

การเขียนโปรแกรมแสดงรูปภาพ บนหน้าจอแบบกราฟิก LCD สี ขนาด 1.8 นิ้ว 128×160 พิกเซล



**ของแผงวงจร
ATX-2, POP-7, POP-X2 หรือ IPST-SE**

1. บทนำ

โดยทั่วไปคุณสมบัติมาตรฐานขนาดจอแบบกราฟิกสีของแผงวงจร IPST-SE, POP-X2, POP-7 หรือ ATX2 มีลักษณะแบบเดียวกัน นั่นคือ มีขนาด 1.8 นิ้ว ความละเอียด 128×160 พิกเซล

ภายในไลบรารีการแสดงผลตัวอักษรและกราฟิกมีคำสั่งพื้นฐานต่างๆ ให้ใช้ ดังนี้

<code>glcd</code>	แสดงตัวอักษรที่หน้าจอ
<code>setTextColor</code>	กำหนดสีตัวอักษร
<code>setTextBackgroundColor</code>	กำหนดสีพื้นหลังตัวอักษร
<code>setTextSize</code>	กำหนดขนาดตัวอักษร
<code>glcdClear</code>	เคลียร์หน้าจอ
<code>glcdFillScreen</code>	ถมสีหน้าจอ
<code>glcdMode</code>	กำหนดทิศทางการแสดงข้อความ
<code>glcdPixel</code>	แสดงเม็ดพิกเซลในตำแหน่งที่กำหนด
<code>glcdRect</code>	แสดงรูปสี่เหลี่ยม
<code>glcdFillRect</code>	แสดงรูปสี่เหลี่ยมแบบถมสี
<code>glcdLine</code>	แสดงเส้น
<code>glcdCircle</code>	แสดงวงกลม
<code>glcdFillCircle</code>	แสดงวงกลมแบบถมสี
<code>glcdArc</code>	แสดงส่วนของเส้นโค้ง

แต่คำสั่งทั้งหมดที่กล่าวมา แสดงได้เฉพาะข้อความที่เป็นภาษาอังกฤษ และกราฟิกพื้นฐาน

2. การแปลงค่าสีรูปภาพเป็น 16 บิต (5-6-5 บิต)

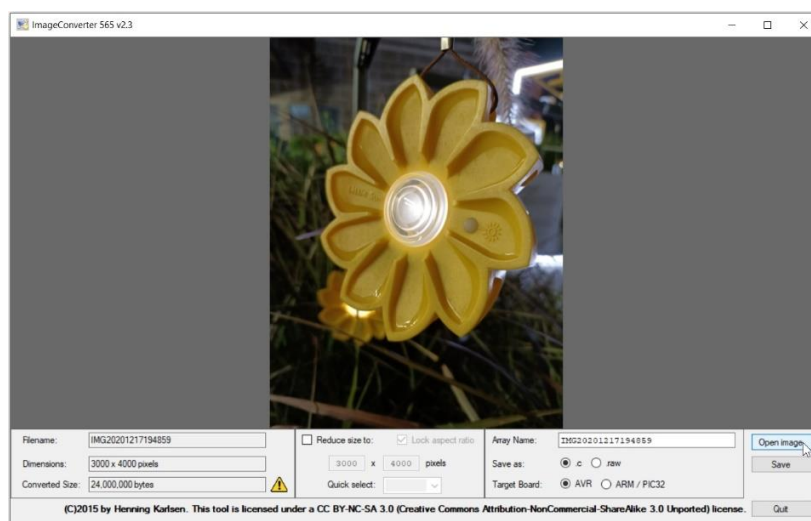
ก่อนอื่นต้องทำความเข้าใจก่อนว่า หน้าจอแบบกราฟิก LCD สีของแผงวงจร INEX แสดงกราฟฟิคลายเส้นและพื้นที่ 65,536 สี ซึ่งกำหนดการเรียงบิตข้อมูลสีให้เป็นแบบ BGR (5-6-5) นั่นคือค่าของสีน้ำเงิน 5 บิต ต่อด้วยสีเขียว 6 บิต และปิดท้ายด้วยค่าของสีแดง 5 บิต ทั้งนี้เนื่องจากผู้ผลิตจอแสดงผลกราฟิก LCD สีมีการผลิตจอแสดงผลแบบนี้มี 2 รุ่น โดยมีการเรียงบิตข้อมูลสีแบบ BGR และแบบ RGB

หากผู้ใช้งานแผงวงจร ATX-2, POP-7, POP-X2 หรือ IPST-SE และทดลองกำหนดสีของภาพหรือตัวอักษรแล้วพบว่า สีที่ได้ไม่ถูกต้อง จะต้องเรียกใช้ฟังก์ชัน `glcdSetColorWordRGB()` ; โดยบรรจุไว้ใน `setup()` ที่ตอนต้นของโปรแกรม

รูปภาพส่วนใหญ่จะมีค่าสีเป็น 24 บิต (8-8-8 บิต) นั่นคือค่าของสีแดง 8 บิต ต่อด้วยสีเขียว 8 บิต และปิดท้ายด้วยค่าของสีน้ำเงิน 8 บิต เราจะต้องแปลงค่าสีรูปภาพให้เป็น 16 บิตก่อน (5-6-5 บิต) โดยใช้เครื่องมือการแปลงที่ชื่อว่า ImageConvertor565 (ผนวกในไฟล์เดือร์ไลบารี `glcdBitmap` เรียบร้อยแล้ว)


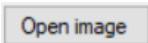
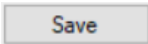
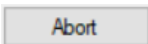
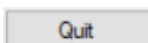
2.1 เครื่องมือการแปลงค่าสีรูปภาพเป็น 16 บิตด้วยโปรแกรม ImageConvertor565

เมื่อดาวน์โหลดเครื่องมือและเปิดโปรแกรมชื่อไฟล์ `ImageConvertor565.exe` ขึ้นมา หน้าตาโปรแกรมมีลักษณะดังนี้



รูปที่ 2-1 หน้าตาโปรแกรมการแปลงค่าสีรูปภาพเป็น 16 บิต ImageConvertor565

2.1.1 องค์ประกอบ

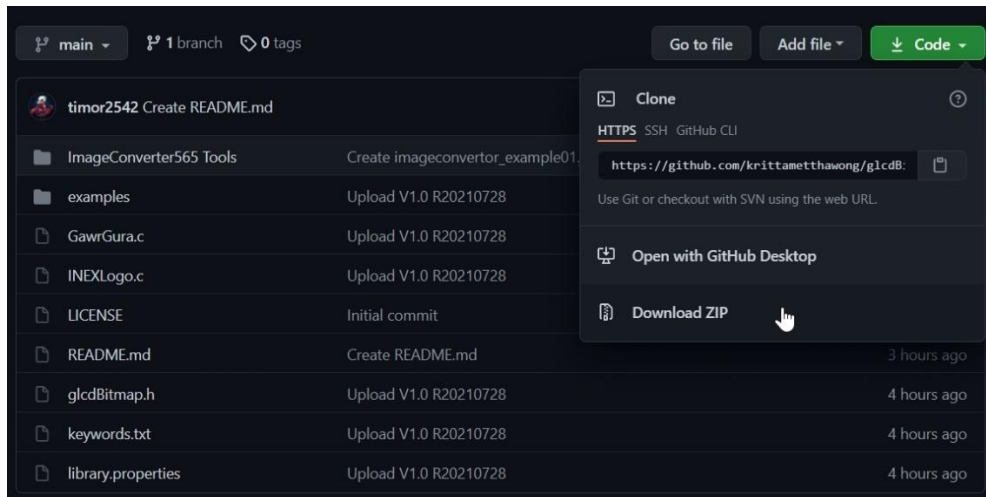
Filename:	เมื่อเปิดไฟล์รูปภาพที่รองรับ จะแสดงชื่อไฟล์บนฟิลด์นี้ (ไม่แสดงประเภทไฟล์)
Dimensions:	แสดงขนาดความกว้างและความสูงของภาพที่เปิดไฟล์ปัจจุบัน ในหน่วยพิกเซล
Converted Size:	เมื่อภาพที่ถูกแปลงจากการปรับขนาดความกว้างและความสูงของภาพ เรียบร้อยแล้ว (ขยาย/ย่อ/คงเดิมขนาดของภาพ) โปรแกรมจะคำนวณและแสดง จำนวนขนาดการเขียนตัวแปรอาร์เรย์ในภาษา C ในหน่วยไบต์อัดโน้มติ
	หากไอคอนตัวนี้ปรากฏขึ้น แสดงว่าหลังจากปรับขนาดความกว้างและความสูง ของภาพ และโปรแกรมได้คำนวณจำนวนขนาดการเขียนตัวแปรอาร์เรย์ใน ภาษา C ออกมามีขนาดเกินหน่วยความจำที่ตัวคอมไพเลอร์ AVR-GCC สามารถคอมไพล์ผ่านได้ (กำหนดขนาดต้องน้อยกว่า 32,768 ไบต์)
Reduce size to:	เลือกช่องนี้เพื่อเปิดใช้งานการปรับลดขนาดความกว้างและความสูงของภาพ
Lock aspect ratio	เมื่อเลือกช่องนี้อัตราส่วนความกว้างต่อความสูงจะถูกล็อก
_____ × _____ pixels	ป้อนขนาดความกว้างและความสูงของภาพที่ต้องการ
Quick Select:	เลือกขนาดความกว้างและความสูงที่ถูกกำหนดไว้แบบเร่งด่วน
Array Name:	ชื่อของอาร์เรย์เมื่อบันทึกเป็นไฟล์อาร์เรย์ .c ชื่อของอาร์เรย์จะถูกกำหนดเป็นชื่อไฟล์ของไฟล์ภาพที่เปิดขึ้นมาเป็นค่าเริ่มต้น ฟิลด์นี้จะถูกปิดใช้งาน เมื่อเลือกตัวเลือกการบันทึกเป็นประเภทไฟล์ .raw
Save as:	เลือกประเภทของไฟล์ที่ต้องการบันทึก มี 2 ประเภท คือ .c และ .raw
Target Board	เลือกประเภทของบอร์ด มี 2 ประเภท คือ AVR และ ARM/PIC32 ฟิลด์นี้จะถูกปิดใช้งาน เมื่อเลือกตัวเลือกการบันทึกเป็นประเภทไฟล์ .raw
	คลิกปุ่มนี้เพื่อเปิดหน้าต่างหาภาพใหม่สำหรับการแปลง
	คลิกปุ่มนี้เพื่อบันทึกเป็นไฟล์อาร์เรย์ .c ปุ่มยกเลิกจะปรากฏในขณะบันทึก
	คลิกปุ่มนี้เพื่อยกเลิกกระบวนการแปลง/การบันทึกปัจจุบัน ปุ่มนี้จะปรากฏในระหว่างการแปลงและการบันทึกเท่านั้น
	คลิกปุ่มนี้เพื่อออกจากโปรแกรม

2.1.2 ขั้นตอนการแปลงค่าสีรูปภาพ 16 บิตให้เป็นไลบรารี

2.1.2.1 ดาวน์โหลดไลบรารี glcdBitmap และเครื่องมือโดยเข้าลิงก์

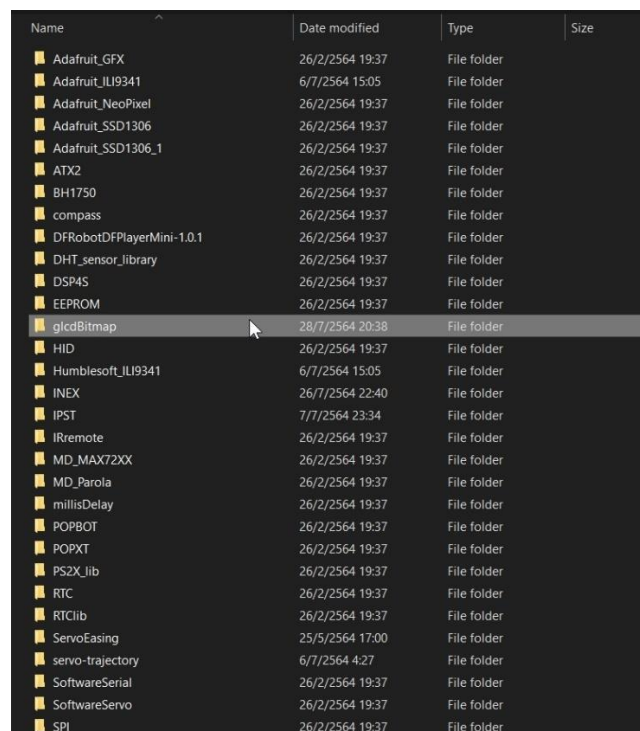
<https://github.com/krittametthawong/glcdBitmap-Library>

จากนั้นคลิกปุ่ม  Code ▾ และเลือก “Download Zip”



2.1.2.2 แยกไฟล์เป็นโฟลเดอร์ตั้งชื่อ glcdBitmap ไว้ในโฟลเดอร์

C:\Arduino18\hardware\INEX_AVR\avr\libraries\

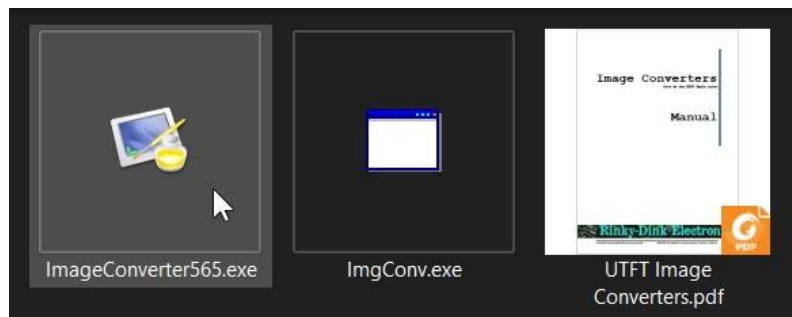


6 • การเขียนโปรแกรมแสดงรูปภาพบนหน้าจอแบบกราฟิก LCD สีของแผงวงจร INEX

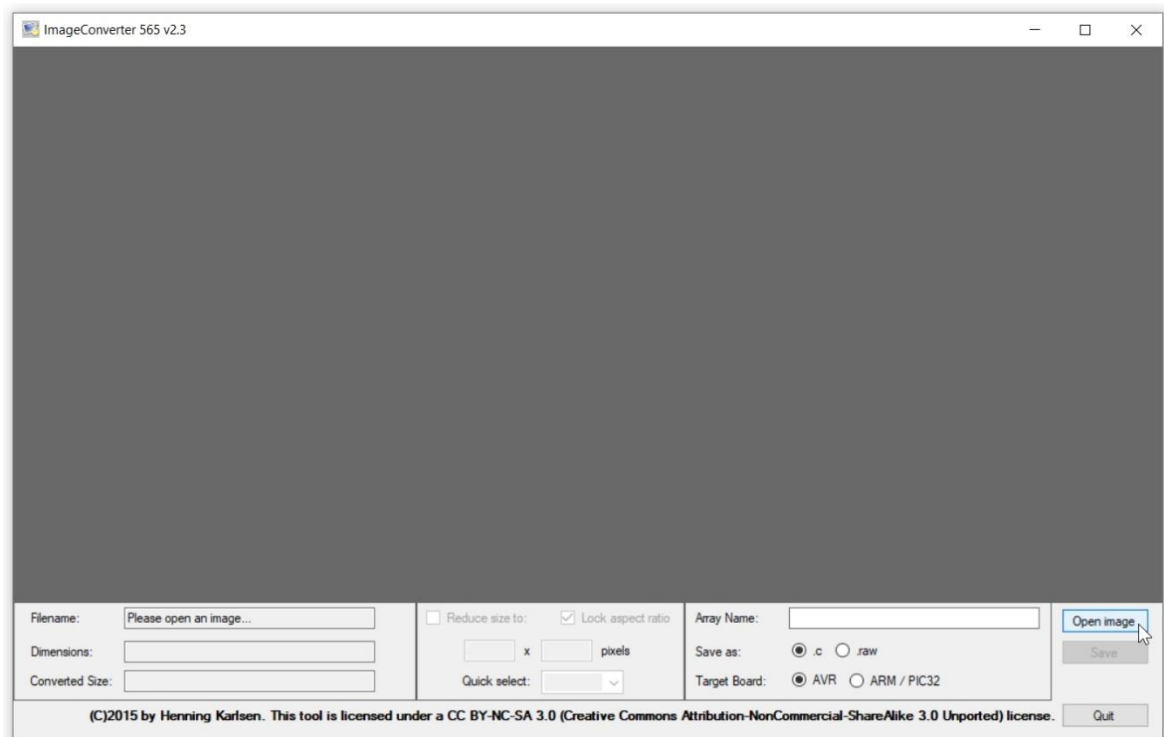
2.1.2.3 เข้าไปในโฟลเดอร์ glcdBitmap จากนั้นเข้าโฟลเดอร์ ImageConverter565 Tools

examples	27/7/2564 1:16	File folder	
ImageConverter565 Tools	29/7/2564 1:07	File folder	
GawrGura.c	27/7/2564 1:40	C File	144 KB
glcdBitmap.h	28/7/2564 20:24	H File	2 KB
INEXLogo.c	27/7/2564 1:33	C File	110 KB
keywords.txt	28/7/2564 20:36	Text Document	1 KB
library.properties	28/7/2564 20:39	PROPERTIES File	1 KB
LICENSE	26/7/2564 22:34	File	35 KB
README.md	28/7/2564 23:03	MD File	1 KB

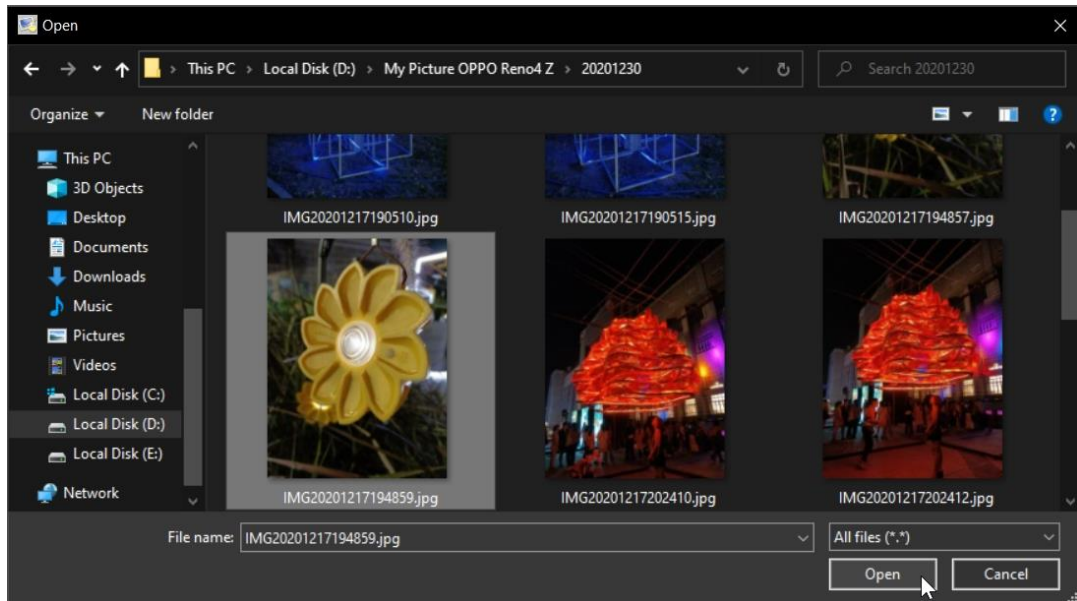
2.1.2.4 ดับเบิ้ลคลิกชื่อไฟล์ ImageConverter565.exe เพื่อเปิดโปรแกรมขึ้นมา (หากหน้าต่างโปรแกรมไม่ปรากฏขึ้น ให้ลองดับเบิ้ลคลิกอีกครั้ง)



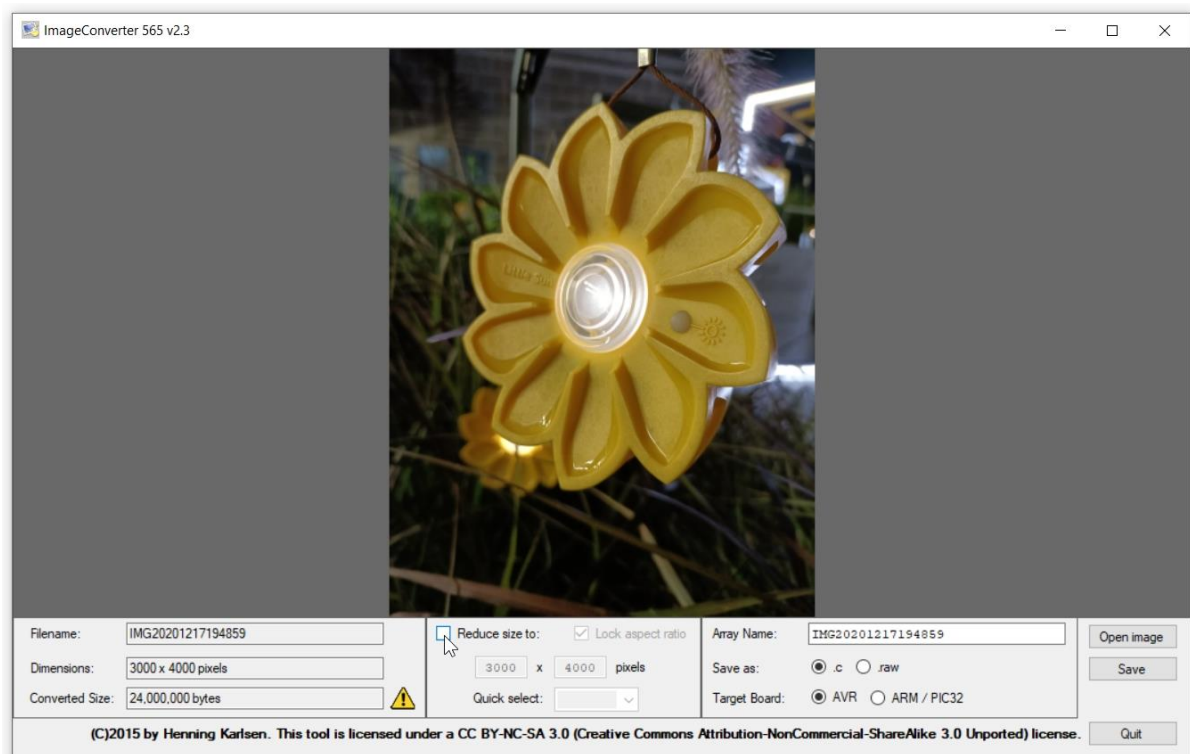
2.1.2.5 คลิก **Open image** เพื่อเปิดหน้าต่างและหารูปภาพที่ต้องการแปลง



2.1.2.6 ค้นหาภาพที่ต้องการแปลง เมื่อเลือกภาพเรียบร้อยแล้ว กดปุ่ม Open เพื่อรับภาพเข้าไปโปรแกรม

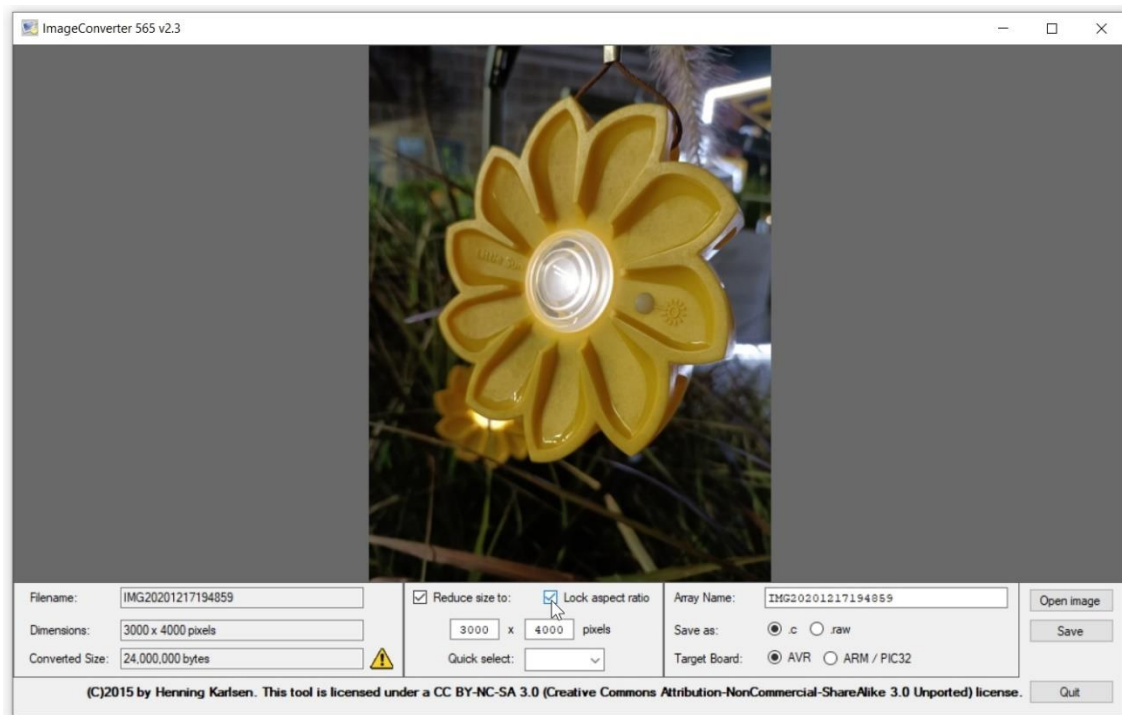


2.1.2.7 สังเกตตรงที่ฟิลด์ Converted Size โปรแกรมได้คำนวณจำนวนขนาดการเขียนตัวแปรอาร์เรย์ในภาษา C ออกมา มีขนาดเกินกำหนดที่ต้องน้อยกว่า 32,768 ไบต์ (ในตัวอย่าง: มีขนาด 24,000,000 ไบต์) ต้องลดขนาดของภาพลง โดยคลิกช่องถูกต้องที่ Reduce size to: เพื่อเปิดฟิลด์การประกอบลดขนาดของภาพ

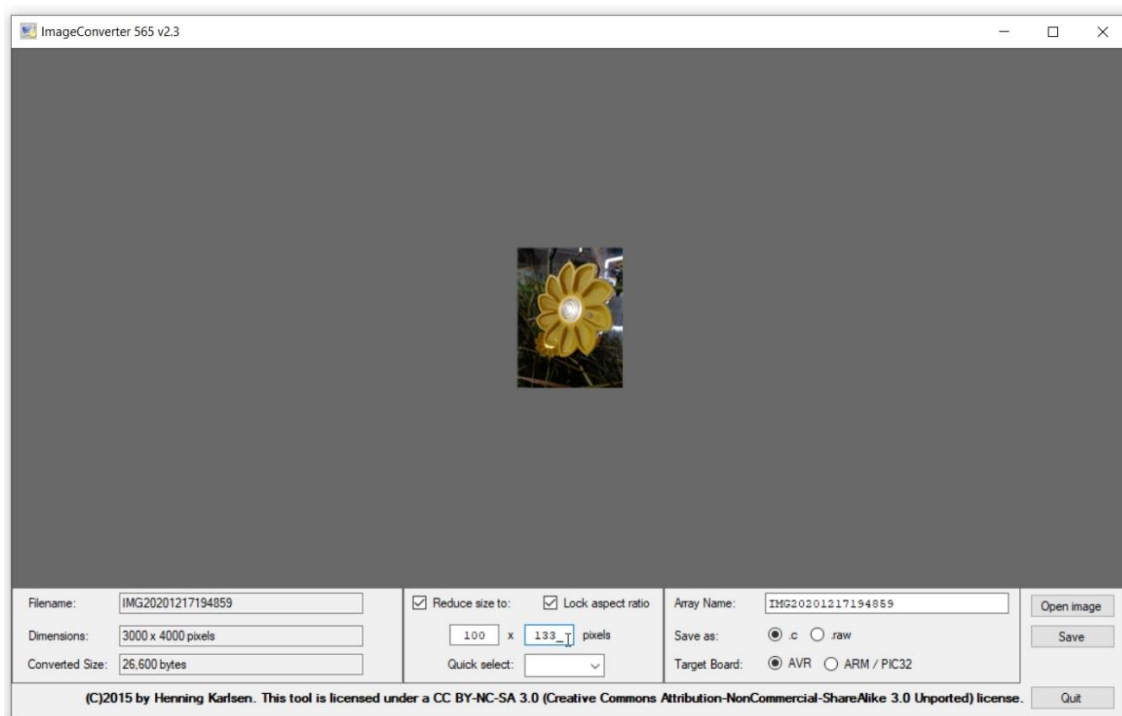


8 • การเขียนโปรแกรมแสดงรูปภาพบนหน้าจอแบบกราฟิก LCD สีของแผงวงจร INEX

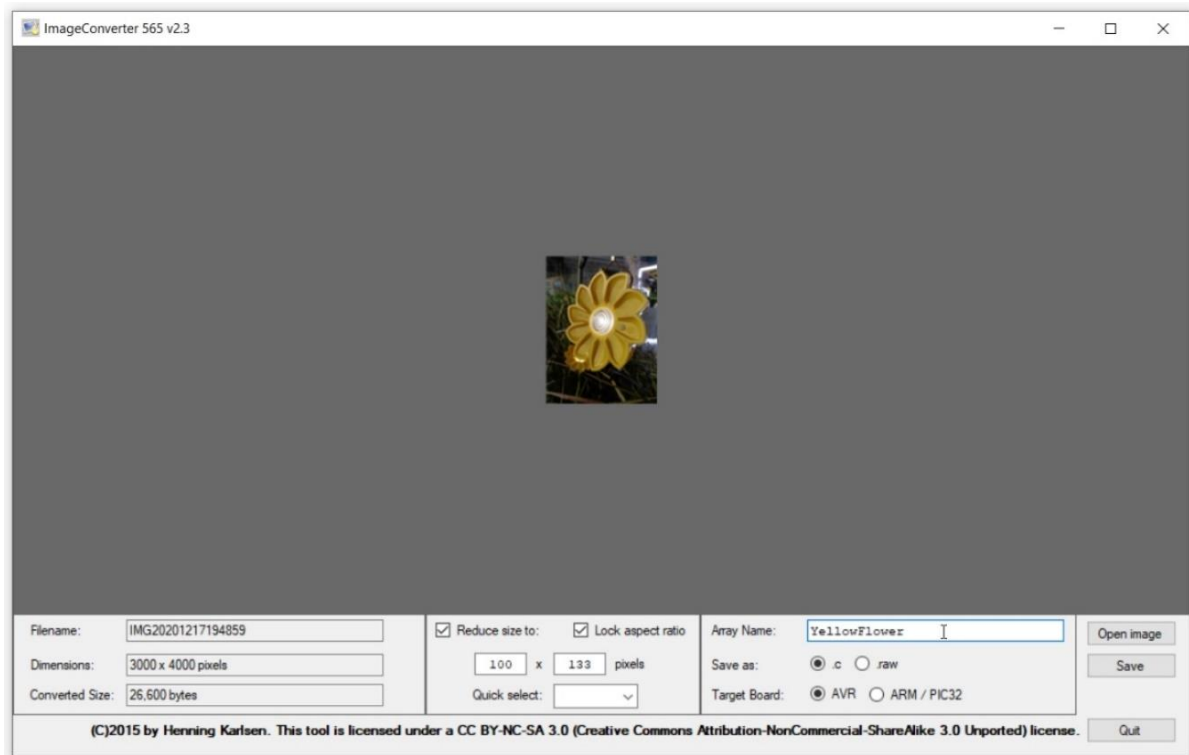
กรณีช่อง Lock aspect ratio ยังไม่ได้ถูกเลือก คลิกให้มีเครื่องหมายถูกต้องปรากฏขึ้นมา เพื่อให้ภาพมีความสัดส่วน



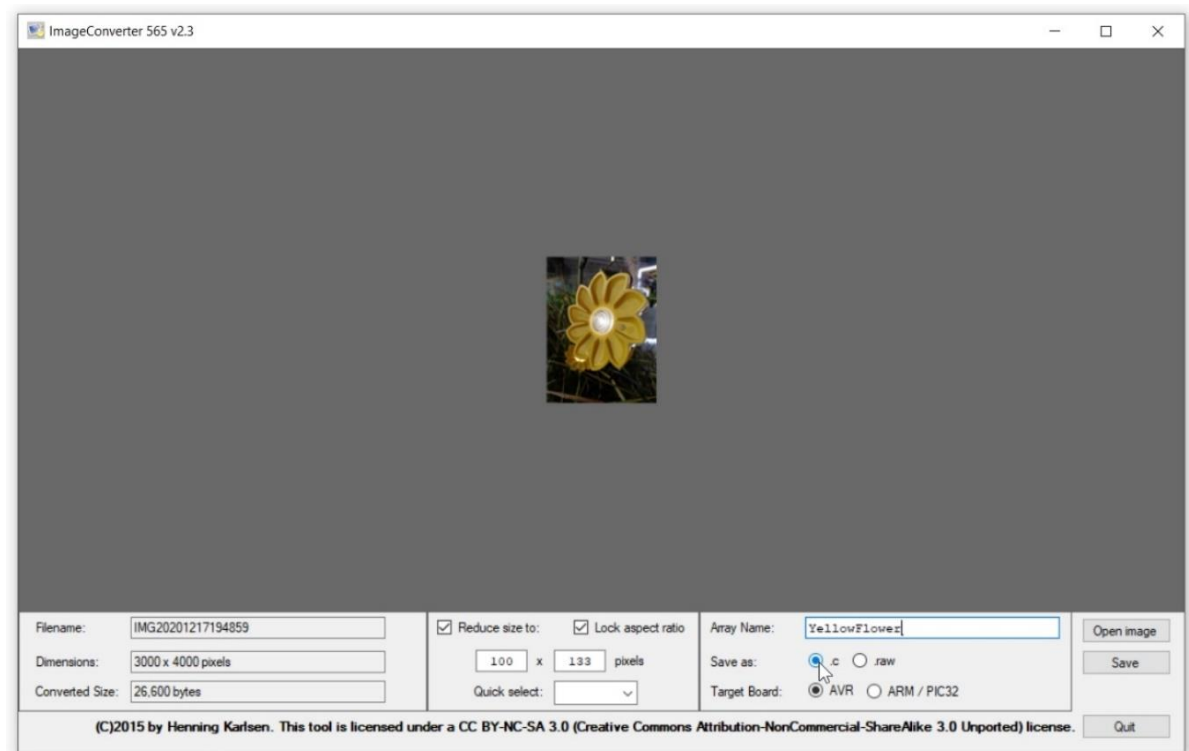
2.1.2.8 จากนั้นลดขนาดของภาพลง (ในตัวอย่าง: ลดขนาดของภาพเหลือ 100×133 พิกเซล)
ให้โปรแกรมคำนวณจำนวนขนาดการเขียนตัวแปรอาร์เรย์ในภาษา C ออกมาต้องมีขนาดน้อยกว่า
32,768 ไบต์ (ในตัวอย่าง: มีขนาด 26,600 ไบต์)



2.1.2.9 ตั้งชื่อตัวแปรอาร์เรย์ลงในฟิลด์ Array Name: (ในตัวอย่าง: YellowFlower)

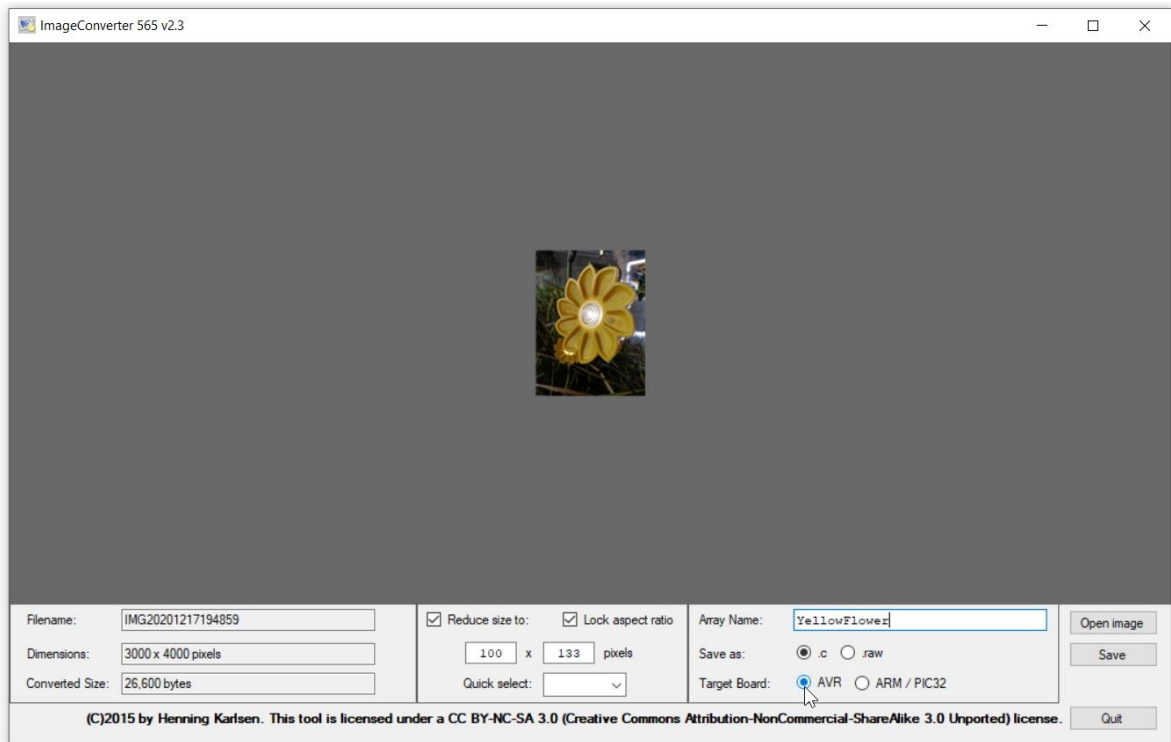


2.1.2.10 เลือกประเภทบันทึกไฟล์ไบนารีเป็น .c



10 ● การเขียนโปรแกรมแสดงรูปภาพบนหน้าจอบนบอร์ดไมโครคอนโทรลเลอร์ LCD สีของแผงวงจร INEX

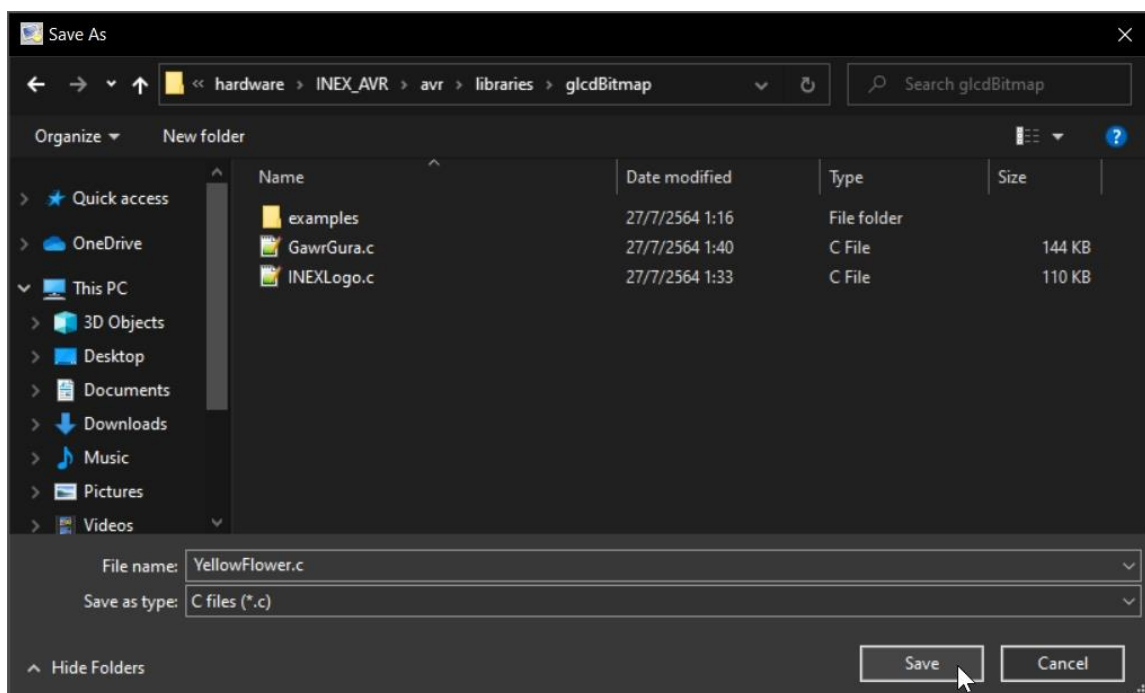
2.1.2.11 เลือกประเภทบอร์ดเป็น AVR



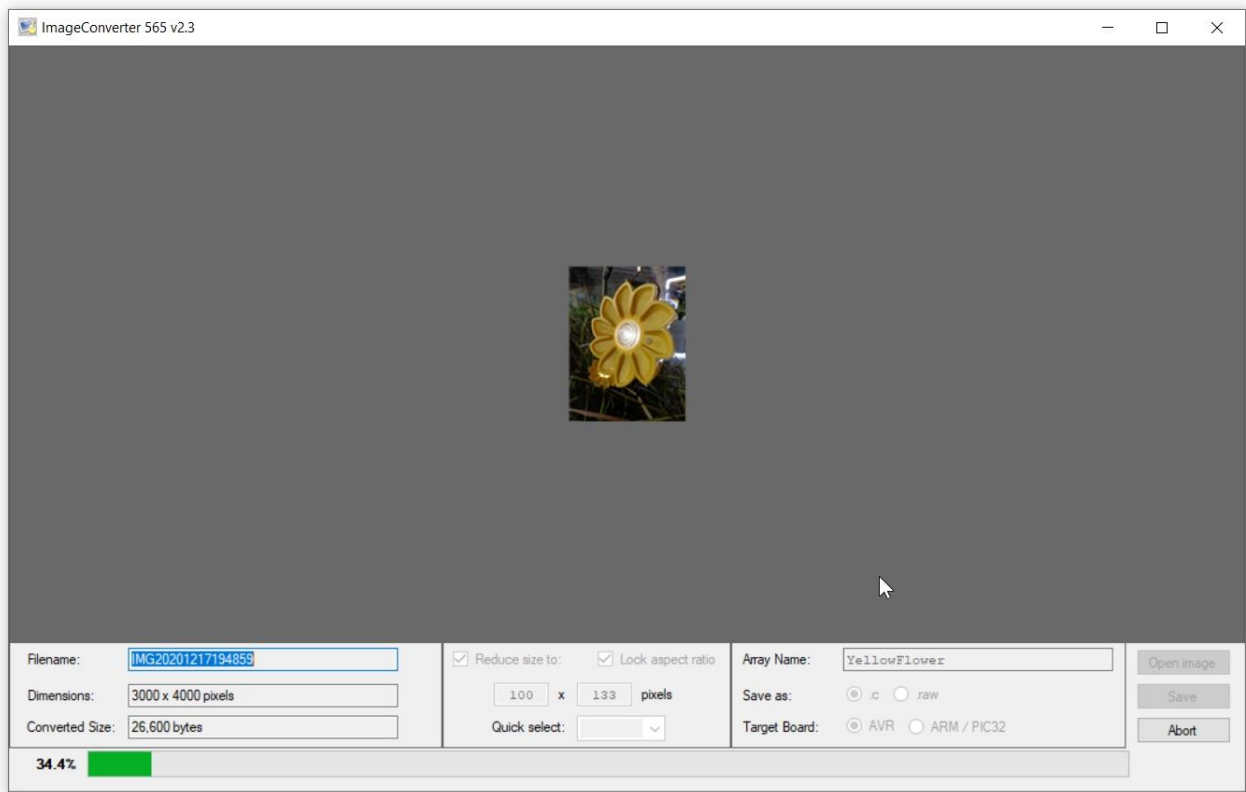
2.1.2.12 คลิก เพื่อบันทึกไฟล์ไลบรารีอาร์เลย์ .c ไว้ในโฟลเดอร์

C:\Arduino18\hardware\INEX_AVR\avr\libraries\glcdBitmap\

เพื่อให้ง่ายกับการบันทึก ให้ตั้งชื่อไฟล์เดียวกับชื่อตัวแปรอาร์เลย์ (ในตัวอย่าง: YellowFlower.c)



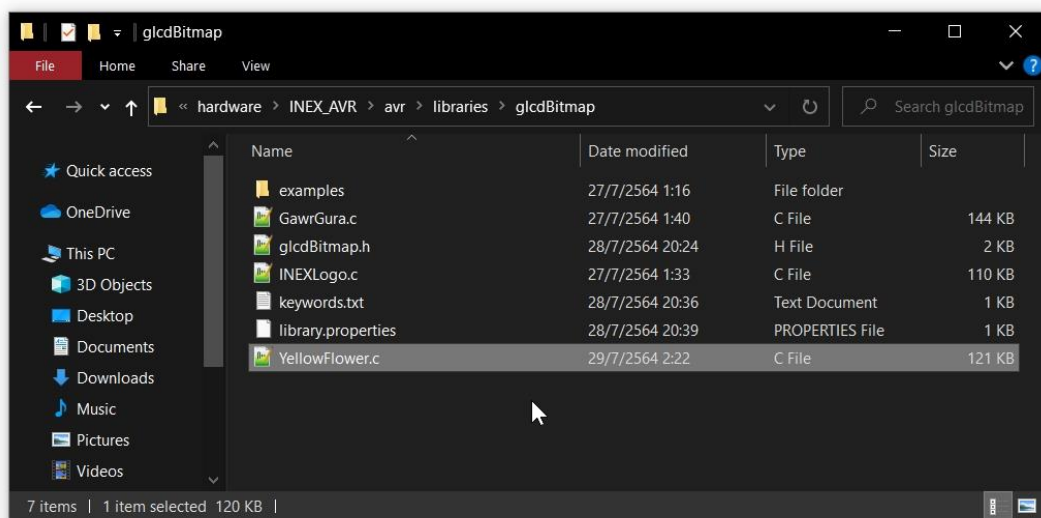
2.1.2.13 รอสักครู่ โปรแกรมกำลังประมวลผลการแปลงภาพให้อยู่รูปแบบตัวแปรอาร์เลย์ลงในไลบรารี



2.1.2.14 เมื่อโปรแกรมประมวลผลการแปลงภาพเสร็จสิ้น ให้ลองเข้าไปดูในโฟลเดอร์

C:\Arduino18\hardware\INEX_AVR\avr\libraries\glcdBitmap\

จะมีไฟล์ที่บันทึกจากการแปลงภาพมาให้เรียบร้อยแล้ว (ในตัวอย่าง: YellowFlower.c)



3. ไลบรารี glcdBitmap

Bitmap Graphics Library for INEX Microcontroller

ไลบรารีนี้สร้างเพื่อวาดกราฟิกรูปภาพบิตแมป (Bitmap) โดยต้องใช้เครื่องมือการแปลงค่าสีรูปภาพเป็น 16 บิต ImageConvertor565 (ผนวกในไฟล์เดอร์ไลบรารี glcdBitmap เรียบร้อยแล้ว)

ไลบรารีตัวนี้สามารถใช้งานได้กับแผงวงจร IPST-SE, POP-X2, POP-7 หรือ ATX2

3.1 glcdBitmap

เป็นคำสั่งแสดงภาพบิตแมป (Bitmap) โดยกำหนดจุดเริ่มต้น เรียกชื่อตัวแปรอาร์เลย์ภาพบิตแมปสี 16 บิตและขนาดของภาพ ซึ่งต้องกำหนดขนาดภาพที่ได้จากการแปลงให้ถูกต้องรูปแบบ

```
void glcdBitmap(int16_t x, int16_t y, const uint16_t bitmap[], int16_t w, int16_t h)
```

พารามิเตอร์

x คือ ค่าตำแหน่งเริ่มต้นของภาพบิตแมปในแกน x มีค่าระหว่าง 0 ถึง 127

y คือ ค่าตำแหน่งเริ่มต้นของภาพบิตแมปในแกน y มีค่าระหว่าง 0 ถึง 127

bitmap[] คือ การเรียกชื่อตัวแปรอาร์เลย์ภาพบิตแมปสี 16 บิต

w* คือ ค่าความกว้างของภาพบิตแมป (แนวนอน) มีค่าระหว่าง 1 ถึง 128

h* คือ ค่าความสูงของภาพบิตแมป (แนวตั้ง) มีค่าระหว่าง 1 ถึง 128

หมายเหตุ * ขนาดภาพที่ได้จากการแปลงจะต้องกำหนดให้ถูกต้อง

ตัวอย่างที่ 3-1

```
#include <ipst.h>           // ผนวกไลบรารีแผงวงจรหลัก IPST-SE
#include "glcdBitmap.h"     // ผนวกไลบรารีคำสั่งแสดงภาพบิตแมป
#include "INEXLogo.c"       // ผนวกไลบรารีอาร์เลย์ภาพบิตแมปโลโก้ INEX (ติดตั้งมาให้อยู่แล้ว)
```

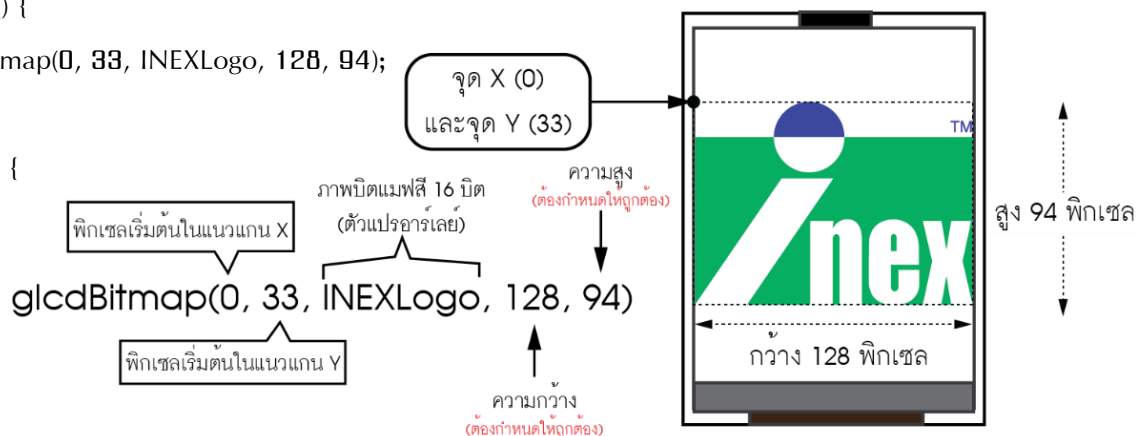
```
void setup() {
```

```
    glcdBitmap(0, 33, INEXLogo, 128, 94);
```

```
}
```

```
void loop() {
```

```
}
```



4. ปฏิบัติการเขียนโปรแกรมบนซอฟต์แวร์ Arduino IDE

ปฏิบัติการที่ 4-1 แสดงกราฟิกภาพบิตแมปจากการแปลง

เป็นการทดลองเขียนโปรแกรมภาษา C เพื่อแสดงกราฟิกภาพบิตแมป (อ้างอิงการใช้ภาพจากการแปลงมาจากข้อ 2. และใช้แผงวงจรควบคุมหลัก IPST-SE เป็นตัวอย่างการทำ)

ขั้นตอนการทดลอง

4.1.1 เปิดซอฟต์แวร์ Arduino IDE ขึ้นมาสร้างไฟล์ พิมพ์โปรแกรมที่ 4-1 บันทึกในชื่อ

IPST_glcdBitmapTest.ino


4.1.2 เขียนโปรแกรม

```
#include <ipst.h>           // ผนวกไลบรารีแผงวงจรหลัก IPST-SE
#include "glcdBitmap.h"     // ผนวกไลบรารีคำสั่งแสดงภาพบิตแมป
#include "YellowFlower.c"   // ผนวกไลบรารีอาร์เรย์ภาพบิตแมปจากการแปลง

void setup() {
    glcdBitmap(14, 13, YellowFlower, 100, 133);    // แสดงภาพบิตแมปจากการแปลง
}

void loop(){
}
```

โปรแกรมที่ L4-1 ไฟล์ IPST_glcdBitmapTest.ino โปรแกรมภาษา C สำหรับทดลองแสดงกราฟิกภาพบิตแมปจากการแปลง

4.1.3 กดปุ่มไฟลล์และอัปโหลดโปรแกรมไปยังแผงวงจร IPST-SE โดยคลิกที่ปุ่ม  หรือเลือกที่เมนู Files > Upload

14 ● การเขียนโปรแกรมแสดงรูปภาพบนหน้าจอแบบกราฟิก LCD สีของแผงวงจร INEX

4.1.4 สังเกตผลการทำงานบนหน้าจอของแผงวงจรควบคุมหลัก IPST-SE จะได้ภาพbitmap เหมือนกับต้นฉบับที่มีขนาด 100×133 พิกเซล



ปฏิบัติการที่ 4-2 หมุนภาพบิตแมปพร้อมกับคำสั่ง glcdMode

เป็นการทดลองเขียนโปรแกรมภาษา C เพื่อแสดงกราฟิกภาพบิตแมปโลโก้ INEX และทำการกำหนดทิศทางแสดงผลของจอกราฟิก LCD สี หมุนขวาวัดละ 90 องศา ทุกๆ 3 วินาที (ใช้แผงวงจรควบคุมหลัก IPST-SE เป็นตัวอย่างการทำ)

ขั้นตอนการทดลอง

4.2.1 เปิดซอฟต์แวร์ Arduino IDE ขึ้นมาสร้างไฟล์ พิมพ์โปรแกรมที่ 4-2 บันทึกในชื่อ

IPST_glcdBitmapRotationTest.ino

4.2.2 เขียนโปรแกรม


```
#include <ipst.h>           // ผนวกไลบรารีแผงวงจรหลัก IPST-SE
#include "glcdBitmap.h"      // ผนวกไลบรารีคำสั่งแสดงภาพบิตแมป
#include "INEXLogo.c"        // ผนวกไลบรารีอาร์เลย์ภาพบิตแมปโลโก้ INEX (ติดตั้งมาให้แล้ว)

void setup() {}

void loop() {
    glcdClear();             // เคลียร์หน้าจอ
    glcdMode(0);              // โหมดแสดงผลตั้งฉากตรงหน้า (โหมด 0)
    glcdBitmap(0, 33, INEXLogo, 128, 94); // แสดงภาพบิตแมปโลโก้ INEX
    delay(3000);              // หน่วงเวลา 3 วินาที
    glcdClear();
    glcdMode(1);              // โหมดแสดงผลหมุนขวา 90 องศา (โหมด 1)
    glcdBitmap(16, 17, INEXLogo, 128, 94);
    delay(3000);
    glcdClear();
    glcdMode(2);              // โหมดแสดงผลหมุนขวา 180 องศา (โหมด 2)
    glcdBitmap(0, 33, INEXLogo, 128, 94);
    delay(3000);
    glcdClear();
    glcdMode(3);              // โหมดแสดงผลหมุนขวา 270 องศา (โหมด 3)
    glcdBitmap(16, 17, INEXLogo, 128, 94);
    delay(3000);
}
```

โปรแกรมที่ L4-2 ไฟล์ IPST_glcdBitmapRotationTest.ino โปรแกรมภาษา C สำหรับทดลองหมุนภาพบิตแมปพร้อมกับคำสั่ง glcdMode

16 ● การเขียนโปรแกรมแสดงรูปภาพบนหน้าจอแบบกราฟิก LCD สีของแผงวงจร INEX

4.2.3 คอมไพล์และอัปโหลดโปรแกรมไปยังแผงวงจร IPST-SE โดยคลิกที่ปุ่ม  หรือเลือกที่เมนู Files > Upload

4.2.4 สังเกตผลการทำงานบนหน้าจอของแผงวงจรควบคุมหลัก IPST-SE

4.2.4.1 เริ่มต้นการทำงานของหน้าจอกราฟิกสีจะทำการเคลียร์หน้าจอ จากนั้นแสดงภาพบิตแมปโลโก้ INEX ตั้งฉากตรงหน้า (โหมด 0) จนเสร็จ



4.2.4.2 ผ่านไป 3 วินาที หน้าจอกราฟิกสีจะทำการเคลียร์หน้าจอ จากนั้นแสดงภาพบิตแมปโลโก้ INEX หมุนขวา 90 องศา (โหมด 1) จนเสร็จ



4.2.4.3 ผ่านไป 3 วินาที หน้าจอกราฟิกสีจะทำการเคลียร์หน้าจอ จากนั้นแสดงภาพ

บิตแมปโลโก้ INEX หมุนขวา 180 องศาหรือกลับหัว (โหมด 2) จนเสร็จ



4.2.4.4 ผ่านไป 3 วินาที หน้าจอกราฟิกสีจะทำการเคลียร์หน้าจอ จากนั้นแสดงภาพ

บิตแมปโลโก้ INEX หมุนขวา 270 องศา (โหมด 3) จนเสร็จ



4.2.4.5 ผ่านไป 3 วินาที กลับไปทำขั้นตอนที่ 4.2.4.1 วนซ้ำ

เพียงเท่านี้ ผู้ใช้งานสามารถเขียนโปรแกรมแสดงรูปภาพบนหน้าจอแบบกราฟิก LCD ได้ โดยไม่จำเป็นต้องใช้อุปกรณ์อ่าน SD Card จากภายนอกพ่วงเข้ากับแผงวงจร INEX แล้ว