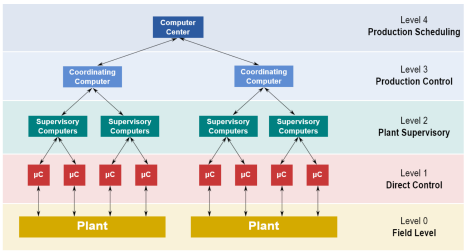
1 SW1 Software und tools

1.1 Design und Architektur

- **Mikrocontroller:** Verwendung bei geringem Kosten und Stromverbrauch. Eignet sich für integrierung auf PCB
- **FPGA:** Verwendung wenn mehr Leistung gewünscht als bei μC und man Funktionen direkt in HW implementieren möchte (vgl. hoher Stromverbrauch)
- **Embedded Linux:** Verwendung wenn man Netzwerkstacks und Internet nutzen möchte. Bietet grosse Funktionalität. Nachteil; nicht gut für "harteËchtzeit Anwendungen (langer bootvorgang).
- **Host:** Verwendung bei grossen Systemen. Host ist ähnlich wie PC. Wird oft für GUI und SCADA (Control And Data Acquisition) verwendet.

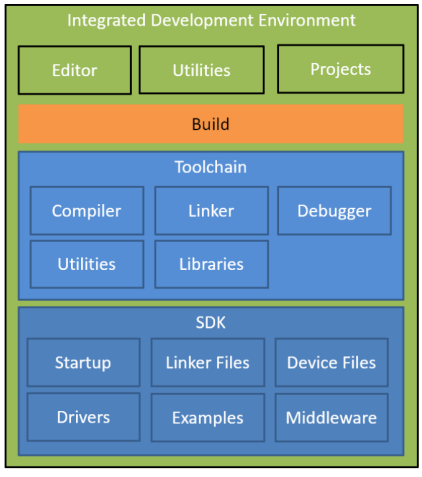
Oft werden Systeme als Kombination verschiedener Blöcke Realisiert.



1.2 Crossdevelopment

Wenn man nicht auf der selben Entität entwickeln kann spricht man von Host und Target. Auf dem host wird entwickelt, auf dem target ausgeführt. Auf dem Host wird für das Target entwickelt. Dazu nutzt man eine Toolchain.

- **Target:** Ist Zielsystem für das entwickelt wird (wofür)
- **Host:** Umgebung auf der entwickelt wird (womit)
- **Toolchain:** Besteht aus Compiler, linker, debugger, standard libraries und anderen Tools
- **Buildumgebung:** Steuert Toolchain und Übersetzungsvorgang. Wird oft mit makefiles gemacht.
- **IDE:** nicht zwingend notwendig



2 SW2 Software und Device Treiber

2.1 Device Driver

- **Interface:** Abstrahiert von Hardware. Sollte einfach und verständlich sein
- **Synchronisation:** Kann Synchron sein Gadfly, Polling oder asynchron mit Interrupts, Events oder Callbacks (was ist mit snchronisation gemeint dude)
- **Organisation:** Einfach: Eine Schnittstellendatei, eine Quelltextdatei (UART,SPI I^2C) Komplex: Mehrere Dateien mehrere Verzeichnisse
- **Konfiguration:** Treiber sollte konfigurierbar sein. Gängig ist durch Konfdatei, über Schnittstelle oder mit Makros. Siehe: https://mcuoneclipse.com/2019/02/23/different-ways-of-software-configuration/

2.2 File Formate

- **ELF/Dwarf:** ist ein Standard Format zur Beschreibung eines 'Executable' (Elf) zusammen mit der Debug (Dwarf) Information
- **S19 Motorola S-Record:** Repräsentation der Daten in textueller Form.
  Das S19 Format ist ein textuelles und zeilenorientiertes Format, welches 'S' Record ID , Länge , Adresse , Daten und eine Checksumme beinhaltet. Im Folgenden ein Beispiel:
  S1 13 7AF0
  0A0A0D0000000000000000000000000 61
- **Intel Hex:** Das Intel Hex Format ist auch ein text- und zeilenbasiertes Format, bei dem ein Start Code , Länge , Adresse , Typ , Daten und eine Checksumme verwendet wird. Nachfolgend auch hier ein Beispiel:
  : 10 0100 00
  21460136012147013600 7EFE09D21901 40
- **Binary:** Beim Binary Format sind keine zusätzlichen Informationen vorhanden. Bei diesem Format sind in der Datei einfach die 'rohen' Bytes abgespeichert.

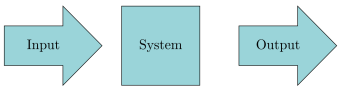
Gemeinsam zu Datei-Formaten von S19, Intel Hex und Binary ist es dass diese keine Debug Information enthalten (mussen).
Die benötigen Formate können entweder mit den GNU Werkzeugen direkt oder in Eclipse in einem Post-Build Step generiert werden. In der MCUXpresso Eclipse IDE können gleich mehrere Formate gleichzeitig in eine Post-Build Step erstellt werden

3 SW3 System

3.1 Systeme

Ein System ist eine Menge von interagierenden oder zusammenhängenden Einheiten, welche ein integrales Ganzes bilden.

- **Transformierende Systeme:** Verarbeitet Eingabestrom in Ausgabestrom. Verarbeitet Daten typischer weise kontinuierlich.



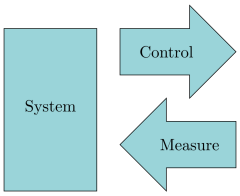
$$O(s) = P(I(s)) \tag{1}$$

Dabei ist $I()$ eine Eingabe Funktion, $I(s)$ ein Eingabestrom und $P(s)$ die Systemfunktion. Der Eingabestrom wird von der Systemfunktion verarbeitet und erzeugt einen Ausgabestrom $O(s)$. Def. "Multichannel System": Ein transformierendes System mit mehreren Ein- und Ausgabeströmen.

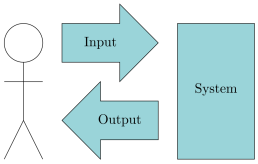
$$O_m(s) = P(I_n(s)) \qquad m,n \in \mathbb{R} \tag{2}$$

Eigenschaften: Verarbeitungsqualität (gut oder was?), Durchsatz, Systemausnutzung. Benötigen eine gewisse Menge an Speicher. Optimiert auf geringen speicherverbrauch. Konflikt mehr speicher schnellere verarbeitungszeit, aber teurer. Sind oft periodische Systeme (Datenlogger, lesen und verarbeiten Daten mit einer gewissen Periode).

- **Reaktive Systeme:** Unterscheiden sich zu transformierenden Systemen dadurch, dass Sie auf Ereignisse warten. Sind typischerweise Steuer- und Regelsysteme.



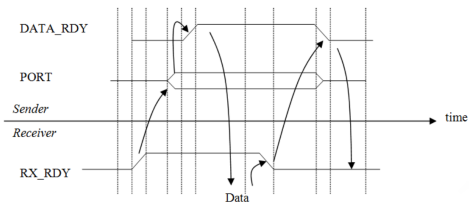
- **Interaktive Systeme:** Stellen Schnittstelle zu Benutzer her. Auf kurze Antwortzeit und weil teuer auf hohe auslastung optimiert.



4 Synchronisation

- **Systeme** Computer arbeiten in ihrer eigenen Zeitdomäne. Dies führt zu synchronisationsproblemen mit der Zeitdomäne der echten welt. Ein Echtzeitsystem muss die richtige Antwort zur Richtigen Zeit liefern.Timing = Zeitverhalten, wichtiger Begriff. Betrachtet man beispielsweise ein I/O System, stellt man fest, dass dieses nur korrekt funktioniert, wenn Daten zuerst eingelesen und dann gesendet werden. Sendet man zu frühm, gehen daten verloren. Um dies zu bewerkstelligen muss man den Ablauf synchronisieren. Es gibt für verschiedene Probleme/Systeme unterschiedliche Synchronisationsvarianten.
- **Handshaking** Synchronisation für Kommunikation. Ziel ist sicherzustellen, dass eine Meldung empfangen wird.
  1. Empfänger Signalisierten zuerst dass bereit Rx_RDY.
  2. Sender legt daten an PORT, Teilt mit dass bereit $DATA_RDY$
  3. Empfänger liest daten, signalisiert dass Daten empfangen RX_RDY
  4. Sender bestätigt dies mit $DATA_RDY$ um neuen Zyklus zu beginnen

  Dieser Prozess wird "Handshaking"genannt. Wird oft bei Kommunikationsprotokollen verwendet.




```
1  #define WAIT_TIME 10000
2  void read(void){
3    size_t i;
4
5    PORTB.DDRO = 1; /*pin B0 as output
         pin*/
6    for (i=0; 9<sizeof(buffer); i++) {
7      int j;
8      PORTB.B0 = 1; PORTB.B0 = 0; /*
           Pulse Handshake*/
9      j = WAIT_TIME;
10     while (j--){
11       --asm("nop");
12     }
13     buffer[i] = PORTA;/*was macht
           diese Zeile*/
14   }
15 }
```

i initialisieren für Arraylänge, also Bitlänge. PortB als Output definieren, Was macht dieses struct? In der For Schlaufe wird B0 kurz auf high und dann auf Low gesetzt. Es wird also ein sehr kurzer Puls ausgesendet (woher weiss ich wie lange der Puls andauert?).Danach wird dann jeweils eine kurze Zeit gewatet, Befor dann die Daten des Port A eingelesen werden.

```
1  void read(void){
```

```
  size_t i;

  PORTB.DDR1 = 1;
  PORTB.B1 = 1;
  for (i=0; i<sizeof(buffer); i++){
    PORT.B1=0;
    while (!PORT.B0){}
    while (PORT.B0){}
    buffer[i]=PORTA;
    PORTB.B1=1;
  }
}
```

- **Handshaking**

5 RTOS