Funktionsweise von I/O Komponenten

Überblick

Ein µC-System besteht aus mehreren Komponenten:

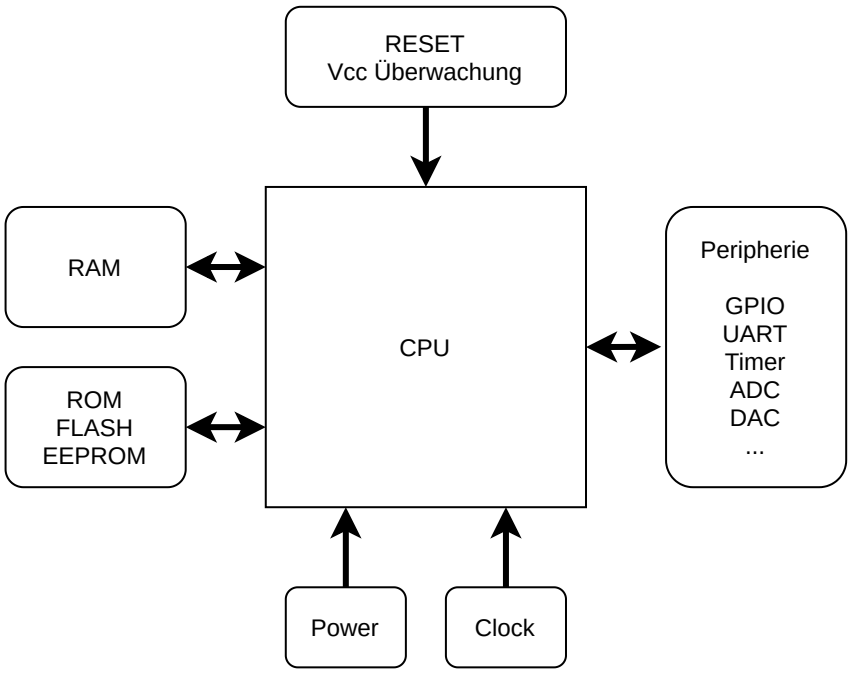

Fig. 1: µC Aufbau

Unterschied Prozessor - Controller

Ein Prozessor (etwa ein i7) braucht immer externe Komponenten (Speicher, …). Im Zuge der Digitalisierung von Steuerungen verbesserter Fertugungsmöglichkeiten wurden diese Komponenten zusammen mit dem Prozessor in einem Chip integriert. Das daraus entstehende System ist der Microcontroller.

Er hat …

- CPU, Speicher, Takterzeugung, Reset und Komponenten auf einem Chip
- spart somit Platinenfläche und
- ermöglicht kleinere Hardware-Designs

Im Idealfall braucht der µC nur noch eine Versorgungsspannung.

Stand der Dinge: SoC - System on a Chip
Die Idee besteht darin, alle Komponenten auf einem Chip unterzubringen (Ein-Chip-Computer als Basis für Handys).

CPU

Die CPU, der Microcontrollerkern oder einfach Core, ist das Herzstück des Controllers. Er beinhaltet

- Steuerbefehlsaufarbeitung
- ALU (Rechenwerk)
- Register
- Bussystem zur Ansteuerung aller Komponenten
- Interrupt-Controller

Er bestimmt mit seiner Bit-Breite, dem Befehlssatz und seiner Architektur die Leistungsfähigkeit des Gesamtsystems.
Beachte dabei die Vesorgungsspannung: 5V/3.3V/1.7V

Gängige Type:

Name	Architecture	Instruction Set	RAM	Clock
Phillips 8XC51-Familie	8Bit	CISC	64kB	bis 33MHz
Microchip PIC24F	16Bit	RISC	128kB	bis 70MHz
Cypress PSoC 5LP ARM CM3	32Bit	CISC	256kB	24 MHz

CISC … **C**omplex **I**nstruction **S**et **C**omputer
RISC … **R**educed **I**nstruction **S**et **C**omputer

Speicher

Um Daten und Programm speichern zu können, stehen unter anderem folgende Möglichkeiten zur Verfügung:

Programmspeicher

- Flash: wiederbeschreibbarer Speicher
- OTP: einmalig programmierbar, für Massenerzeugnisse ohne Update
- NOVRAM: nichtflchtiges RAM (Batterie gepuffert)

Frage: kann das Programm selber den Programmspeicher beschreiben?

Datenspeicher

- SRAM: statisches RAM
- EEPROM: dauerhafte Speicherung von Daten, etwa für Setup, langsam

Die Bitbreite hängt von der CPU und dem verwendeten Bus-System ab.
8/16/32 Bit

Überblick

Wir werden uns im folgenden mit diesen Komponenten beschäftigen:

- GPIO - allgemeine IO-Pins
- UART - asynchrone serielle Schnittstelle
- Timer - Zeitgeber bzw. Ereigniszähler
- ADC - analoge Signale einlesen
- DAC - analoge Signale ausgeben
- SPI - synchrone serielle Schnittstelle

GPIO

- generelle Aufgabe eines Pins
- Unterschied Analog / Digital-Pin
- Digital-Pin: Drive-Mode
- Read/Modify/Write
- Schaltpegel bei Digital-Pin

Probleme:

- Prellen beim Einlesen eines Tasters
- Achtung: PSoC5 LP
  - Kondensator an Pins 0/2, 0/3, 0/4, 15/2, 15/3, 15/4
  - Debugger an Pins 1/0, 1/1, 1/3
- Zwar lassen sich durch den Pin-Konfigurator logische Pins an beliebige physikalische Pins anschließen, doch hat das System einige dedizierte (OPV, UART/USB-Bridge)

UART - asynchrone serielle Schnittstelle

Aufgabe:

Serieller Datenaustausch zwischen zwei (oder mehreren) Systemen.

Beispiel:

- Anschluss eines Terminals an einem router
- Anschluss einer Bluetooth-Tastatur an einen PC

Wichtige Begriffe / Fragen:

- Half / Full Duplex Betrieb
- UART-Parameter (Baudrate, Parität, Anzahl Stopp-Bits)
- Frame
- wie synchronisiert der Sender den Empfänger?
- Probleme der Baudrate-Generierung
- Pegelanpassung an die physikalische Übertragunsleitung

Interner Aufbau

Die UART sendet die zu übertragene Daten mit Hilfe eines Schieberegisters vom Sender zum Empfänger Dort werde die serielle Daten wiederrum mittels Schieberegister einlesen.
Wichtig: beide Schieberegister müssen diese Taktung

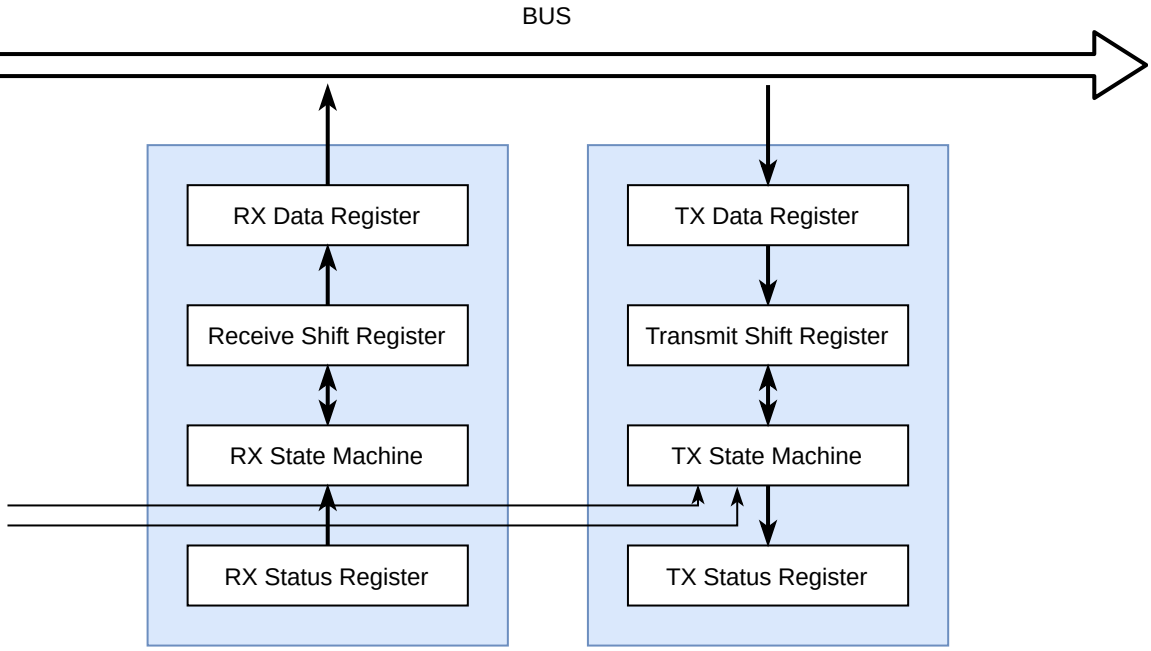

Fig. 2: UART Aufbau

Der Frame einer UART-Übertragung besteht aus folgenden Teilen

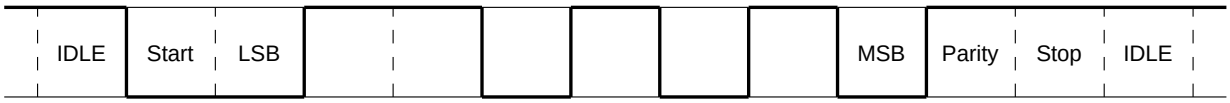

Fig. 3: Protokoll

1. Idle: im Ruhezustand immer 1
2. Um den Start einer Pbertraguns anzuzeigen: Flanke 1 $\Longrightarrow$ 0 und Beginn des Start-Bits
3. Datenbits (LSB bis MSB)
4. Optional: Parität
5. Stopp-Bit(s): immer 1 (somit ident zum Idle-Zustand)

Werden 2 Übertragungen znmittelbar hintereinander durchgeführt, folgt nach dem Stopp-Bit (1) sofort das Start-bit (0) mit der zugehörigen Flanke 1 $\Longrightarrow$ 0

Interfacing zur Aussenwelt

Das UART-Signal der CPU ist leistungsschwach und daher anfällig für Störungen. Da es direkt von einem CPU-Pin abgegriffen wird, ist dieser - und somit die gesamte empfindliche CPU - für zerstörende Signale (Überspannung, …) anfällig.
Daher wird zwischen dem Datenkabel, das die kommunizierenden Systeme verbindet, und der CPU ein Interface-Baustein geschalten.

Zwei Möglichkeiten:

- RS232: kann 2 Systeme miteinander verbinden und verwendet $\pm$ 10V
- RS485: kann bis zu 32 Systeme im Master / Slave-Betrieb verbinden und benutzt ein Differenzsignal

Timer

Allgemein

zyklische Ausführung, um in definierte Intervalle arbeiten zu können
(zb. 1x in der Sekunde die Uhr um eine Sekunde weitekr führen)

Realisation:

Zählregister, das mit einem definierten Takt getaktet wird.
Achtung: auf Grund der Breite des Zählregisters /8/16/32 Bit) sind dem System Grenzen gesetzt.

Takt

Dieser wird immer vom Systemtakt (Quart) abgelgeitet

SW-Implementierung

Entweder polling (Status-Register) oder Interrupt (ISR)

Capture

Mit Hilfe der capture-Funktion ermittelt das System, wann ein externes Signal statt gefunden hat.

- Timer started bei 0 (etwa mit Hilfe eines Triggers)
- wird bei einem dedizierten Eingangspin eine ensprechende Flanke erkannt, wird der aktuelle Wert des Zählers gespeichert.

Beispiel:

- Takt: 1µs
- nach dem Starten des Timers wurde beim Zählstand von 103 am Capture-eingang eine positive Flanke registriert und der Zählstand gespeichert
- Ergebnis: die Flanke trat nach 103µs auf

Counter

Allgemein

Asfinag würde David nicht anstellen (mentale Probleme).

Takt

Dieser wird of auch extern realisiert.