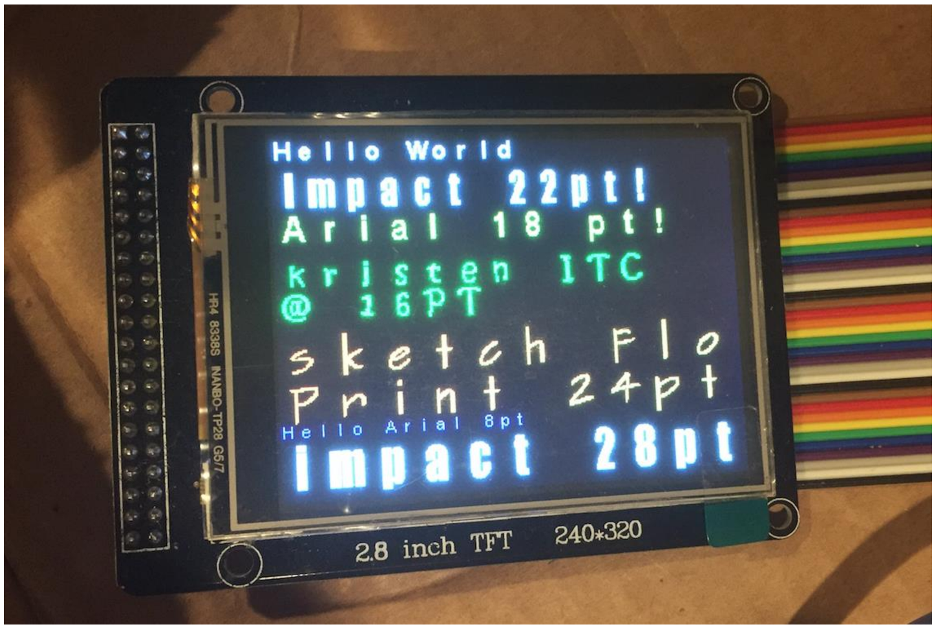
dfUTFTFork By Philip Vallone



This document provides information on the dfUTFTFork library. The dfUTFTFork is a fork of the UTFT Library by Henning Karlsen. The original UTFT library is well written and works with a number of TFT drivers. The UFTF library has a number of limitations when creating fonts. For example, with the original UTFT library, all fonts **MUST** have a character width that is dividable by 8 (i.e. 8, 16, and 24 etc...). You have 2 choices to create fonts; 1, by hand or 2 with the online Font Maker, which converts an image into a font array. Both, are not easy to do and are time consuming.

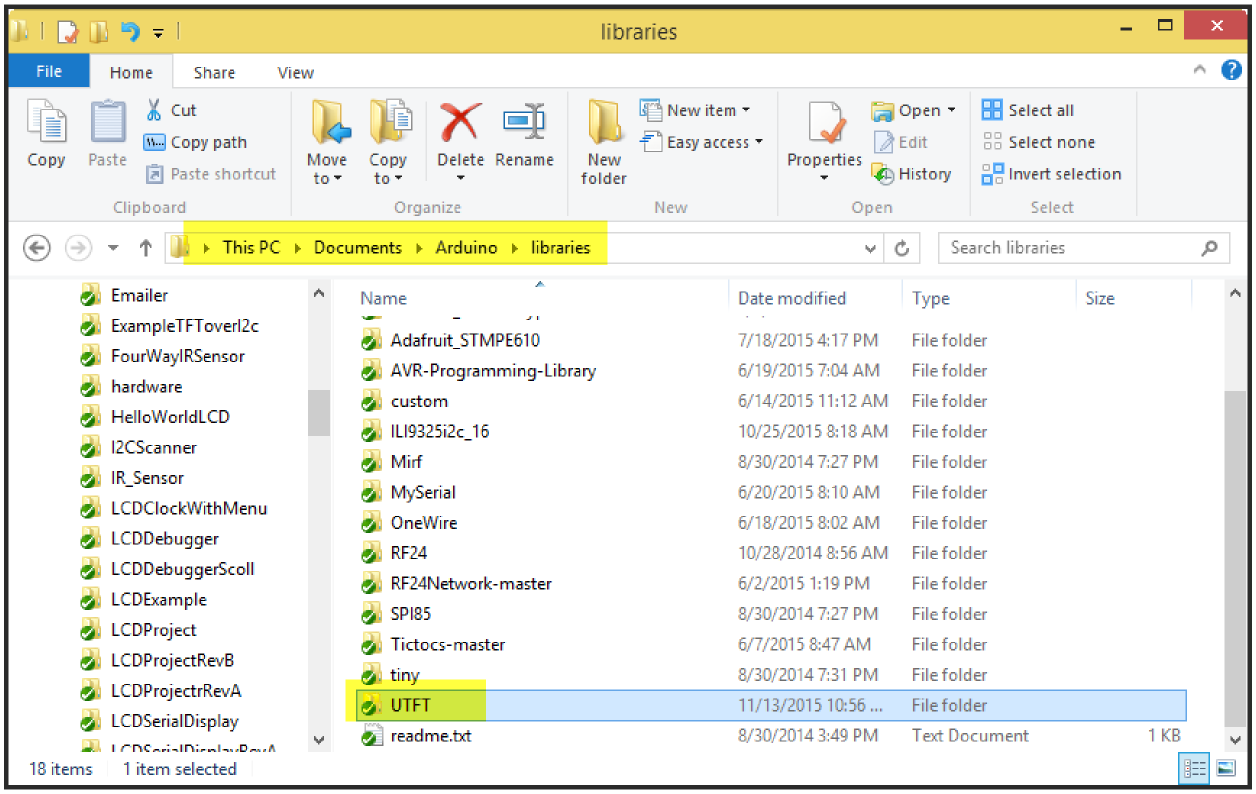
So I set out to modify the UTFT library with work with The Dot Factory. “The Dot Factory is a small, GPL, open source tool intended to generate the required C language information to store many fonts and images, as efficiently as possible, on a microcontroller”. Basicly, if you have the font installed on your window PC, you can create a Font to use with the UFTF library.

Important Notes

- The dfTFTFork library can co-exist with existing UTFT fonts. You just call different functions.
- I have tested the library with a number of fonts generated by The Dot Factory. This doc provides instructions on how to create the fonts. You must follow these directions. Don’t worry, its easy
- The Library was tested on the Sain Smart 2.8 TFT W/ ILI9325 driver running in 16 bit mode. I have not tested on other TFTs
- The smallest font I have tested with was Aerial 8pt, which is very small. The width of the characters should be no smaller than 16 bits wide (2 bytes). That’s really small.
- The largest font I tested with was 5 bytes wide (48pt font).
- Rotating font is not supported

Get the Fork

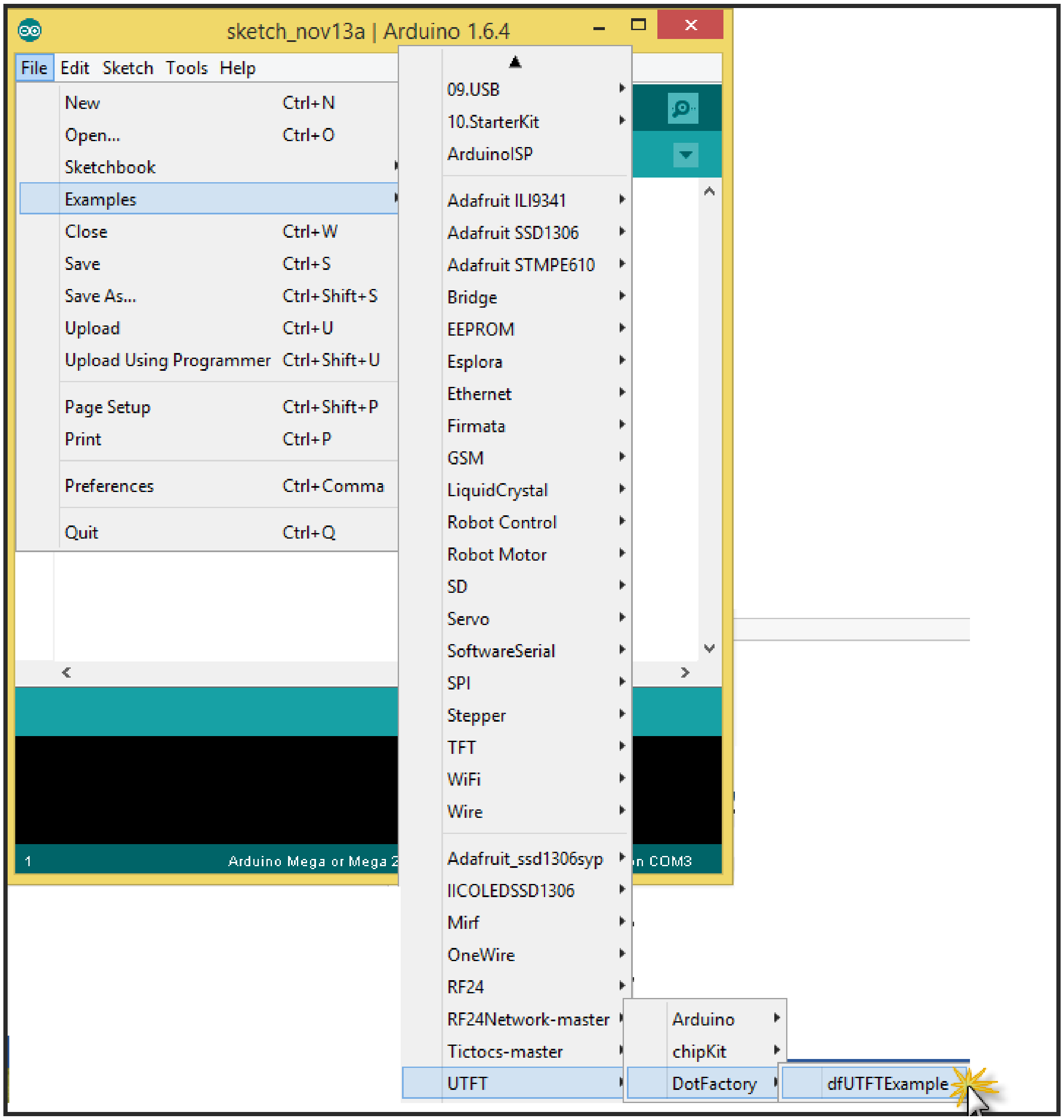
Head over to xx and download the dfUTFTFork. Unzip the file and place the UTFT folder in the libraries directory. For example, my libraries folder is located in C:\Users\Philip\Documents\Arduino



Name	Date modified	Type	Size
Adafruit_STMPE610	7/18/2015 4:17 PM	File folder	
AVR-Programming-Library	6/19/2015 7:04 AM	File folder	
custom	6/14/2015 11:12 AM	File folder	
ILI9325i2c_16	10/25/2015 8:18 AM	File folder	
Mirf	8/30/2014 7:27 PM	File folder	
MySerial	6/20/2015 8:10 AM	File folder	
OneWire	6/18/2015 8:02 AM	File folder	
RF24	10/28/2014 8:56 AM	File folder	
RF24Network-master	6/2/2015 1:19 PM	File folder	
SPI85	8/30/2014 7:27 PM	File folder	
Tictocs-master	6/7/2015 8:47 AM	File folder	
tiny	8/30/2014 7:31 PM	File folder	
UTFT	11/13/2015 10:56 ...	File folder	
readme.txt	8/30/2014 3:49 PM	Text Document	1 KB

Example Sketch:

The dfUTFTFork comes with a sample sketch with a few fonts. Click File>Examples>UTFT>Dot Factory>dfUTFTExample



This library was tested with the ILI9325 Driver on the Sain Smart 2.8 TFT (16 bit mode). You will have to adjust the UTFT constructor to match your TFT/Driver. See your TFT's documentation for your driver and pins:

```
UTFT myGLCD(ILI9325D_16, RS, WR, CS, RST);
```

The example sketch was tested on the Arduino Mega. You can use a Uno, but there is not a lot of memory on the Uno. The fonts consume a lot of space.

How it works

In order to display fonts, we need to know all about the character we are displaying. To do this, we need 3 arrays:

1. Character Map
2. Font Bit Map
3. Font Descriptor

Character Map

Maps the ASCII character to the Font Bit Map and Font Descriptor arrays.

Bit Map

Bit Map array generated by the Dot Factory.

Font Descriptor

Provides needed font information, character width in bits, character height in bits and character offset. The descriptor is generated by the Dot Factory.

Methods

1. include your font and character map

   Example:

   ```
   #include "arcena_22pt.h"
   #include "character_map.h"
   ```

2. Define the font. The setFont functions takes 3 arguments

   ```
   setFont([Bit Map Array],[Font Descriptor], [Number of Characters in Character Map]);
   ```

   Example:

   ```
   myGLCD.setFont(arcena_22ptBitmaps, arcena_22ptDescriptors, 95);
   ```

3. define the character map

   ```
   myGLCD.setCharMap(charMap);
   ```

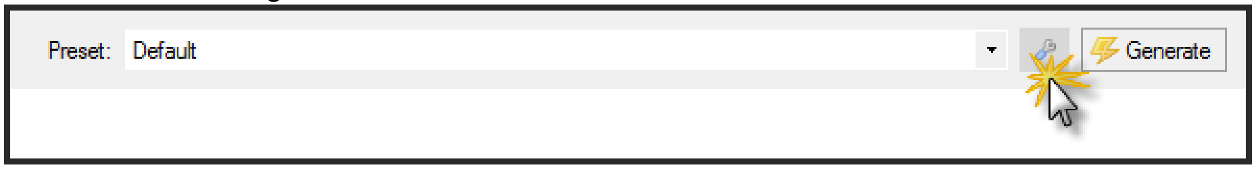
Create Fonts

Creating fonts with The Dot Factory is easy. Download the exe and open the program.

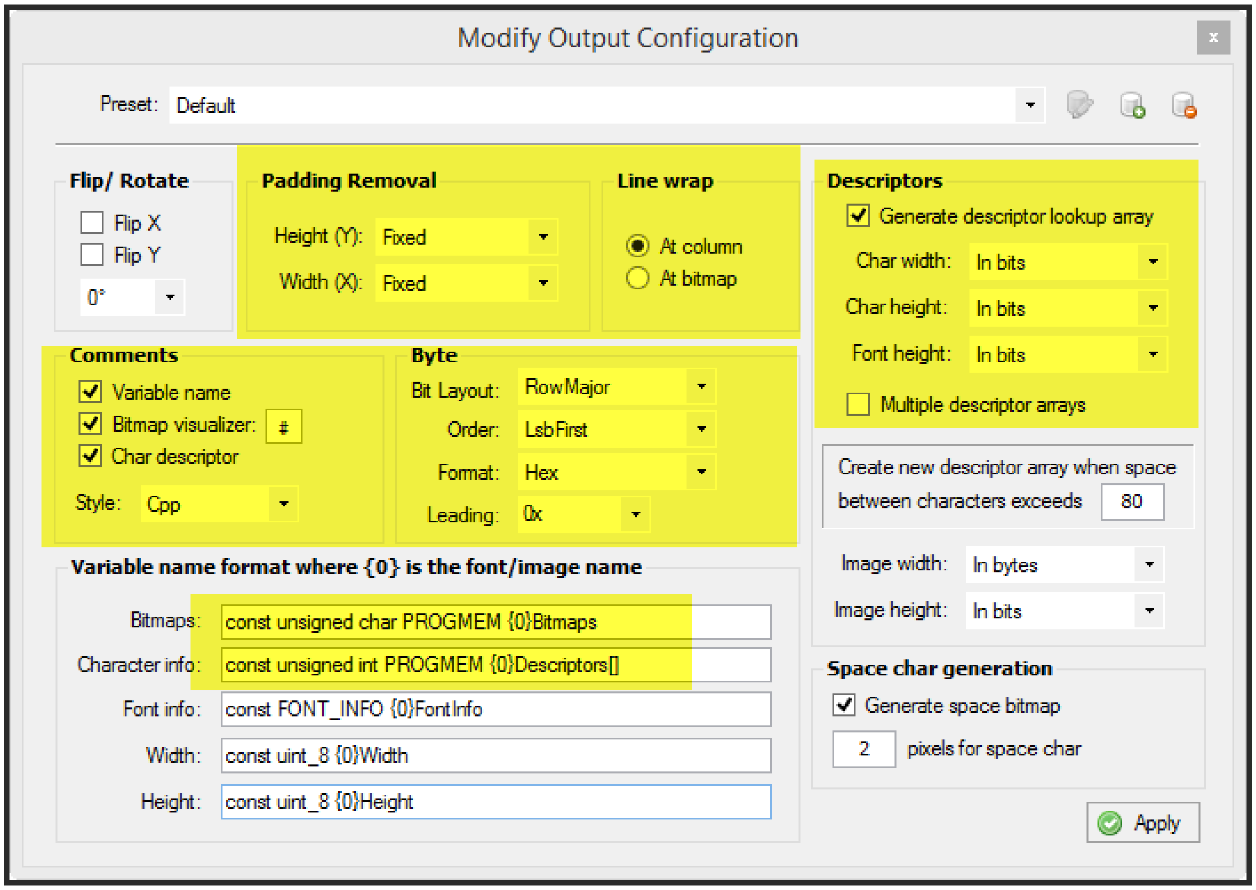
Modify Output Configuration

First thing we need to do is modify the Output Configuration. **If you don't set the configuration as documented below, your fonts will now display.**

1. Click the configuration button



2. Set your configuration as highlighted below. Click Apply



Modify Output Configuration

Preset: Default

Flip/ Rotate
- [ ] Flip X
- [ ] Flip Y
- 0°

Padding Removal
- Height (Y): Fixed
- Width (X): Fixed

Line wrap
- (●) At column
- ( ) At bitmap

Descriptors
- [✓] Generate descriptor lookup array
- Char width: In bits
- Char height: In bits
- Font height: In bits
- [ ] Multiple descriptor arrays
- Create new descriptor array when space between characters exceeds 80
- Image width: In bytes
- Image height: In bits

Comments
- [✓] Variable name
- [✓] Bitmap visualizer: #
- [✓] Char descriptor
- Style: Cpp

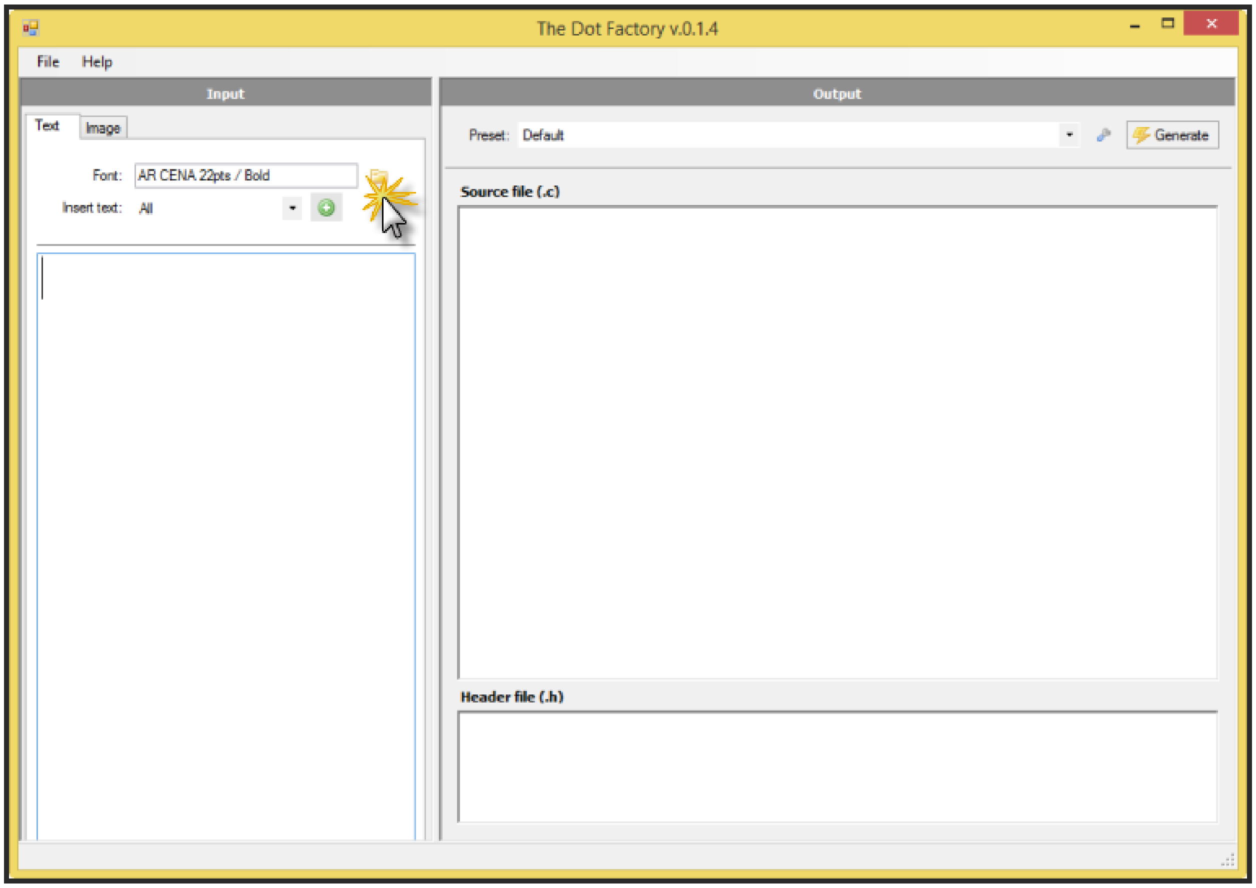
Byte
- Bit Layout: RowMajor
- Order: LsbFirst
- Format: Hex
- Leading: 0x

Variable name format where {0} is the font/image name
- Bitmaps: const unsigned char PROGMEM {0}Bitmaps
- Character info: const unsigned int PROGMEM {0}Descriptors[]
- Font info: const FONT_INFO {0}FontInfo
- Width: const uint_8 {0}Width
- Height: const uint_8 {0}Height

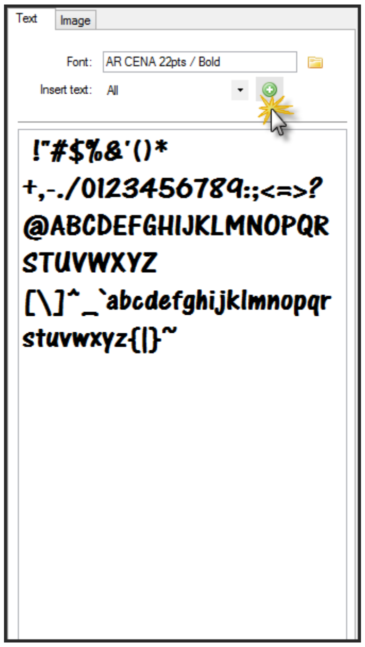
Space char generation
- [✓] Generate space bitmap
- 2 pixels for space char

Apply

3. Click the folder icon to choose your font

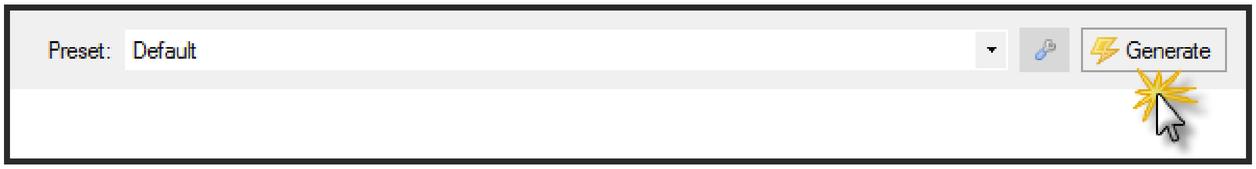


4. Click the + icon to load some default text

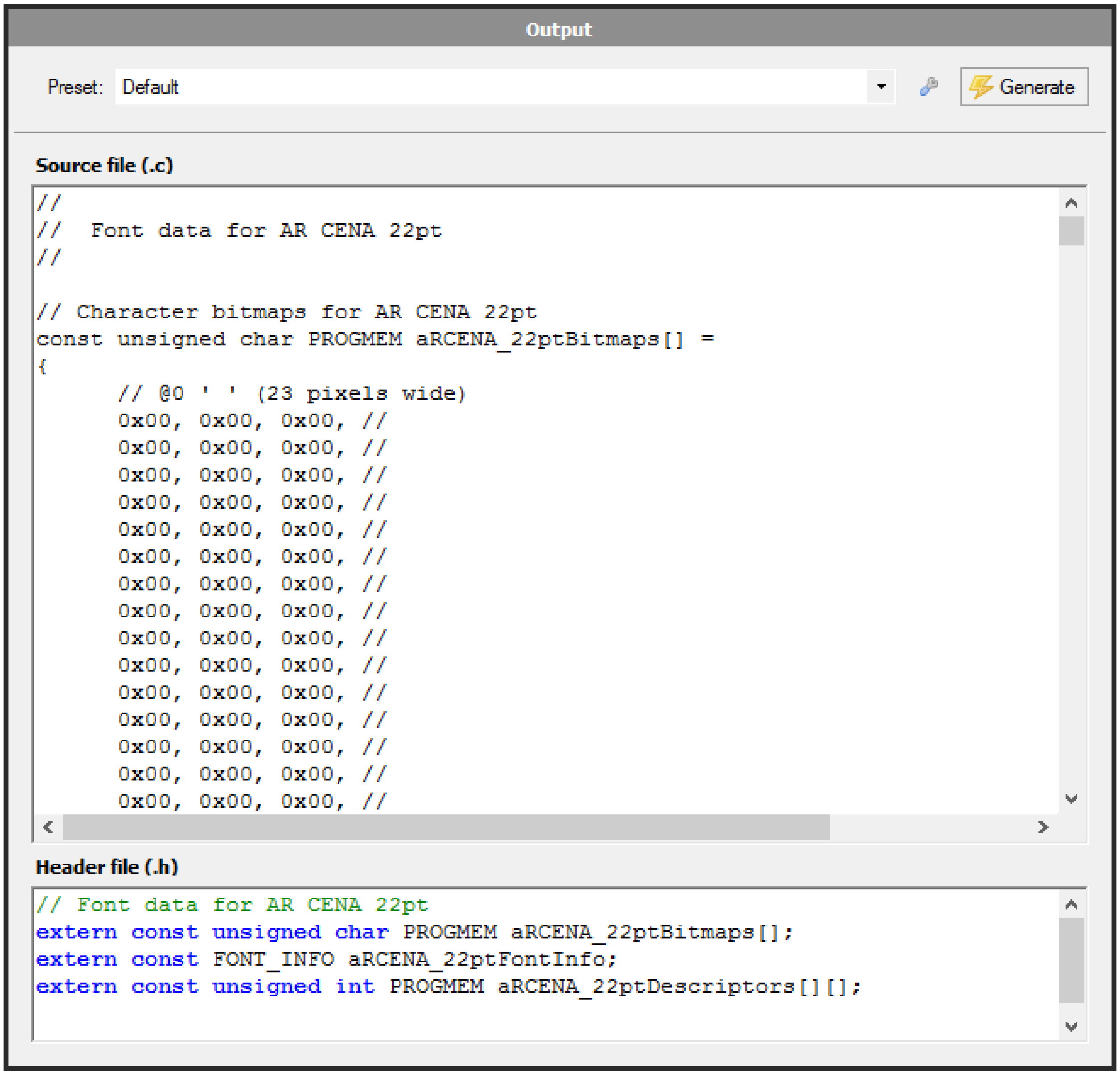


```
!"#$%&'()*
+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQR
STUVWXYZ
[\]^_`abcdefghijklmnopqr
stuvwxyz{|}~
```

5. Click the Generate button



6. Your Font Preview should display

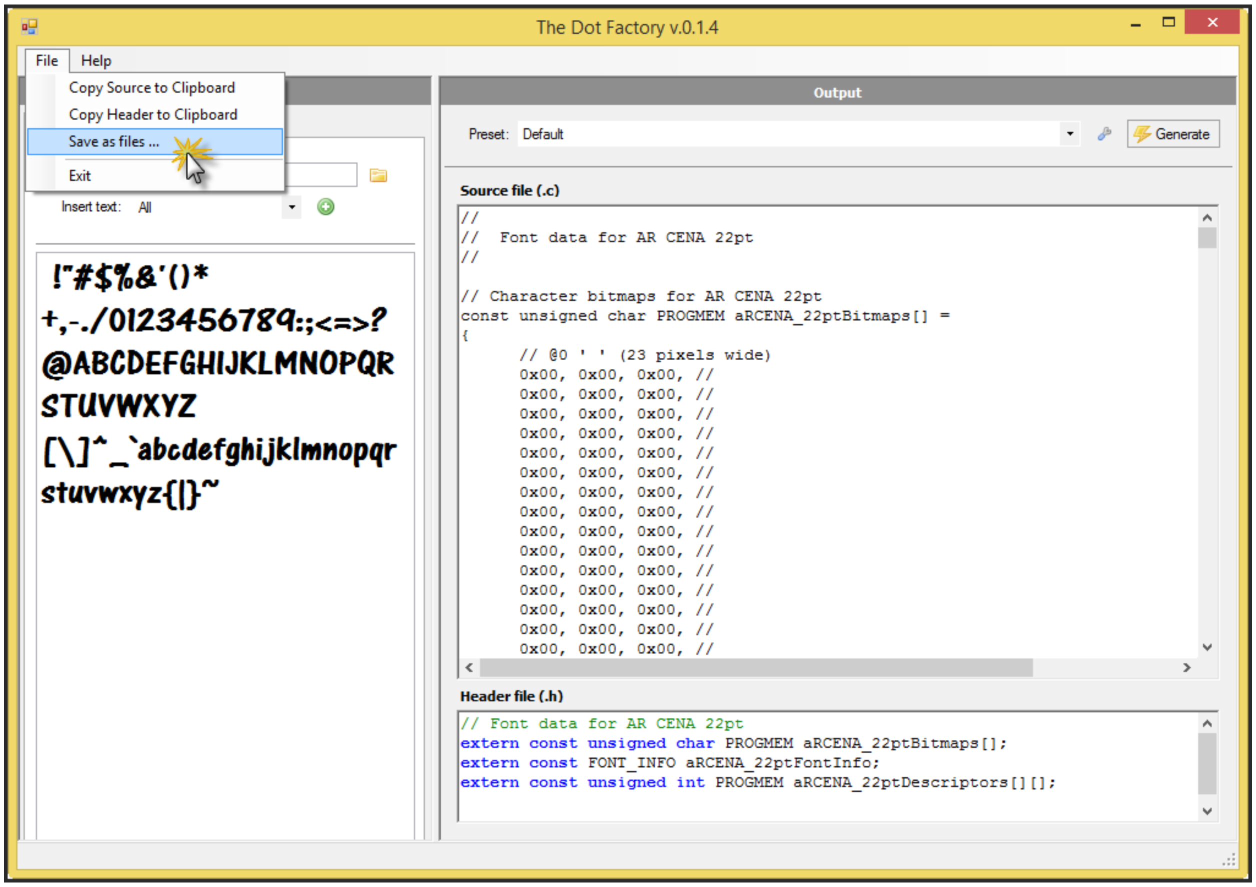


```
//
//  Font data for AR CENA 22pt
//

// Character bitmaps for AR CENA 22pt
const unsigned char PROGMEM aRCENA_22ptBitmaps[] =
{
      // @0 ' ' (23 pixels wide)
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
```

```
// Font data for AR CENA 22pt
extern const unsigned char PROGMEM aRCENA_22ptBitmaps[];
extern const FONT_INFO aRCENA_22ptFontInfo;
extern const unsigned int PROGMEM aRCENA_22ptDescriptors[][];
```

7. Save the files. Click File>Save as files…



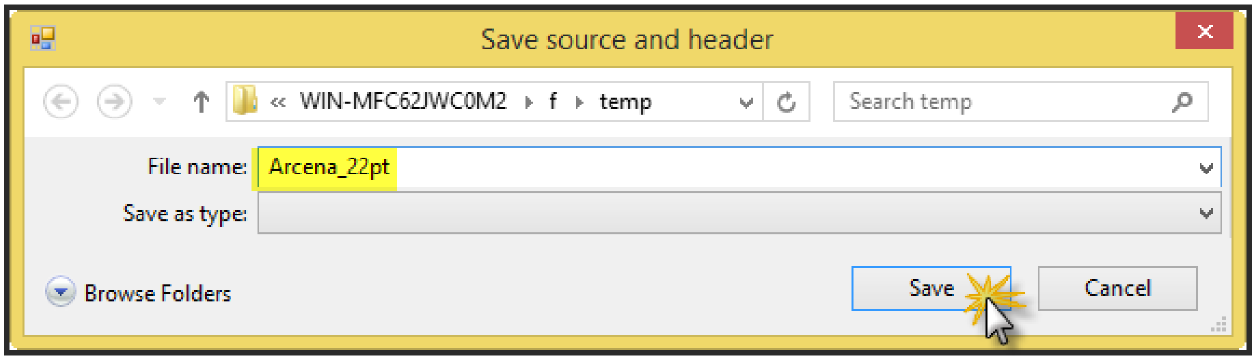
!"#$%&'()*
+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQR
STUVWXYZ
[\]^_`abcdefghijklmnopqr
stuvwxyz{|}~

```
//
//  Font data for AR CENA 22pt
//

// Character bitmaps for AR CENA 22pt
const unsigned char PROGMEM aRCENA_22ptBitmaps[] =
{
      // @0 ' ' (23 pixels wide)
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
      0x00, 0x00, 0x00, //
```

```
// Font data for AR CENA 22pt
extern const unsigned char PROGMEM aRCENA_22ptBitmaps[];
extern const FONT_INFO aRCENA_22ptFontInfo;
extern const unsigned int PROGMEM aRCENA_22ptDescriptors[][];
```

8. Enter a name. **Do not add a file extension. The program creates the .h and .c file for you.** Click Save



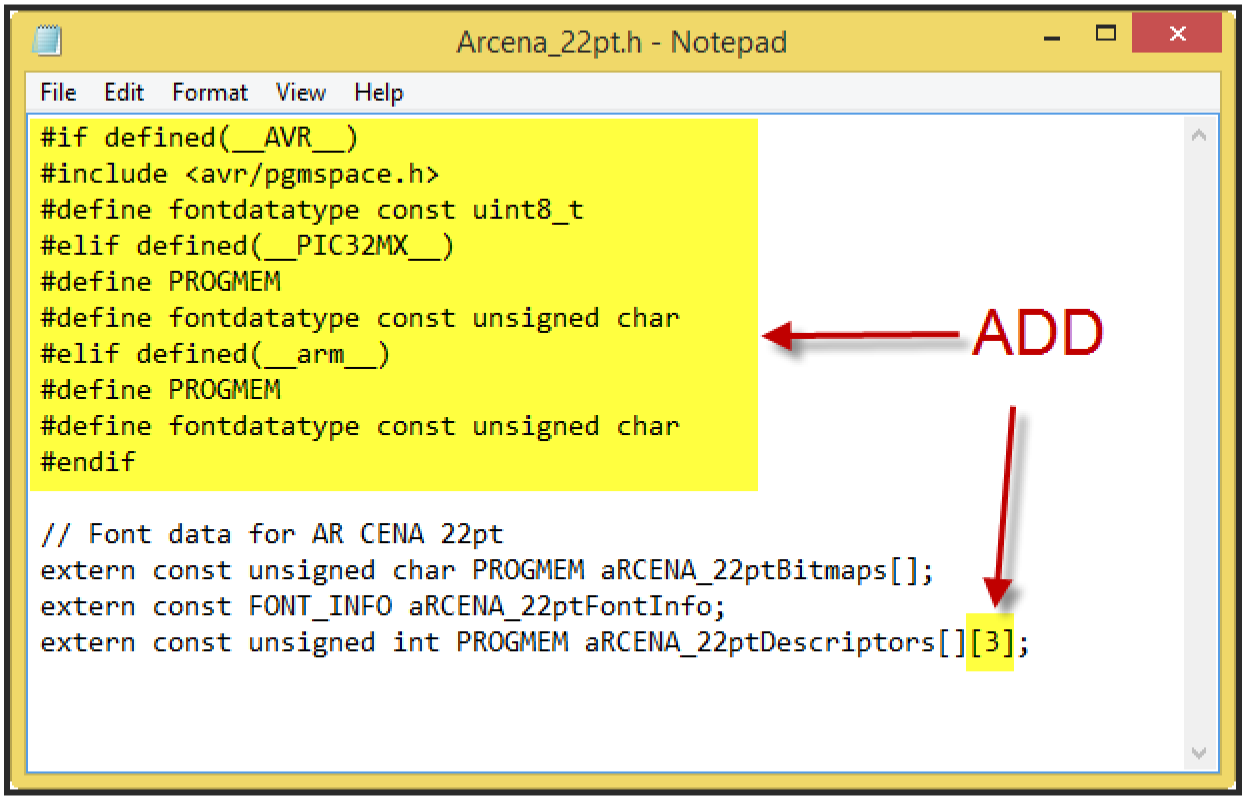
File name: Arcena_22pt
Save as type:

Modify the files

Once the files are saved, we need to modify the .h and .c files so they compile. Copy the following:

```
#if defined(__AVR__)
#include <avr/pgmspace.h>
#define fontdatatype const uint8_t
#elif defined(__PIC32MX__)
#define PROGMEM
#define fontdatatype const unsigned char
#elif defined(__arm__)
#define PROGMEM
#define fontdatatype const unsigned char
#endif
```

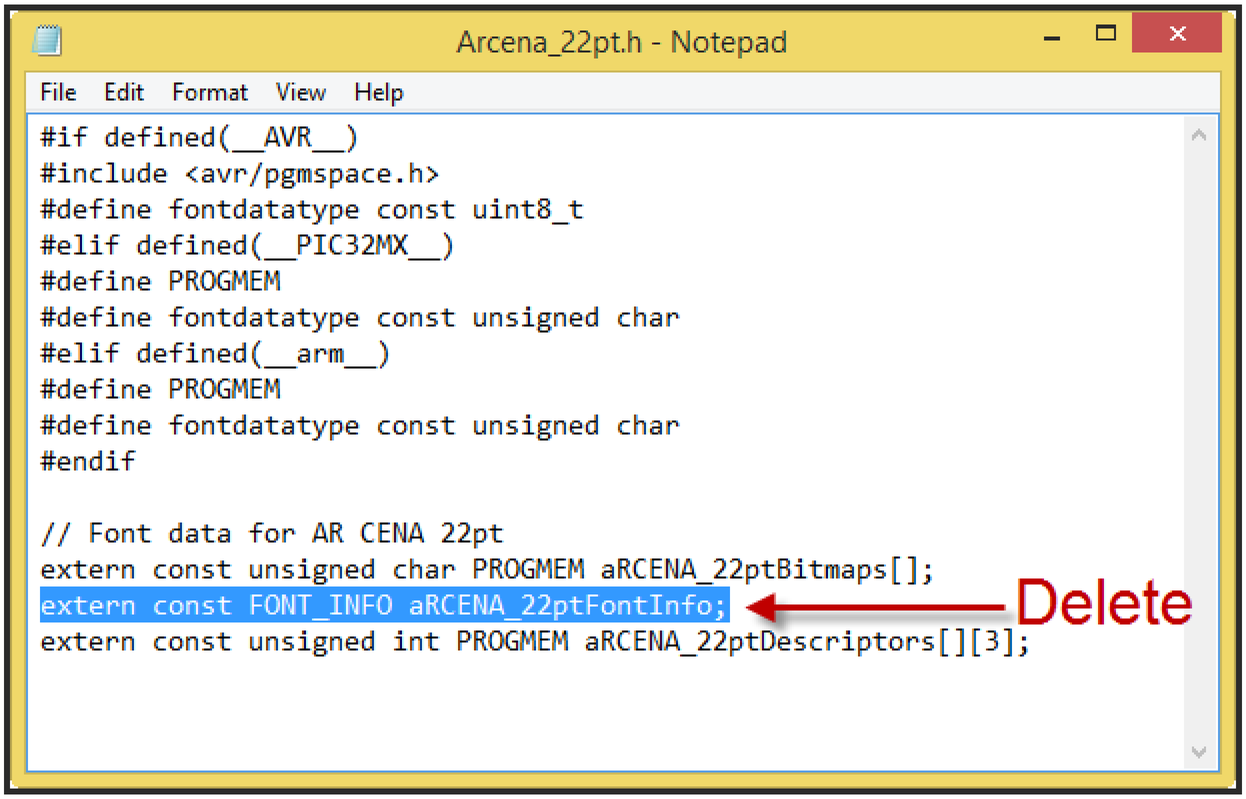
Open the .h file and the defines as shown. Also add a 3 to the descriptor to indicate the array size



```
#if defined(__AVR__)
#include <avr/pgmspace.h>
#define fontdatatype const uint8_t
#elif defined(__PIC32MX__)
#define PROGMEM
#define fontdatatype const unsigned char
#elif defined(__arm__)
#define PROGMEM
#define fontdatatype const unsigned char
#endif

// Font data for AR CENA 22pt
extern const unsigned char PROGMEM aRCENA_22ptBitmaps[];
extern const FONT_INFO aRCENA_22ptFontInfo;
extern const unsigned int PROGMEM aRCENA_22ptDescriptors[][3];
```

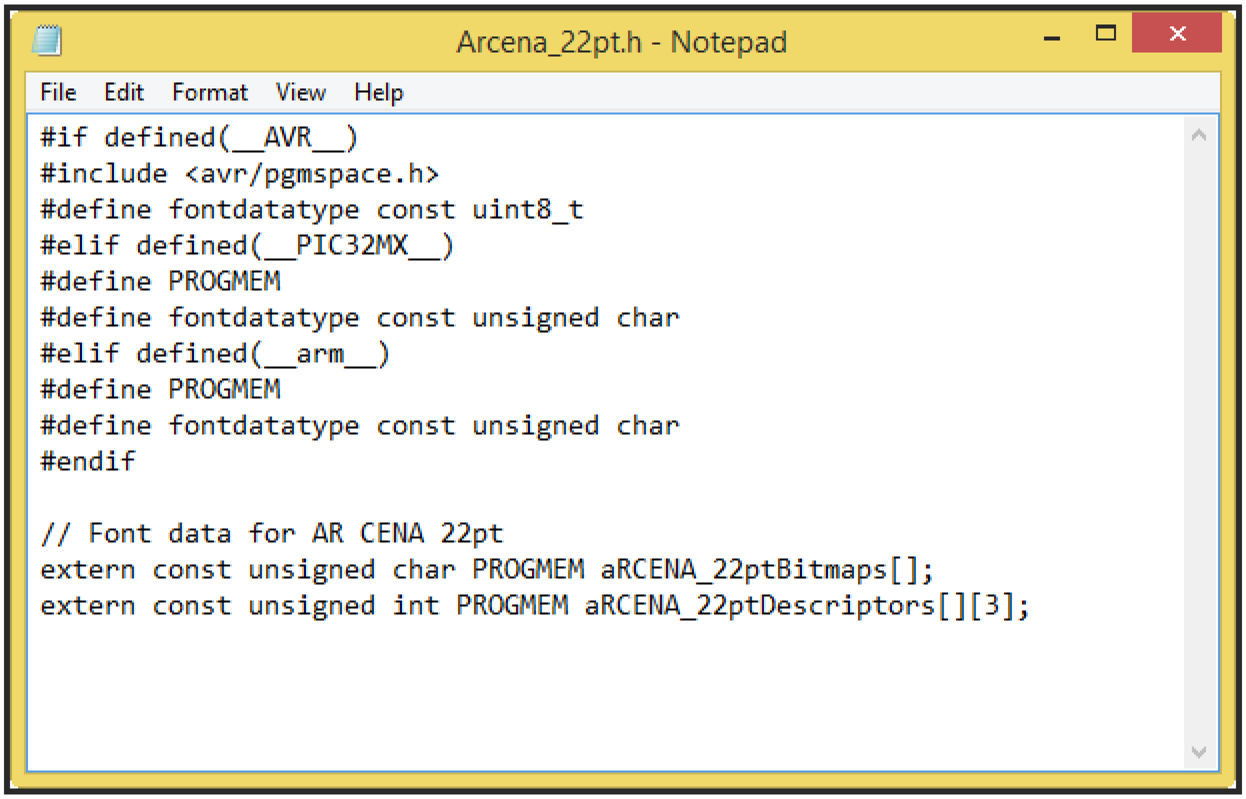
Now delete the FONT_INFO function.



```
#if defined(__AVR__)
#include <avr/pgmspace.h>
#define fontdatatype const uint8_t
#elif defined(__PIC32MX__)
#define PROGMEM
#define fontdatatype const unsigned char
#elif defined(__arm__)
#define PROGMEM
#define fontdatatype const unsigned char
#endif

// Font data for AR CENA 22pt
extern const unsigned char PROGMEM aRCENA_22ptBitmaps[];
extern const FONT_INFO aRCENA_22ptFontInfo;
extern const unsigned int PROGMEM aRCENA_22ptDescriptors[][3];
```

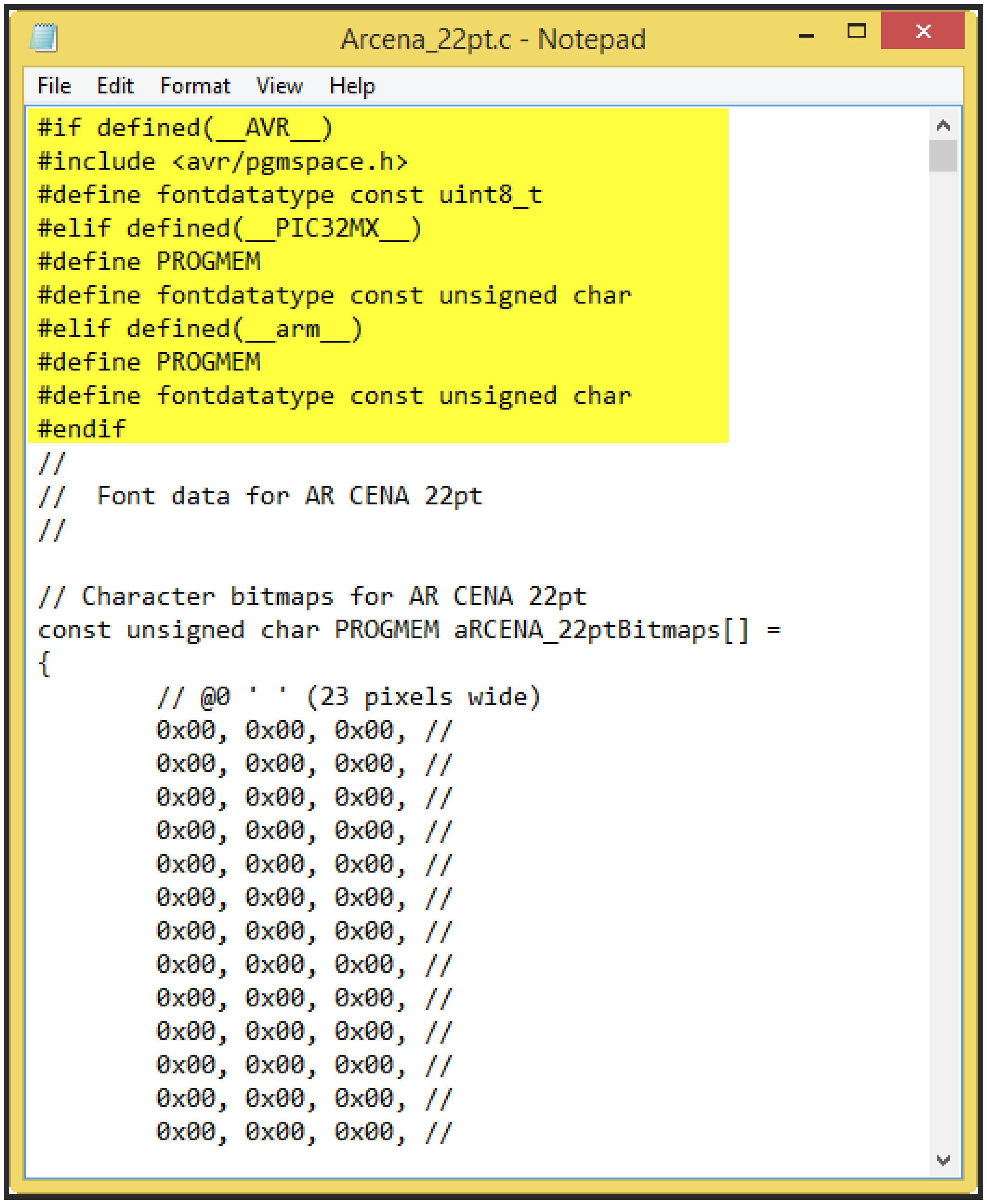
Your file should look like this. When you are done, click Save.



```
#if defined(__AVR__)
#include <avr/pgmspace.h>
#define fontdatatype const uint8_t
#elif defined(__PIC32MX__)
#define PROGMEM
#define fontdatatype const unsigned char
#elif defined(__arm__)
#define PROGMEM
#define fontdatatype const unsigned char
#endif

// Font data for AR CENA 22pt
extern const unsigned char PROGMEM aRCENA_22ptBitmaps[];
extern const unsigned int PROGMEM aRCENA_22ptDescriptors[][3];
```

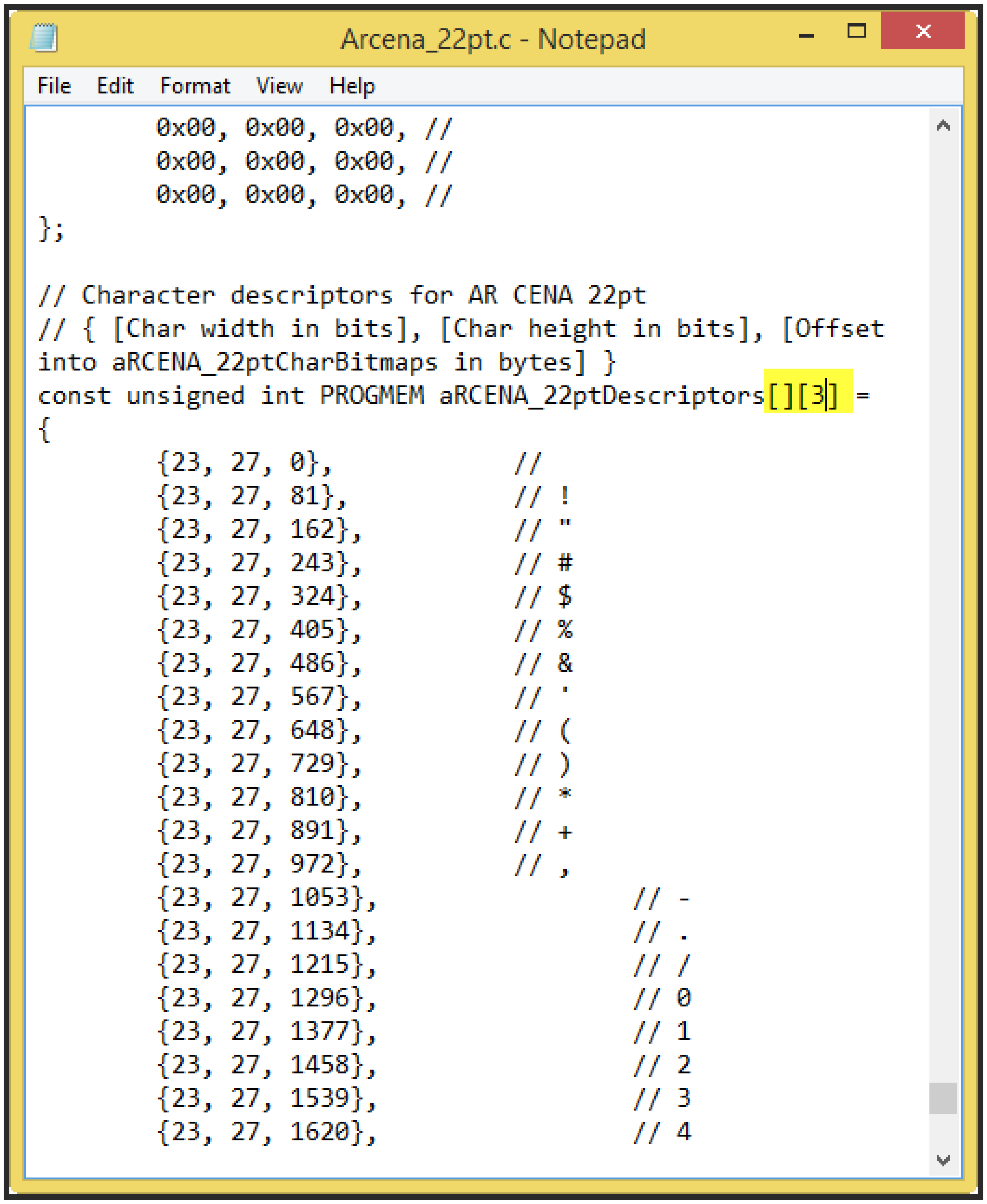
Now open the .c file and add the define code as shown below.



```
#if defined(__AVR__)
#include <avr/pgmspace.h>
#define fontdatatype const uint8_t
#elif defined(__PIC32MX__)
#define PROGMEM
#define fontdatatype const unsigned char
#elif defined(__arm__)
#define PROGMEM
#define fontdatatype const unsigned char
#endif
//
//  Font data for AR CENA 22pt
//

// Character bitmaps for AR CENA 22pt
const unsigned char PROGMEM aRCENA_22ptBitmaps[] =
{
	// @0 ' ' (23 pixels wide)
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
	0x00, 0x00, 0x00, //
```

Now, scroll down to the descriptor and add the 3 to the array.



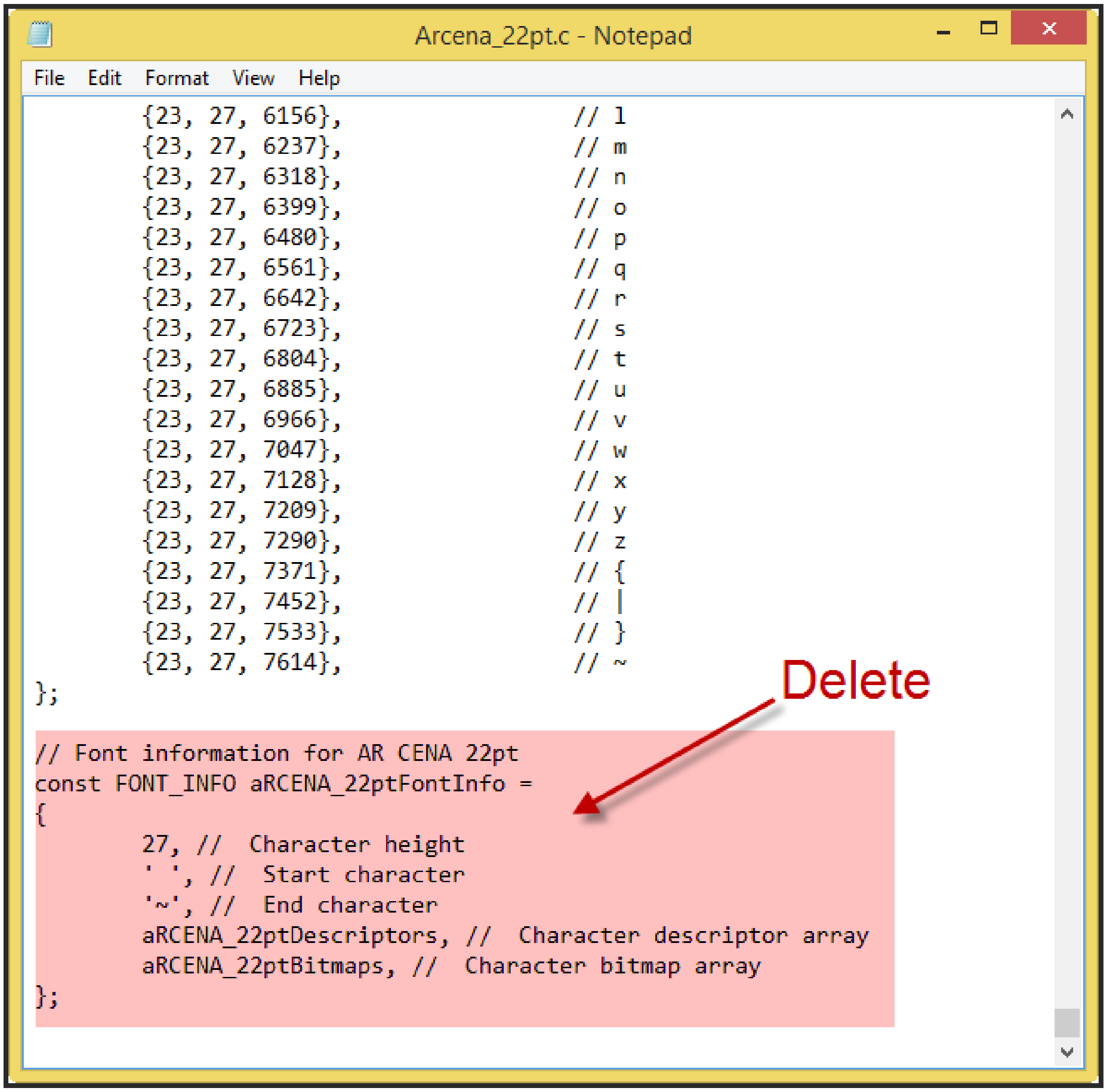
Arcena_22pt.c - Notepad

File Edit Format View Help

```
        0x00, 0x00, 0x00, //
        0x00, 0x00, 0x00, //
        0x00, 0x00, 0x00, //
};

// Character descriptors for AR CENA 22pt
// { [Char width in bits], [Char height in bits], [Offset
into aRCENA_22ptCharBitmaps in bytes] }
const unsigned int PROGMEM aRCENA_22ptDescriptors[][3] =
{
        {23, 27, 0},            //
        {23, 27, 81},           // !
        {23, 27, 162},          // "
        {23, 27, 243},          // #
        {23, 27, 324},          // $
        {23, 27, 405},          // %
        {23, 27, 486},          // &
        {23, 27, 567},          // '
        {23, 27, 648},          // (
        {23, 27, 729},          // )
        {23, 27, 810},          // *
        {23, 27, 891},          // +
        {23, 27, 972},          // ,
        {23, 27, 1053},                 // -
        {23, 27, 1134},                 // .
        {23, 27, 1215},                 // /
        {23, 27, 1296},                 // 0
        {23, 27, 1377},                 // 1
        {23, 27, 1458},                 // 2
        {23, 27, 1539},                 // 3
        {23, 27, 1620},                 // 4
```

Scroll down and delete the FONT_INFO function.



```
        {23, 27, 6156},                 // l
        {23, 27, 6237},                 // m
        {23, 27, 6318},                 // n
        {23, 27, 6399},                 // o
        {23, 27, 6480},                 // p
        {23, 27, 6561},                 // q
        {23, 27, 6642},                 // r
        {23, 27, 6723},                 // s
        {23, 27, 6804},                 // t
        {23, 27, 6885},                 // u
        {23, 27, 6966},                 // v
        {23, 27, 7047},                 // w
        {23, 27, 7128},                 // x
        {23, 27, 7209},                 // y
        {23, 27, 7290},                 // z
        {23, 27, 7371},                 // {
        {23, 27, 7452},                 // |
        {23, 27, 7533},                 // }
        {23, 27, 7614},                 // ~
};

// Font information for AR CENA 22pt
const FONT_INFO aRCENA_22ptFontInfo =
{
        27, //  Character height
        ' ', //  Start character
        '~', //  End character
        aRCENA_22ptDescriptors, //  Character descriptor array
        aRCENA_22ptBitmaps, //  Character bitmap array
};
```

Copy the .h and .c files to Arduino/libraries director or your sketch folder