STM32

u23 2012

andy, darthrake

Chaos Computer Club Cologne e.V.
http://koeln.ccc.de

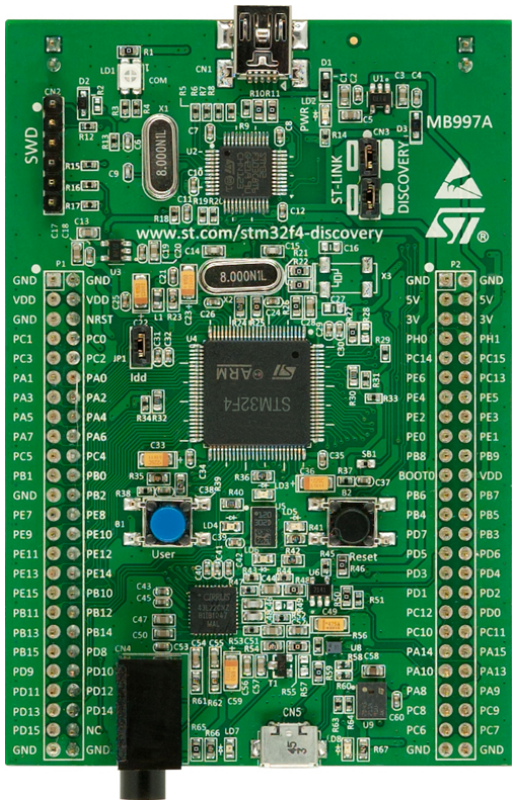
Cologne
2012-10-22

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

- Einführung
  - 2012-10-20 11:00 - C99
- Reguläre Termine
  - 2012-10-22 19:30 - STM32-Einführung (heute)
  - 2012-10-29 19:30 - Einführung in Pixelart
  - 2012-11-05 19:30 - VGA, Bilder
  - 2012-11-12 19:30 -
  - 2012-11-19 19:30 -
  - 2012-11-26 19:30 -
  - 2012-11-29 19:30 - OpenChaos

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

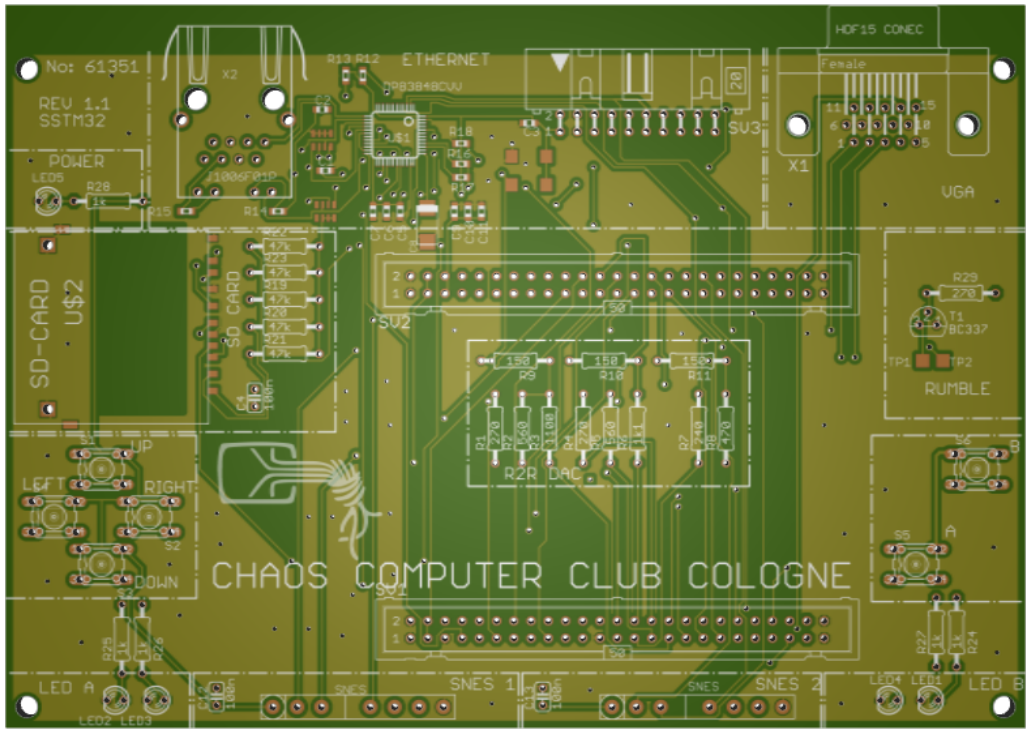
STM32



- STM32F407VGT6
  32-bit ARM
  Cortex-M4F
- 1 MB Flash
- 192 KB RAM
- JTAG via
  ST-Linkv2
- USB OTG
- 100pin LQFP

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Expansionboard



andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Expansionboard

- Unser Expansionboard
- VGA
- Buttons
- 2x SNES-Controller Ports
- Vibrationsmotor
- SD-Karte
- Optional Ethernet (zum Bestücken oder Dranstecken)
- Hängt leider bei den Chinesen :(

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Unsere Library

- Abstraktionslayer für die Hardware
- Zu finden unter git://github.com/cccc/U23-Library.git
- In der VM auf dem Desktop

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Ordnerstruktur

- applications/ - unbenutzt
- firmware/ - Eure Applikationen für das Board
- libs/ - Libraries
- build/ - Makefiles für das Buildsystem
- utils/ - Utilities (ImageConverter etc.)

Neues Projekt anlegen

- Im Ordner *firmware/* einen Ordner komplett kopieren
- Makefile des neuen Ordners bearbeiten und *TARGET* ändern
- Im Makefile in *firmware/* selbst den neuen Ordner hinzufügen
- Im neuen Projekt anfangen zu arbeiten

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Kompilieren und Flashen

- Im Stammordner reicht ein *make* um alles zu kompilieren
- *make upload* lädt die Firmware gametest auf euer Board
- Mit *make upload-firmwarename* oder *make upload FIRMWARE=firmwarename* ladet ihr eine bestimmte Firmware auf das Board

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Debuggen via gdb

```
$ st-util [19:25:24]
2012-10-22T19:25:55 INFO src/stlink-usb.c: -- exit_dfu_mode
2012-10-22T19:25:55 INFO src/stlink-common.c: Loading device parameters....
2012-10-22T19:25:55 INFO src/stlink-common.c: Device connected is: F4 device, id 0x10016413
2012-10-22T19:25:55 INFO src/stlink-common.c: SRAM size: 0x30000 bytes (192 KiB), Flash: 0x100000 bytes (1
Chip ID is 00000413, Core ID is 2ba01477.
KARL - should read back as 0x03, not 60 02 00 00
init watchpoints
Listening at *:4242...
```

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

```
$ arm-none-eabi-gdb gametest.elf
(gdb) target extended-remote :4242
Remote debugging using :4242
Reset_Handler () at src/Startup.c:23
23 for(uint32_t *dest=_data;dest<_edata;dest++) *dest=*src++;
(gdb) c
Continuing.
^C
Program received signal SIGTRAP, Trace/breakpoint trap.
SD_Detect () at src/sdcard.c:515
515 {
(gdb) bt
[\char "2026\relax ]
```

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Ein neues Spiel

```
#include <game/Game.h>

void Init(struct Gamestate*);
void Update(uint32_t);
void Draw(Bitmap* surface);

Gamestate InitState =
{ Init, NULL, NULL, Update, Draw };
Game* TheGame = &(Game) {&InitState};

void Init(struct Gamestate* state) { }
void Update(uint32_t delta) { }
void Draw(Bitmap* surface) { }
```

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Ein neues Spiel

- Init() wird ein mal beim Start aufgerufen
- Update() wird für jeden Frame aufgerufen, um Spielelogik zu implementieren
- Draw() wird für jeden Frame aufgerufen, um Inhalt zu malen

LEDs

- LEDs kann man mit *SetLEDs(int);* setzen
- Parameter ist eine Bitmaske
- LED 1 und 4 einschalten: (1«0) | (1«3)

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Accelerometer

- Ein Accelerometer misst auf 3 Achsen (X, Y und Z) die Beschleunigung des Board relativ zur Erde
- Zu gut Deutsch: Wie liegt das Board grade im dreidimensionalem Raum
- 3 Werte: Für jede Achse einen

Accelerometer Demo-Code

```
#include <game/Game.h>

void Init(struct Gamestate*);
void Update(uint32_t);
void Draw(Bitmap* surface);

Gamestate InitState =
{ Init, NULL, NULL, Update, Draw };
Game* TheGame = &(Game) {&InitState};

void Init(struct Gamestate* state) {
InitializeAccelerometer();
printf("Init Accelerometer: %s\r\n", PingAccelerometer() > 0 ? "OKAY" : "FAILED");
CalibrateAccelerometer();
}

void Update(uint32_t delta) {
int8_t components[3];
ReadCalibratedAccelerometerData(components);
printf("x: %d y: %d z: %d", components[0], components[1], components[2]);
}

void Draw(Bitmap* surface) { }
```

Debug Output

- Ist noch nicht fertig *hust*
- Schlagt den Andy dafür
- Gedacht war USB-Debugging mit einem kleinen Hilfsutility, dass einem eine Konsole gibt
- Momentan haben wir nur Serial-Ouput und 2 Serial-Adapter mit denen wir rumgehen können

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Debug Output

- Debugging einschalten mit *EnableDebugOutput(DEBUG_USART);*
- Danach kann man *printf()* benutzen

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

Aufgaben

- Besorgt euch die Library
- Kompiliert sie ein mal
- Legt ein neues Projekt an
- Lasst LEDs blinken

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

git

- git ist ein Versionskontrollsystem
- Verwaltet alle Arten von Quellcode
- Lässt euch Änderungem, die ihr an Quellcode gemacht habt, verteilen
- Wir benutzen es für unsere Software
- Hier nur kurze Einführung
- Im Wiki stehen zwei längere Howtos:
- http://try.github.com/
- http://githowto.com/

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

git clone <url>

- Lädt das Repository vom Server und legt euch eine lokale Kopie an

```
~ git clone git://github.com/cccc/U23-Library.git
Cloning into 'U23-Library'...
remote: Counting objects: 971, done.
remote: Compressing objects: 100% (713/713), done.
remote: Total 971 (delta 455), reused 739 (delta 223)
Receiving objects: 100% (971/971), 1.50 MiB | 650 KiB/s, done.
Resolving deltas: 100% (455/455), done.
```

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

git init

- Erstellt euch ein neues lokales Repository

```
~ git init
Initialized empty Git repository in /home/andy/Desktop/test/.git/
```

git status

- Zeigt einem Dateistatus an

```
~ git status
# On branch master
#
# Initial commit
#
# Untracked files:
# (use "git add <file>..." to include in what will be committed)
#
# testfile
nothing added to commit but untracked files present (use "git add" to track)
```

git add <filename>

- Fügt eine Datei dem Index hinzu
- Damit weiß git, dass es diesen Inhalt der Datei beachten soll
- Muss für alle Dateien gemacht werden mit deren Inhalt man was gemacht hat
- Geht auch direkt auf ganzen Ordnern

```
~ git add testfile
```

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

git commit

- Fasst den aktuellen Index zusammen zu einem commit
- Commit = ein Änderungssatz zu einer bestimmten Sache
- Jeder Commit bekommt eine Commitmessage der ihn beschreibt, sodass andere Leute nachvollziehen können, was ihr gemacht habt

```
~ git commit -m "Neues file hinzugefuegt"
[master (root-commit) d9ca032] Neues file hinzugefuegt
 0 files changed
 create mode 100644 testfile
```

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

git rm <filename>

- Löscht eine Datei

```
~ git rm testfile
rm 'testfile'
```

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

git mv <source> <destination>

- Verschiebt eine eine Datei
- Wird auch zum Umbenennen verwendet

```
~ git mv testfile neuesfile
~ git status
# On branch master
# Changes to be committed:
# (use "git reset HEAD <file>..." to unstage)
#
# renamed: testfile -> neuesfile
#
```

git push

- Schiebt eigene Änderungen auf den Server
- Server ist nicht zwingend notwendig, aber praktisch um Änderungen zu verteilen
- Kann sein, dass es abgelehnt wird, dann einmal pullen und wieder pushen

git pull

- Holt Änderungen vom Server

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>

git log

- Zeigt euch alle Commits der Reihe nach an
- gitk ist das selbe in grafisch

andy <andy@koeln.ccc.de>, darthrake <darthrake@koeln.ccc.de>