Fontmaker für Bascom-fonts

Warum ein Fontgenerator?

Bascom liefert (meines Wissens) kein Tool mit, um eigene Fonts aus einem Systemfont zu erstellen.
Diese Aufgabe ist auch nicht so ganz trivial, denn Systemfonts sind keine einfachen Schwarz-Weiß Bitmaps mehr, wie zu Computer-Urzeiten.

Es gibt mehrere Notwendigkeiten und Wünsche, weshalb man eigene "Controllerfonts" aus Systemfonts erzeugen möchte:

1. Moderne Standardfonts sind meist mit einem Anti-Aliasing (Kantenfarbverlauf) versehen und sie sind somit also nicht so leicht z.B. auf einfache Controllerdisplays zu übersetzen.
2. Es ist oft notwendig (vor allem aus Speicherplatzmangel:ATTiny), spezialisierte Fontlibs zu erstellen. So kann es sein, dass man für ein Projekt nur Zahlen benötigt, oder aber nur Buchstaben eines Fonttyps.
3. Man möchte keinen überflüssigen Overhead erzeugen, also blanke Bereiche um die Fonts sind zu vermeiden (auch wieder Speicherplatzeinsparung und bessere Les- und Plazierbarkeit).
4. Man benötigt fast immer (eigene) Sonderzeichen oder eine bestimmte grafische Anmutung der Zeichen auf dem Display.
5. Hin und wieder verwendet man gerne invertierte Buchstaben, um z.B. einen Edit-Modus anzuzeigen
5a. **Tip:** Hier muss man den Frame um den Buchstaben oben und unten um 2 Pixel verbreitern, damit wird das Zeichen invertiert besser lesbar!
6. Einzelne Buchstaben müssen manchmal bei bestimmten Fonts innerhalb ihres Frames etwas verschoben werden, damit die Lesbarkeit bessert wird...
7. Frage: "Es gibt doch schon viele Fonts fertig im Netz?!" ...ja, ein paar gibt es. Aber leider passen die -zumindest bei mir- nie zum Anwendungsfall, den ich gerade bearbeite. Außerdem ist auf kleinen Displays nur sehr eingeschränkter Platz, der muss ideal ausgenutzt werden, noch dazu im passenden Stil.

Das alles kann man händisch nicht erreichen und sich für jeden Font eine Excell-Tabelle zu basteln, in der man die für die Fonts nötigen Bitcodes "zusammenzufrimelt", ist nicht sonderlich sinnvoll.
Grafisch kontrollierbar ist das Vorgehen schon gar nicht.

Ich habe mich daher an einen solchen (ausbaubaren) Converter gemacht, der aktuell (März 2018) noch in einer Beta vorliegt, das wird wohl auch so bleiben.
Bei der Entwicklung habe ich Wert auf eine sehr einfache Oberfläche gelegt, die keine besonderen Tools oder Elemente verwendet, die unter Windows extra installiert werden müssen.
Trotzdem kann es sein, dass der eine oder andere User irgendwas nicht korrekt auf dem Rechner installiert hat, daher ist der 'Fontmaker' als Installationssetup aufgesetzt.
Wer in Zukunft ein Update ziehen möchte (das ich immer wieder mal hochladen werde) der kann sich einfach die neue 'Fontmaker.exe' ziehen und die alte 'Fontmaker.exe' überschreiben, es muss dann nichts mehr neu installiert werden.

Erläuterungen zum Programm

1. Installation

2. Oberfläche
2.1 Auswählen eines Systemfonts
2.2 Einstellen der gewünschten "Frames", der Pixelanzahl in Breite und Höhe, sowie die Textgröße (Punkt) für den neu zu erstellenden Font
2.3 Wichtung der Kantenglättung
2.4 Weitere Schritte...
2.5 Font-Kombinationen in einer ".font"-Fontdatei

3. Beispiel für einen typischen Vorgang

4. Displaysimulator

5. Pixeler

Zu 1. Installation

Die Installation geht wie üblich vonstatten: Entzippen des Files "Fontmaker_Setup.zip", die Setup-Exe doppelklicken und die Installation starten.

Updates werde ich in unregelmäßigen Abständen immer mal wieder hochladen.

Wer das Programm bereits einmal installiert hat, kann ab dann auch nur die reine "Fontmaker.exe" ziehen ("Bascom_Fontmaker.zip") und damit die alte "Fontmaker.exe" einfach überschreiben, ohne das Programm neu installieren zu müssen, da sich an den Registryeinträgen wohl auf absehbare Zeit nichts ändern wird.

Zu 2. Oberfläche

Die grundlegenden Schritte bzw. die Reihenfolge der sinnvollen Anwendung ist mit den gelb hinterlegten Zahlen zur Reihenfolge Ihrer Abarbeitung vorgegeben.
Hält man den Mauspfeil länger auf einem solchen gelben Punkt, ploppen weitere Erklärungen auf. Damit ist quasi eine rudimentäre online-Hilfe enthalten (werde ich noch mehrsprachig machen).
Arbeitet man nun die Punkte ihrer Reihenfolge nach ab, kommt man ganz schnell zum Ziel.

Zu 2.1 Auswählen des Systemfonts [1]

Es gibt verschiedene Fonttypen. Neben den Hauptvertretern mit Kantenglättung gibt es auch Fonts ohne diese Glättung. Bei diesem zweiten Typ kommt nach der Konvertierung exakt der gleiche Font hinten heraus, der vorne eingegeben wurde. Diese Fonts sind natürlich einfach in der Verarbeitung und haben den Vorteil, dass sie auf jedem Display gut lesbar sein werden.
Außerdem gibt es Fonts, bei denen alle Zeichen die gleiche Breite haben. Man nennt diese "Fixfonts". Besonders Fixfonts sind gut geeignet für kleine Displays.
Für mich sind diese Fonts aber meist eher "langweilig".
Weiter gibt es die "TrueType Fonts". Diese TT-Fonts werden heute häufig benutzt, denn es sind sogenannte Outline-Fonts, also Vectorfonts.
TrueType Fonts bestehen im Gegensatz zu Pixelfonts nicht aus absolut positionierten Pixeln, sondern sie sind parametrisch beschrieben, eben durch Vectoren.
Damit lassen sie sich bequem skalieren, ohne an den Rändern pixelig zu werden, wenn man sie vergrößert. Sie, die Buchstaben oder Zeichen, haben wie viele andere (auch Pixelfonts) variable Breiten und sind dadurch eine Herausforderung, also was die Anpassung an kleine Displays ohne TrueType-Funktionalität angeht. Diese Fonts werden, je nach Zeichenbreite, mehr oder weniger zusammengeschoben, Zeichen für Zeichen, womit alle Zeichen, ob breit oder schmal, einen harmonischen Abstand untereinander aufweisen.
So kann es sein, dass schmale Zeichen nach dem Konvertieren nicht mittig im Frame ausgerichtet sind, weil unsere Controllerfonts keine Proportional- oder TrueType-Funktionalität haben, sie sind starre Pixelfonts, bezeichnend für das "l" oder das "i". Diese Asymetrie unter den Zeichen kann aber im Fontmaker "zu Fuss" kontrolliert und verändert werden, Zeichen für Zeichen.
Am besten kontrolliert man die Änderungen zeitgleich im Display-Simulator. So kann man unschön ausgerichtete Buchstaben nach dem Verschieben gleich kontrollieren.
Mehr zum Simulator weiter unten.

Das Schöne: Mit dem Fontmaker bist Du absolut frei, was die Plazierung im Frame und die genaue Form der Zeichen angeht. Außerdem kann man noch ein paar weitere Dinge mit den Fonts machen, die die tägliche Arbeit mit Controllern doch sehr erleichtern. Ich werde die Optionen bzw. Möglichkeiten mit der Zeit z.B. um eine "Stapelverarbeitung" der Fonts erweitern.

Vorschläge zu Erweiterungen sind sehr erwünscht! Immer her mit den Ideen.

Zu 2.2 Einstellen der Frame-Maße und Anpassung der Font-Größe [2/3/4]

Der "Frame" (ich nenne das einfach mal so) ist die anvisierte Ausgabegröße des Controllerfonts (Pixelanzahl in X und Y-Richtung).

Der Vorgehensweg ist hier so, dass man...

a. ein gewünschtes Pixel-Maß -meist in X-Richtung- vorgeben möchte, um die maximale Zeichenanzahl, die in der Horizontalen auf dem kleinen Ziel-Display nebeneinander passen, zu erreichen. Vorgegeben ist die Anzahl der Zeichen durch die maximal mögliche Textlänge pro Zeile und die Auflösung des Displays. Das muss dann jeder selber ergründen:
>> Zeichenbreite in Pixel X = Displayauflösung-Pixel-X-richtung / Anzahl gewünschter Zeichen pro Zeile<<
b. Danach stellt man die maximal mögliche Textgröße ein, die noch gerade so in den Frame passt.
c. Dabei muss man beachten, dass vor allem die breitesten Zeichen in den Frame passen. Das sind die Zeichen wie "W" oder "M", auch "@" usw. Ich gehe dabei auch gerne so vor, dass ich bei TrueTypeFonts das M und das W (und andere breite Zeichen) gerne per Hand neu zeichne, damit ich den Frame möglichst klein halten kann. Fixfonts sind natürlich komplett unkompliziert, weil hier M und W exakt so breit sind, wie alle anderen Zeichen.
Würde man bei TrueTypeFonts den Frame genau so breit machen wie M und W es erfordern, dann würden alle anderen Zeichen am Ende auf dem kleinen Display sehr weit auseinander stehen.
Das sieht nicht nur unschön aus, es ist auf dem Display auch schlecht leserlich.
d. Die Fonts haben nun noch die Eigenschaft, dass sie verschiedene "Kopffreiheit" haben. Das heißt, die Zeichen haben nach oben manchmal zu viel Freiraum. Auch die seitliche Ausrichtung ist nicht immer so, wie es für die reduzierten Fonts ideal ist. Beide Einstellungen kann man mit "X-Offset" und "Y-Offset" genau justieren. Für Buchstaben wie "g" und "j" benötigt man noch Extra-Platz nach unten.
Benötigt man nur die Zahlen 0-9 und ein paar Sonderzeichen, dann kann man diesen Freiraum vergessen und den Frame entsprechend kleiner machen.

Zu 2.3 Wichtung der Kantenglättung bei TrueTypeFonts [5]

Die Wichtung der Kantenglättung ist ein wichtiger Punkt, der den "Fontmaker" auszeichnet.
Eigentlich keine große Sache.
Damit kannst Du die Grauverläufe an den Zeichenkanten mehr oder weniger mit in die neu zu erzeugenden Zeichen einfließen lassen.
So kann man sich vor der folgenden händischen Nachbearbeitung eine Menge Arbeit sparen und die Qualität der Konvertierung stark verbessern.
Durch die Wichtung werden nicht einfach alle Farbabstufungen der Kantenpixel mit in die Konvertierung einbezogen oder auch ausgeschlossen, sondern die Nutzung kann individuell durch die Wichtung ein- oder ausgeschlossen werden.
Das verbessert das Ergebnis erheblich.

Um das Endergebnis noch zu perfektionieren, muss man die Zeichen noch per Hand idealisieren und nachpixeln...

...oh man, das riecht nach Arbeit...

Aber Keine Angst: Dieser Vorgang geht sehr schnell von der Hand und macht richtig Spaß, denn er ist auch sehr kreativ, wie ich finde. Ihr erzeugt so quasi Eure eigenen Fonts!

Das Ergebnis und das Nutzen der "eigenen" Fonts macht dann umso mehr Spaß und bekommt einen ganz anderen Wert für Euch.

Zu 2.4 Weitere Schritte

Um nun den Font zu konvertieren, musst Du nur noch auf "convert all chars >>>" klicken.
Man kann die Zeichen auch einzeln übertragen (Doppelklick im linken ASCII-Fenster), das schnellere Vorgehen ist aber das Übertragen des Gesamtfonts, um dann die einzelnen Zeichen noch im Pixeleditor zu überarbeiten, wenn nötig.

Achtung: Die Parameter sind jetzt (vorläufig) gesperrt zum Schutz bei weiteren Arbeiten am neuen Font!

Einmal übernommen macht es in den meisten Fällen auch keinen Sinn mehr, an den Parametern herumzuändern. Es enstehen außerdem Probleme im Font, wenn Du nach der Konvertierung noch an z.B. Pixelbreite oder Höhe herumbasteln würdest, denn die Konvertierung ist ja auf fixe Maße festgelegt und kann nachträglich aufgrund der Struktur

des Fonts nicht geändert werden, denn dann entsteht ein Pixelchaos im rechten Pixelerfenster, was an der Spaltenspeicherung der Fonts liegt.
Kurz erklärt: Ein Font wird in Bytes gespeichert. Dass heißt, die Pixel eines Fonts werden immer in senkrechten Spalten zu je 8 Pixeln in eine Byte geschrieben. Wenn man nun die Pixelbreite eines Zeichens nachträglich verändert, verschieben sich die Umbrüche am Ende des Frames und Alles gerät durcheinander.

Damit ist es nun zunächst nicht mehr möglich, noch an den Parametern herumzuspielen, zum Schutz des Users, denn man könnte sonst zu schnell etwas überschreiben, was man eigentlich nicht überschreiben wollte.
Aber man kann durchaus durch einen "Reenable" des Sperrmodus ("File/Reenable)") an die Originalzeichen kommen, also das linke ASCII-Fenster wird nun wieder geöffnet, sowie einige der Parameter Zur Schrifteinstellung, außer der Pixelbreite eines Fonts, wie eben erwähnt, die Einstellung bleicht aus logischen Gründen gesperrt.
Damit kann man dann wieder einzelne Zeichen "von links nach rechts" übernehmen, bereits durchgeführte Zeichenänderungen im Pixelbereich (also die geänderten Zeichen im rechten ASCII-Fenster) werden aber hierbei überschrieben und man muss im Zweifel diesen Buchstaben/Zeichen erneut überarbeiten.
Da man das vor allem dann macht, wenn man sich komplett "verpixelt" hatte und das neue Zeichen so gar nicht mehr "In Shape" ist, also deine Pixelkunst aus dem Ruder gelaufen ist, ist das Zeichen schnell neu aus dem Originalfont übertragen und man kann von vorne anfangen, ohne alle anderen bisher bearbeiteten Zeichen verwerfen zu müssen.

2.5 Font-Kombinationen in einer ".font"-Fontdatei

Weiter ist es so möglich, verschiedene Fonts "zusammenzukleistern".
Sinnvoll ist das dann, wenn man z.B. Buchstaben eines Fonts mit Zahlen eines anderen Fonts kombinieren in einem einzigen Font bzw. einer einzigen Fontdatei kombinieren möchte.
Das macht das Handling in der Praxis einfacher, denn man muss beim Programmieren nicht zwischen zwei oder mehreren Fonts hin und herschalten.
Denn es kommt gerne vor, dass einem zwar die Buchstaben eines Fonts gefallen, die Zahlen aber einfach nicht gut aussehen (hatte ich öfter).

Man kann einzelne Zeichen auch durch "Doppelklick" vom linken "Originalfenster" nach rechts ins "New ASCII"-Fenster übertragen.
Einen neuen Font -zusammengebastelt aus zwei oder mehreren unterschiedlichen Fonts- kombiniert man am besten so, dass man den...

1. "Hauptfont" zuerst einstellt, also alle Randparameter auf diesen Font hin optimiert, wie Zeichengröße, Textgröße, Versatz usw....dann...
2. die Zeichen überträgt, entweder alle oder nur ein paar gewünschte, also mit "convert all chars >>>" oder einzeln mit Doppelklick im linken Fenster, und dann...
3. in "File/Reenable" die Auswahl wieder öffnet/frei macht, den nächsten Font auswählt, die Einstellungen anpasst und nun die zusätzlichen Zeichen hinzuklickt.

Das Reenable sperrt nur noch die Zeichenanzahl (...z.B. ASCII45- ASCII145) und die Pixelbreite und Höhe des neuen Fonts. Das muss so sein, weil die Fonts sonst im Pixelbereich durcheinander geraten.
That's it...

Hat man zuvor einzelne Zeichen bereits übertragen, die Einstellungen wieder aufgemacht ("Reenable") und möchte nun alle anderen Zeichen eines Font komplett mit "convert all chars >>>" übertragen, werden die zuvor bereits übertragenen Zeichen NICHT überschrieben, sie bleiben bestehen!

Möchte man, ohne das Programm ganz verlassen zu müssen, einen komplett neuen Font erstellen, dann kann man das über "File/New Font" machen.

Zu 3. Beispiel für einen typischen Konvertierungs-Vorgang

Der Ablauf ist ganz einfach und recht frei gehalten. Es gibt aber einen idealisierten, ersten Weg durch die Erstellung eines "neuen" Fonts.
Zuerst schildere ich einen Durchgang vom ersten bis zum letzten Schritt, dann erläutere ich noch, wie man am Konvertierten Font noch Änderungen durchführt, was der spaßigere Teil ist.
Außerdem gibt es noch eine kleine Einführung in die Nutzung des einfachen Displaysimulators und des Pixelers.

Normaler Durchlauf, die Nummerierung richtet sich nach den Nummern, mit denen die einzelnen Punkte im Programm markiert sind (im Programm gelb hinterlegt)

1. Fontauswahl (selbsterklärend)

2. Pixelbreite und Pixelhöhe, das Richtmaß hierbei ist meist die Breite, denn sie definiert, wieviele Zeichen am Ende auf das Controllerdisplay passen. Die Höhe ergibt sich am Ende durch die Buchstaben, vor allem "q", "g" und "j", weil diese beiden Buchstaben unterhalb des Textstriches verlaufen. Nach oben sind es die Großbuchstaben, die die Ausrichtung der Zeichen im Frame vorgeben.

3. Auswahl eines Zeichens, um in der Folge die Textgröße anzupassen...

4. Textgröße des Originalfonts. Die Textgröße muss der Frame angepasst werden, damit jeder Buchstabe in den Frame passt. Vor allem sind es hier "M" und "W", sowie "Q", die aus dem Rahmen fallen, was deren Breite angeht.

Diese vier Auswahlen definieren zunächst die wichtigsten Grundeinstellungen.

5. Kantenwichtung: Bei Fonts mit Kantenverlauf (Antialiasing) kann man jetzt noch die Kantenwichtung einstellen über den Slider. Hier ist etwas Fingerspitzengefühl gefragt, um am Ende so wenig wie nur möglich nachbearbeiten zu müssen. Bei der Kantenwichtung werden mehr oder weniger die Abstufungen der Kanten übernommen.

...(noch ohne Nummer) Position im Frame einrichten: Dazu kann man die genaue Position der Zeichen im Frame bestimmen. Das ist nötig, wenn die Zeichen irgendwie zu weit oben, unten rechts oder links verschoben sind. Hiermit positioniert man alle Zeichen möglichst im Zentrum des Frames, das ergibt beim Programmieren am Ende am wenigsten Probleme mit der Ausrichtung des Fonts auf dem Display.

6. "convert all chars >>>" drücken, um alle Zeichen auf einmal "nach rechts" in einen neuen Font zu wandeln, unter Berücksichtigung der eingestellten Parameter. Man kann auch alle Zeichen durch "Doppelklick" im linken "ASCII Tab" nach rechts übertragen. Es müssen aber noch nicht alle Anpassung perfekt sein, einzelne Zeichen kann man auch nach der Übertragung noch anpassen, sowohl in der Ausrichtung, als auch in der Wichtung der Kanten.

Jetzt sind die Zeichen schon mal konvertiert. Es folgt die einzelne Bearbeitung der übertragenen Zeichen.

Zuerst bietet es sich an, die übertragenen Zeichen (im rechten ASCII-Tab) und deren Ausrichtung im Frame noch mal einzeln zu kontrollieren, BEVOR man an den Zeichen herumpixelt.
Denn wenn man jetzt z.B. ein Zeichen verschiebt, dann wird das Original von links hergenommen, weil sonst unter Umständen Details an den Kanten nicht im übertragenen Font vorhanden sind (wenn z.B. ein oder mehrere Zeichen nach links an einer Kante abgeschnitten wurden). Das kann so korrigiert werden. Nachteil: Bereits durchgeführte Pixelkorrekturen werden gelöscht und man muss das nochmal machen. Daher zuerst verschieben, wenn nötig, dann Pixeln!

7. Möchte man ein "komisches", für unseren Sprachraum unnützes Zeichen aus z.B. osteuropäischen Sprachen oder sonst welche für unsere geplanten Zwecke "überflüssige Zeichenräume" nutzen, um eigene Zeichen unterzubringen (für mich ist klassisch immer gebraucht und in vielen Fonts nicht im Standard vorhanden ein schönes "°"-Zeichen oder ein kombiniertes "°C", das in einem Zeichen liegen kann), dann ist das einfach machbar, indem man das zu überschreibende Zeichen rechts in der Box "New ASCII-Tab" anklickt und im Pixelfenster links daneben mit der Maus einfach bearbeitet (linke Maustaste schwarze Punkt, rechte Taste weißer Punkt).
Wenn man ein gänzlich neues Zeichen aufsetzen möchte kann man sich das Leben durch "Clear" vor dem Entwurf erleichtern.
Möchte man das Zeichen invertieren, um z.B. einen Font für einem "Edit-Modus" für das Controllerprogramm aufzusetzen, kann man das mit Klick auf "Invers" erledigen.

Wichtig:
8. Ist ein Zeichen fertig bearbeitet, muss es in die rechte Box "New ASCII-Char" übertragen werden durch Klick auf ">>>" !!!

Oben unter **"2.5 Font-Kombinationen in einer ".font"-Fontdatei"** sind weitere Hinweise zur Abeitsweise mit mehreren Fonts in einer Datei und der Umgang mit Nachbearbeitungen bereits erläutert.

Zu 4. Displaysimulator

Unter "Tools/Displaysimulator + Pixeler" kann man das Erscheinungsbild sogleich untersuchen und Änderungen am Font vornehmen und sie so quasi finalisieren.

Nach der Fertigstellung eines neuen Fonts rufe den Simulator einfach mal auf.
Zuerst kann man die Auflösung des geplanten Displays eingeben, die Hintergrundfarbe und Pixelfarbe.
Dann klicke auf "Chars/Sentence". Es erscheint ein Testtext und weiter unten die Parameter dieses Textes: "Text,20,20,Test".
Zuerst steht dort "Text". Dieses erste Wort darf nicht verändert werden, es zeigt dem Parser, was jetzt kommt, nämlich ein Text.
Die beiden Koordinaten sind die Startposition X (20) und Y (20) und dann folgt der auf dem Display gezeigte Text "Test".
Diese drei Daten kann man ändern, einfach im Text.

Außerdem kann man eine Linie setzen (und es folgen in nächster Zeit noch Kreis und Box). Damit kann man die Anmutung des Textes für viele Anwendungen schon mal testen.
Das ist vor allem sehr sinnvoll, wenn man die Abstände und genauen Positionen einzelner Buchstaben in Wörtern und Sätzen überprüfen und evtl. noch anpassen kann.
So kommt es oft vor, dass Buchstaben wie "l" oder "i", also sehr schlanke Zeichen, gerne unschön verschoben sind im Font, das kannst Du zum Glück im Fonteditor einfach korrigieren durch "X-Offset" und Y-Offset".

Das ganze macht sehr viel Sinn, denn so kann man die Ausrichtung der Zeichen zueinander fein austarrieren, ohne das erst am Controller-Display zu bemerken.

1. Öffne einfach man den Displaysimulator und schiebe das Fenster ein Stück zur Seite.
2. Dann klicke auf "Chars/Sentence" und schreibe unten im Eingabefeld (unterhalb der Listbox, darin geht leider noch nicht) den Gewüschten Text, gerne mit problematischen Buchstabe wie "i" oder "j".
3. Klicke auf den grünen Haken oder press Return, damit wird die Eingabe übernommen
4. Bei TT-Fonts (TrueType) wirst Du höchstwahrscheinlich bemerken, dass einige Buchstabe nicht optimal zu den anderen eingerückt wurden, z.B. ein "i".
5. Markiere im "New ASCII Tab" Fenster auf den gewünschten Buchstaben und verschiebe ihn im Hauptfenster mit "X-Offset" ein wenig.
6. Wichtig: Übernehme die Änderung mit Klick auf ">>>" ober halb des "New ASCII Tab"-Fensters.
7. Klicke im "Display-Simulator" wieder auf den grünen Haken (oder Return) und beobachte dabei, wie sich der geänderte Buchstabe verschiebt.

So kann man den Font sehr schnell anpassen und optimieren.
Das "Korrekturpixeln" kann man nach dem gleichen Muster natürlich auch kontrollieren.

Wie man sieht, sollte der Simulator auch schon VOR dem "Korrekturpixeln" zur Kontrolle eingesetzt werden, um Fehler von Anfang an zu beseitigen.
Aber auch nach dem Pixeln sollte man noch mal alle Zeichen auf dem Simulator checken, um letzte Anpassungen am Font vorzunehmen.
Dann spart man sich so manche Fontüberarbeitungen im Nachhinein, also nach dem Übertragen der Programmes auf den Controller.

Eine weitere Funktion ist das Feld für "Format".

Hier kann man Zahlen von 1-4 eingeben. Dieser Wert definiert die Pixelgröße des Displays. Einige Displays sind zwar groß, haben aber nur relativ wenige Pixel (also alles andere als ein Retina-Displays).
Diesem Umstand kann man mit "Format" Rechnung tragen und das Erscheinungsbild besser an die Gegebenheiten des Displays anpassen.
Letztlich simuliert diese Funktion einfach gröbere Pixel.

5. Pixeler

Der "Pixeler" ist ein einfaches Zeichenprogramm, dass aber noch sehr "beta" und "buggy" ist.
Du erreichst ihn über den Display-Simulator mit dem Button "Pixel-Editor".
Ich bin gerade dabei, das Tool ein wenig zu pimpen und intuitiver zu machen. Das wird aber noch ein Weile in Anspruch

nehmen.

Bislang kann man damit nur Bilder im ".bmp"-Format abspeichern und muss diese dann noch mit dem "Bascom-Grafik-Konverter" umwandeln nach ".bgf", das wird im "Pixeler" aber bald auch direkt funktionieren.

Manchmal benötigt man nämlich gar keine Fonts, denn manchmal kann man die nötigen Oberflächen und Interfaces mit ein paar Grafiken aufbauen, eben weil der Nutzumfang der Zielanwendung sehr eingeschränkt ist, z.B. wie bei einer Uhr oder einem Thermometer.
Aber dann wäre es doch schön, wenn man den richtigen Font dafür verwenden könnte, den man gerade mit dem Fontmaker produziert hat, ohne gleich einen ganzen Font ins Programm mit einzubauen, aus dem man dann nur ein paar Buchstaben benötigt.
Wie wäre es denn, wenn man ein Bild mit der eigenen Schrift pixelt und dann als Bilddatei einsetzt?
Das kann der Pixeler.
Meist reicht es dann aus, zusätzlich nur einen "Font-Splitterchen", z.B. mit den Zahlen von 0-9 und ein paar Zeichen, z.B. ":" für eine Uhr oder "°C" für ein Thermometer im Angebot zu haben, das spart unter Umständen Speicherplatz auf dem Controller.
Du kannst dann zwar keine beliebigen Texte auf das Controller-Display zaubern, aber das ist häufig eben gar nicht nötig.

Außerdem sah ich mich gezwungen das Ding mal anzugehen, weil in Zukunft (oder schon in den neuen Win-Versionen) der Painter von Windows gar nicht mehr vorhanden sein soll, schade...

.....wird fortgesetzt.....