Mignon Gamekit Minimal Programm

Hier siehst du das Minimal Program-Gerüst. Damit ein Programm läuft, braucht es dieses Gerüst immer. Die Kommtarzeilen kannst du auch löschen, aber nicht die fett gedruckten Codezeilen!

```
#include <gamekit_2_1.h>

#include <avr/pgmspace.h>
// Dies ist eine Kommentarzeile. Dieser Text ist für den
// Programmierer. Er wird vom Programm nicht ausgeführt.
// Kommentarzeilen beginnen immer mit //


// Hier kannst du deine Variablen definieren


// Das setup() wird beim einschalten des Mignon Gamekits einmal
// ausgeführt

void setup(){

     gamekit.Begin();

}
// Im loop() läuft unser Programm, er wird immer bis zum Ende ausgeführt
// und beginnt dann wieder von vorne.

void loop() {
// Unter diesem Kommentar kommt dein Gamecode rein

}
```

Erklärung zur Gamekit Bibliotheks Referenz

Die Funktionen der Gamekit Bibliothek kannst du verwenden, um deine Spiele zu programmieren. Im Programm rufst du sie auf, indem du immer zuerst **`gamekit.`** Schreibst und dann die Funktion. Um zum Beispiel die erste Funktion `void` **`Begin()`** aufzurufen, schreibst du **`gamekit.Begin()`**

Die Schlüsselwörter `void`, `uint8_t`, `uint16_t`, `uint32_t` und `boolean` lässt du beim Funktions-Aufruf weg.

Diese Schlüsselwörter brauchst du, um eine Variable zu definieren, die das Ergebnis der Funktion abspeichert. Funktionen mit dem Schlüsselwort **`void`** geben kein Ergebnis zurück, du kannst also auch keinen Wert in einer Variablen abspeichern. Funktionen mit dem Schlüsselwort boolean geben nur 1 oder 0 zurück. Zwei Beispiele:
Die Funktion `void` **`set_pixel(Reihe, Spalte, Wert)`** setzt einen Pixel auf den Bildschirm. Für das Programm gibt sie kein Ergebnis zurück.
Die Funktion `uint8_t` **`get_pixel(Reihe, Spalte)`** liest den Wert eines Pixels auf dem Bildschirm ein. Sie gibt diesen Wert als Ergebnis an das Programm zurück. Diesen Wert kannst du in einer Variablen abspeichern. Das geht dann so:

Zuerst definieren wir die Variable:
`uint8_t` **`MeineVariable;`**
Dann können wir das Ergebnis der Funktion in diese Variable einlesen:
`MeineVariable = gamekit.get_pixel(Reihe, Spalte);`

Gamekit 2.1 Bibliotheks Referenz

`void` **`Begin();`**
Initialisiert und startet die Gamekit Library, die zuvor mit "`#include`" importiert wurde.

`void` **`Begin(gamekit_tilt_on);`**
Wenn der Kippschalter benutzt werden soll, muss die Gamekit Library so initialisiert werden

`void` **`set_pixel(Reihe, Spalte, Wert);`**
Setzt einen "Pixel" in Form einer leuchtenden Diode auf das Display. Dabei kann die Position mit den beiden Werten „`Reihe`" von 0–4 und „`Spalte`" 0-6 bestimmt werden. Der dritte Wert „Wert" steht für die Helligkeit zwischen 0 und 15. Höherer Werte lassen den Pixel blinken o.ä. .

`uint8_t` **`get_pixel(Reihe, Spalte);`**
Liest den Helligkeits-/Blink-Wert des Pixels in der „`Reihe`" und der "`Spalte`".

`void` **`load_image(Bild);`**
Lädt das Bild „`Bild`" auf das Display (siehe Beispiel „`load_image`").

`void` **`load_map((uint8_t *) name, BildSpalten, BildReihen, AnfangsSpalte, AnfangsZeile);`**
Lädt einen Ausschnitt eines großen Bildes auf das Display (siehe Beispiel „`load_map`").
„`BildSpalten`" und „`BildZeilen`" geben an, wieviele Spalten und Zeilen das gesamte Bild hat, das man laden will. Mit „`AnfangsSpalte`" und „`AnfangsZeile`" gibt man an, ab welcher Spalte und Zeile das Bild auf dem Bildschirm erscheinen soll.

`uint32_t` **`get_systemcounter();`**
Liest den Systemzähler. Der Systemzähler wird bei jedem Display neu zeichnen um eins erhöht.

`boolean` **`button_pressed(BUTTON);`**
Überprüft ob die Taste „`BUTTON`" gedrückt ist.
Die Konstanten sind:
`butt_UP` (Hoch)
`butt_DOWN` (Runter)
`butt_LEFT` (Links)
`butt_RIGHT` (Rechts)
`butt_FUNCA` (Funktion A)
`butt_FUNCB` (Funktion B)
Nur mit dem Kippschalter gehen auch folgende Konstanten:
`tilt_LEFT` (Gamekit nach links gekippt)
`tilt_RIGHT` (Gamekit nach rechts gekippt)

`void` **`wait_button_pressed(BUTTON);`**
Wartet solange bis die Taste „`BUTTON`" gedrückt ist.

`void` **`wait_button_released(BUTTON);`**
Wartet solange bis die Taste „`BUTTON`" nicht gedrückt ist.

`void` **`set_button_timing(Erster, Folgende);`**
Ändert die Wiederholgeschwindigkeiten der Taste (siehe Beispiel „`button`“). Mit dieser Funktion kann ein Knopf mehrere Male Wiederholt werden. Wie wenn man auf einer Tastatur einen Buchstaben länger gedrückt hält. „`Erster`“ gibt an, wie lange gewartet wird nach dem ersten drücken, bis die erste Wiederholung kommt. „`Folgende`“ gibt an, wie lange zwischen weiteren Wiederholungen gewartet wird.

`void` **`play_tone(Frequenz, Dauer, Lautstärke);`**
Spielt einen Ton der Tonhöhe „`Frequenz`“ über die Zeit von „`Dauer`“ (in Millisekunden) ab. „`Lautstärke`“ kann „`LOUD`“ für laut und „`SILENT`“ für leise sein.

`void` **`play_melody(name);`**
Spielt die Melodie „`name`“ ab. Siehe das Beispiel „`play_melody`“.

`uint16_t` **`get_current_melody_event();`**
Gibt an, welchen Ton der Melodie die „`play_melody`“-Funktion gerade abspielt.

`void` **`set_current_melody_event(nummer);`**
Springe zum Ton mit der Nummer „`nummer`“ in der gespielten Melodie.

`uint8_t` **`get_buttons();`** (für Fortgeschrittene)
Liest alle Tasten auf einmal als binäre Daten. Wenn ein Button nicht gedrückt ist, so liegt er auf 1. Wenn er gedrückt ist, so liegt er auf 0. Die Buttons sind auf folgende Bits gelegt:
Funktion A: Bit 0 (Least significant Bit), Hex-Wert 01
Funktion B: Bit 1, Hex-Wert 02
Hoch: Bit 2, Hex-Wert 04
Links: Bit 3, Hex-Wert 08
Runter: Bit 4, Hex-Wert 10
Rechts: Bit 5, Hex-Wert 20

Beispiele: kein Button gedrückt ergibt Hex-Wert 3F, Hoch und links gedrückt ergibt Hex-Wert 3F-04-08 = 33

`void` **`assign_pixelfunction(uint8_t, uint8_t(*)(uint8_t, uint8_t, uint32_t));`**
Sehr fortgeschrittene Programmierer können mit dieser Funktion einem Pixelwert (muss größer 15 sein) eine selbst geschriebene Funktion zuweisen.

www.mignongamekit.org