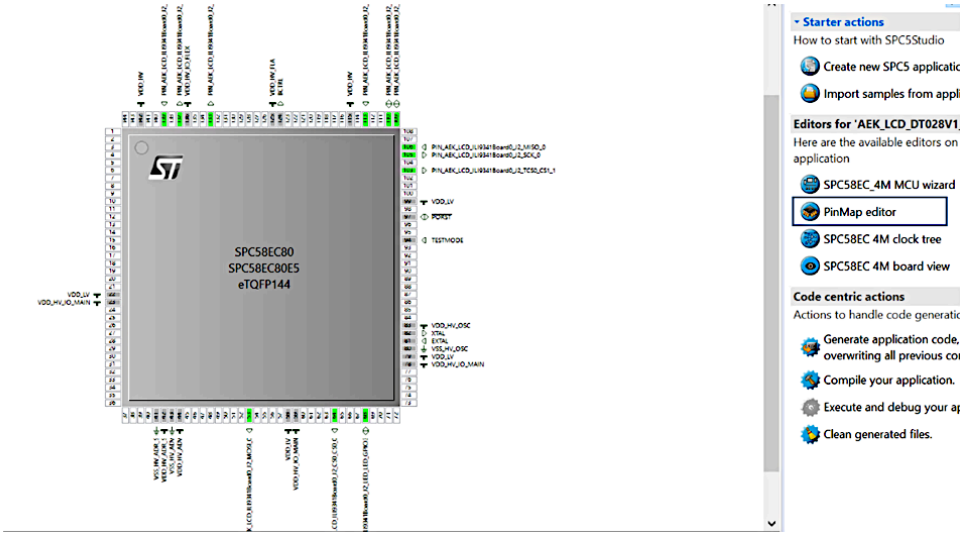


Step 7. Click the [**Allocation**] button below the AEK-LCD-DT028V1 list and click [**OK**] in the confirmation window.

This operation delegates automatic pin allocation to AutoDevKit.

Step 8. Click on the PinMap editor icon to check that required pins have been properly allocated.

Figure 16. AEK-LCD-DT028V1 PinMap configuration



Step 9. Close the PinMap Editor and save the application.

Step 10. Generate and build the application using the appropriate icons in SPC5-STUDIO.

The project folder will be populated with new files, including main.c and the components folder with AEK-LCD-DT028V1 drivers.

Step 11. Open the *main.c* file and include *AEK-LCD-DT028V1.h* file.

```
#include "components.h"
#include "aek_lcd_dt028v1.h"
#include <stdio.h>

void *sbrk(size_t incr)
{
     extern uint8_t __heap_base__;
     extern uint8_t __heap_end__;
     static uint8_t *p=&__heap_base__;
     static uint8_t *newp;

     newp = p+ incr;
     if(newp> &__heap_end__)
    {
        return (void*)-1;
     }
     return p =newp;
}

/*
 * Application entry point.
 */
int main(void)
{

     char buffer[16];
     uint16_t x_value, y_value;
    const FontObject_t *font10pt = (const FontObject_t *)&DejaVuSansMono_10pt_7b;
    const FontObject_t *font8pt = (const FontObject_t *)&DejaVuSansMono_8pt_7b;

  /* Initialization of all the imported components in the order specified in
     the application wizard. The function is generated automatically.*/
  componentsInit();



 /* Enable Interrupts */
  irqIsrEnable();

 /* initialize AEK-LCD-DT028V1 */
  aek_ili9341_init(AEK_LCD_DEV0);
  /* set the orientation */
  aek_ili9341_setOrientation(AEK_LCD_DEV0,ILI9341_PORTRAIT);
  /* crop screen */
  aek_ili9341_cropScreen(AEK_LCD_DEV0,aek_ili9341_getScreenWidth(AEK_LCD_DEV0),
  aek_ili9341_getScreenHeight(AEK_LCD_DEV0));
  /* clear screen */
  aek_ili9341_clearScreen(AEK_LCD_DEV0,ILI9341_COLOR_BLACK);
  /* drow string */
  (void)aek_ili9341_drawString(AEK_LCD_DEV0,60, 30, "Test Touch: ",
ILI9341_COLOR_WHITE, font10pt);
  (void)aek_ili9341_drawString(AEK_LCD_DEV0,3, 50, "Pls. touch the screen to
visualize x and y coordinates of the touched point ",
                                                    ILI9341_COLOR_WHITE, font8pt);
  (void)aek_ili9341_drawString(AEK_LCD_DEV0,3, 60, "x and y coordinates of the
touched point ", ILI9341_COLOR_WHITE, font8pt);
  (void)aek_ili9341_drawString(AEK_LCD_DEV0,3, 72, "point ", ILI9341_COLOR_WHITE,
font8pt);

/* Application main loop.*/
  for ( ; ; )
  {
       /* Detect if LCD touch has been touched. */
       if (aek_lcd_get_touchFeedback(AEK_LCD_DEV0) != 0U)
      {
           aek_lcd_read_touch_pos(AEK_LCD_DEV0, &x_value, &y_value);
           /* clear screen */
```

```
            aek_ili9341_clearScreen(AEK_LCD_DEV0,ILI9341_COLOR_BLACK);

            snprintf(buffer, sizeof(buffer), "X = %d", x_value);
            (void)aek_ili9341_drawString(AEK_LCD_DEV0, 10, 180, buffer,
ILI9341_COLOR_WHITE, font10pt);

             snprintf(buffer, sizeof(buffer), "Y = %d", y_value);
             (void)aek_ili9341_drawString(AEK_LCD_DEV0,10, 200, buffer,
ILI9341_COLOR_WHITE, font10pt);

              /* wait 1sec*/
              osalThreadDelayMilliseconds(1000);
              aek_ili9341_clearScreen(AEK_LCD_DEV0,ILI9341_COLOR_BLACK);

              /* draw string */
              (void)aek_ili9341_drawString(AEK_LCD_DEV0,60, 30, "Test Touch: ",
ILI9341_COLOR_WHITE, font10pt);
              (void)aek_ili9341_drawString(AEK_LCD_DEV0,3, 50, "Pls. touch the
screen to visualize x and y coordinates of the touched point ",

ILI9341_COLOR_WHITE, font8pt);
              (void)aek_ili9341_drawString(AEK_LCD_DEV0,3, 60, "x and y coordinates
of the touched point ", ILI9341_COLOR_WHITE, font8pt);
              (void)aek_ili9341_drawString(AEK_LCD_DEV0,3, 72, "point ",
ILI9341_COLOR_WHITE, font8pt);
              /* reset the state of LCD Touch in order to detect a new touch. */
              aek_lcd_set_touchFeedback(AEK_LCD_DEV0);
        }
    }
}
```

Step 12. Save, generate, and compile the application.

Step 13. Open the BoardView Editor provided by AutoDevKit.
This provides a graphical point-to-point guide on how to wire the boards.

Step 14. Connect the AEK-LCD-DT028V1 to a USB port on your PC using a mini-USB to USB cable.

Step 15. Launch SPC5-UDESTK-SW and open the *debug.wsx* file in the AEK-LCD-DT028V1– Application /UDE folder.

Step 16. Run and debug your code.

4.2 Available demos for AEK-LCD-DT028V1

There are three different demos provided with the AEK-LCD-DT028V1 component:

- SPC58ECxx_RLA AEK-LCD-DT028V1 – LCD Touch - Test Application
- SPC582Bxx_RLA AEK_LCD_DT028V1- 1LCD NO touch - Test Application
- SPC582Bxx_RLA AEK_LCD_DT028V1- 1LCD touch - Test Application

Note: More demos might become available with new AutoDevKit releases.

4.3 How to upload the demos for AEK-LCD-DT028V1

Follow the procedure below to import the demos into SPC5-STUDIO.

Step 1. Select [**Import samples from application library**] from the Common tasks pane.
An Import application Wizard appears.

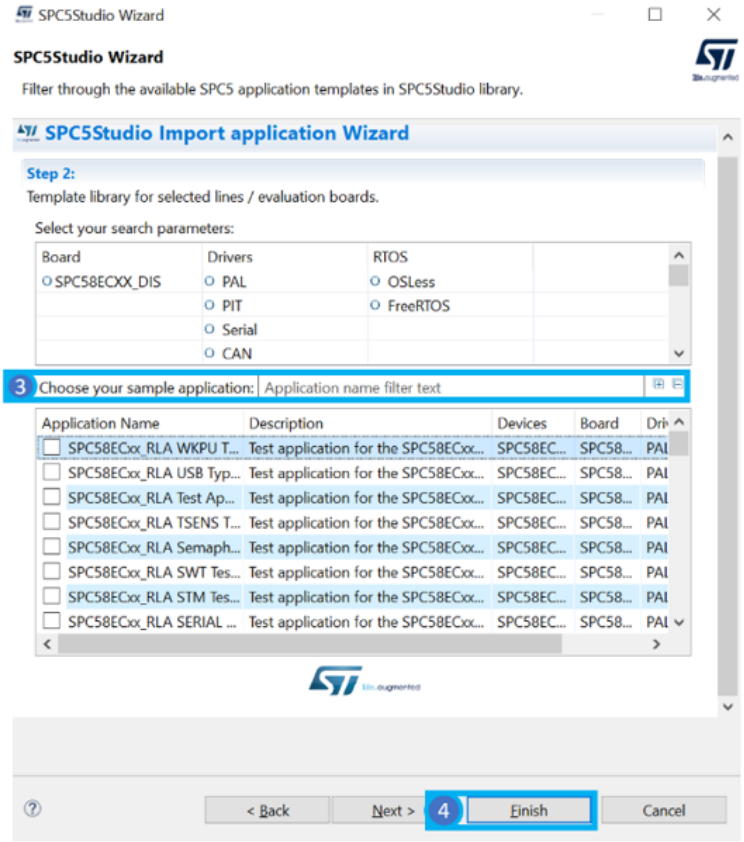
Step 2. Insert the appropriate product MCU family details.

Figure 17. AEK-LCD-DT028V1 demo upload (1 of 2)

Step 3. Select the desired application from the library.

Step 4. Click on the [**Finish**] button.

Figure 18. AEK-LCD-DT028V1 demo upload (2 of 2)



5 Available APIs

aek_ili9341_init	ILI9341 initialization
aek_ili9341_getScreenWidth	Gets the screen width
aek_ili9341_getScreenHeight	Gets the screen height
aek_ili9341_setOrientation	Sets the screen orientation
aek_ili9341_getOrientation	Gets the screen orientation
aek_ili9341_clearScreen	Clears the screen
aek_ili9341_cropScreen	Crops the screen
aek_ili9341_writeHLine	Draws a horizontal line
aek_ili9341_writeVLine	Draws a vertical line
aek_ili9341_drawLine	Draws a line
aek_ili9341_drawRect	Draws a rectangle
aek_ili9341_fillRect	Draws a filled rectangle
aek_ili9341_drawCircle	Draws a circle
aek_ili9341_fillCircle	Fills a circle
aek_ili9341_drawPixel	Draws a pixel
aek_ili9341_drawTriangle	Draws a triangle
aek_ili9341_drawImage	Draws an image
aek_ili9341_drawChar	Draws a character
aek_ili9341_drawString	Draws a string
aek_ili9341_verticalScroll	Vertical scroll
aek_lcd_read_touch_pos	Reads the x and y coordinates of the point touched on the display. The x and y coordinates are detected when the pen or finger is removed from the screen.
aek_lcd_set_touchFeedback	Resets the state of LCD touch in order to detect a new touch.
aek_lcd_get_touchFeedback	Detects whether the LCD touch has been pressed. If this function returns 1, it means an LCD touch has been detected, otherwise if it returns 0, it means that no touch has been detected.

6 Board versions

Table 2. **AEK-LCD-DT028V1 versions**

Finished good	Schematic diagrams	Bill of materials
AEK$LCD-DT028V1A (1)	AEK$LCD-DT028V1A schematic diagrams	AEK$LCD-DT028V1A bill of materials

1. *This code identifies the AEK-LCD-DT028V1 evaluation board first version.*

7 Regulatory compliance information

Formal Notice Required by the U.S. Federal Communications Commission

FCC NOTICE:

This kit is designed to allow:

(1) Product developers to evaluate electronic components, circuitry, or software associated with the kit to determine whether to incorporate such items in a finished product and

(2) Software developers to write software applications for use with the end product.

This kit is not a finished product and when assembled may not be resold or otherwise marketed unless all required FCC equipment authorizations are first obtained. Operation is subject to the condition that this product not cause harmful interference to licensed radio stations and that this product accept harmful interference. Unless the assembled kit is designed to operate under part 15, part 18 or part 95 of this chapter, the operator of the kit must operate under the authority of an FCC license holder or must secure an experimental authorization under part 5 of this chapter 3.1.2.

The evaluation kit has been designed to comply with part 15 of the FCC Technical Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation.

Standard applied: FCC CFR 47 Part 15 Subpart B. Test method applied: ANSI C63.4 (2014).

Formal Product Notice Required by Industry Canada Innovation, Science and Economic Development

Canada compliance:

For evaluation purposes only. This kit generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to Industry Canada (IC) rules.

À des fins d'évaluation uniquement. Ce kit génère, utilise et peut émettre de l'énergie radiofréquence et n'a pas été testé pour sa conformité aux limites des appareils informatiques conformément aux règles d'Industrie Canada (IC).

This device has been tested with Innovation, Science and Economic Development RSS standards. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Standard applied: ICES-003 Issue 7 (2020), Class B. Test method applied: ANSI C63.4 (2014).

Cet appareil a été testé pour les normes RSS d'Innovation, Science et Développement économique. L'utilisation est soumise aux deux conditions suivantes: (1) cet appareil ne doit pas causer d'interférences nuisibles, et (2) cet appareil doit accepter de recevoir tous les types d’interférence, y comprises les interférences susceptibles d'entraîner un fonctionnement indésirable.

Norme appliquée: NMB-003, 7e édition (2020), Classe B. Méthode d'essai appliquée: ANSI C63.4 (2014).

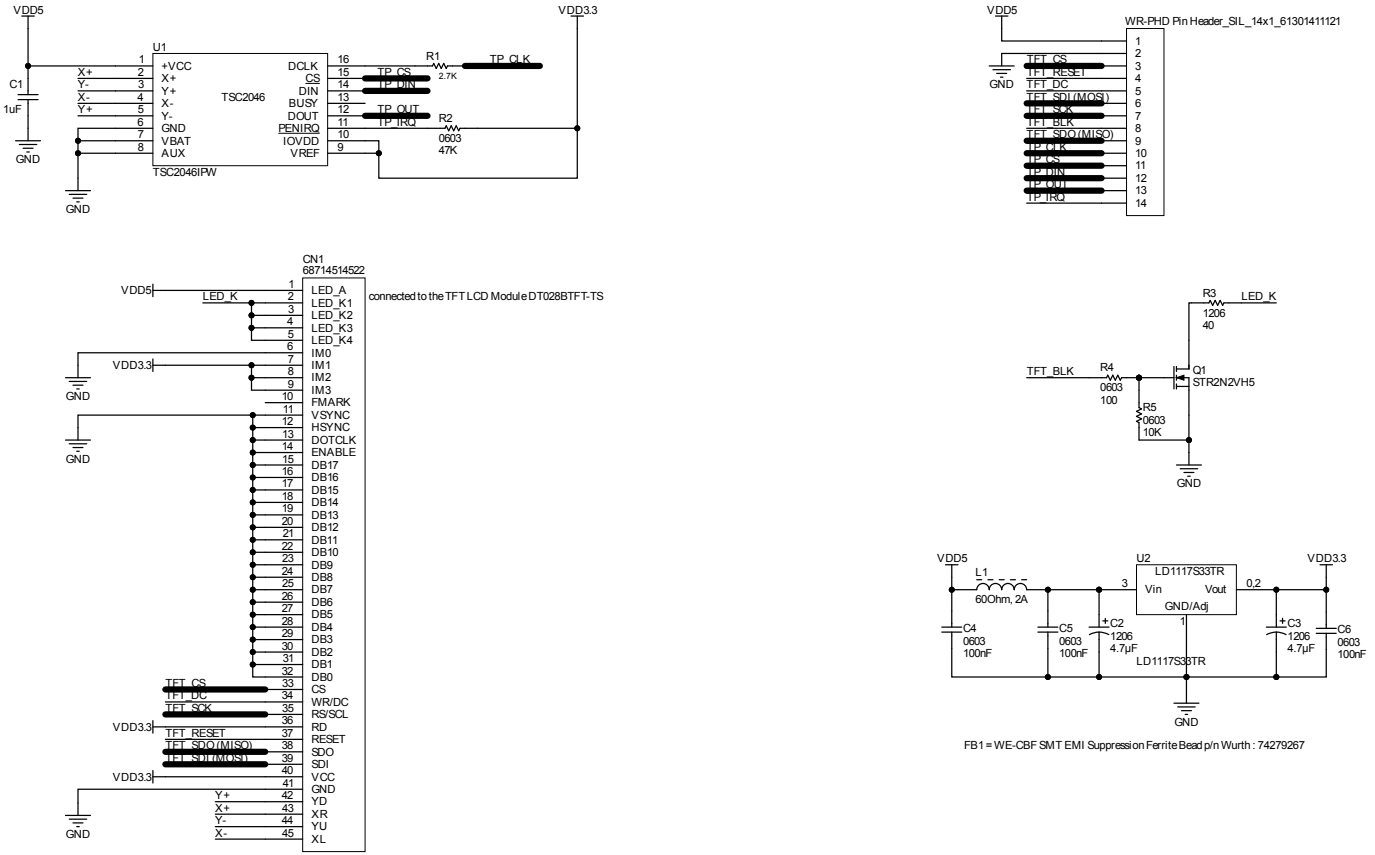
Formal product notice required by EU

This device is in conformity with the essential requirements of the Directive 2014/30/EU (EMC) and of the Directive 2015/863/EU (RoHS).

Standards applied: EN 61000-6-1:2019, EN 61000-6-3:2021, EN 55032:2015 + A1:2020, EN 55035:2017 + A11:2020, EN 61000-3-2:2019, EN 61000-3-3:2013 + A1:2019.

8 Schematic diagrams

Figure 19. **AEK-LCD-DT028V1 circuit schematic**



9 Bill of materials

Table 3. **AEK-LCD-DT028V1 bill of materials**

Item	Q.ty	Ref.	Part/Value	Description	Manufacturer	Order code
1	1	C1	1uF, 0805, 25V, +/-10%	SMD ceramic capacitor	TDK	CGA4J3X7R1H105K125AB
2	2	C2, C3	4.7µF, 1206, 25V, +/-10%	SMD polarized capacitor	Kemet	T491A475K025AT
3	3	C4, C5, C6	100nF, 0603, 50V, +/-10%	SMD ceramic capacitor	TDK	CGA3E2X7R1H104K080AA
4	1	CN1	FPC 0.50 mm, 68714514522	WR-FPC SMT ZIF horizontal bottom contact	Wurth Elektronik	68714514522
5	1	L1	60Ohm, 0603, 2A	EMI suppression ferrite	Wurth Elektronik	74279267
6	1	P1	Pin Header_2.54 mm, SIL_14x1	WR-PHD Pin Header_SIL_14x1_61301411121	Wurth Elektronik	61301411121
7	1	Q1	STR2N2VH5, SOT-23	N-channel 20 V, 0.025 Ohm typ., 2.3 A STripFET H5 power MOSFET in SOT-23 package	ST	STR2N2VH5
8	1	R1	2.7K, 0402, 1/16W, +/-1%	SMD RESISTOR	Vishay	CRCW04022K70FKED
9	1	R2	47K, 0603, 1/3W, +/-1%	SMD resistor	Vishay	CRCW060347K0FKEAHP
10	1	R3	40R, 1206, 1W, +/- 1%	SMD resistor	Vishay	PHP01206E40R2BST3
11	1	R4	100R, 0603, 1/10W, +/-5%	SMD resistor	Vishay	CRCW0603100RJNEA
12	1	R5	10K, 0603, 1/10W, +/-1%	SMD resistor	Vishay	CRCW060310K0FKTA
13	1	U1	TSC2046IPW, TSSOP-16	Touch screen controller	Texas Instruments	TSC2046IPW
14	1	U2	LD1117S33TR, SOT-223	Adjustable and fixed low drop positive voltage regulator	ST	LD1117S33TR
15	1	DISP1	TFT LCD Module DT028BTFT-TS	TFT LCD module + touch screen	Displaytech	DT028BTFT-TS

Revision history

Table 4. **Document revision history**

Date	Revision	Changes
21-Sep-2021	1	Initial release.

Contents

List of tables

List of figures