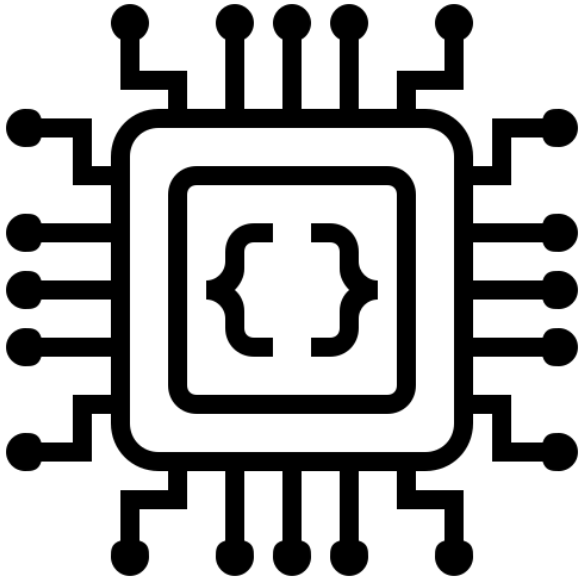
Hardwarenahe Programmierung

HTL ST. PÖLTEN



Quelle: I Putu Kharismayadi on www.flaticon.com

Function Pointer

HTL ST. PÖLTEN

- Pointer können auf (Anfangs-) Adresse einer Funktion zeigen
- Syntax: `<Rückgabetyp> (*<Pointername>) ([Parametertyp Parametername]);`

- Beispieldeklaration:
```
int (*funcPtr) (int param);
```
- Beispielzuweisung:
```
funcPtr = meineFunktion;
```
- Beispielaufruf:
```
(*funcPtr) (wert);
```
Oder:
```
funcPtr(wert);
```

Function Pointer Beispiel

HTL ST. PÖLTEN

```
01. #include <iostream>
02.
03. int quadrat(int wert) {
04.     return wert * wert;
05. }
06.
07. int main() {
08.
09.     int wert = 10;
10.
11.     int (*qPtr) (int);
12.     qPtr = quadrat;
13.
14.     std::cout << "Quadrat von " << wert << " = " << qPtr(wert) << std::endl;
15. }
```

Function Pointer als Parameter

HTL ST. PÖLTEN

- Function Pointer können auch als Parameter in Funktionen verwendet werden

```
void qsort(void *_Base, size_t _NumOfElements, size_t _SizeOfElements, int (*_PtFuncCompare)(const void *, const void *))
```

- Beispiel QuickSort

```
01. #include <iostream>
02.
03. int vergleich(const void* a, const void* b) {
04.     return (*(int*) a - *(int*)b);
05. }
06.
07. int main() {
08.     int arr[] = { 65, 27, 4, 1, 70};
09.     int len = sizeof(arr)/sizeof(arr[0]);
10.
11.     qsort(arr, len, sizeof(int), vergleich);
12.
13.     for (auto n : arr) {
14.         std::cout << n << " - ";
15.     }
16. }
```

Function Pointer Anwendung

HTL ST. PÖLTEN

- Verwendung als Callback Function → Funktion die aufgerufen werden soll, wenn Aktion abgeschlossen ist
- Anwendung zum Beispiel bei ISR:

```
void timerAttachInterrupt(hw_timer_t * timer, void (*userFunc)(void));
```

- Eigene ISR kann als Parameter verwendet werden
- Funktion: `void IRAM_ATTR meineISR ()`

ISR

HTL ST. PÖLTEN

- Interrupt Service Routine → ausgeführt bei jedem Interrupt
- Soll möglichst schnell durchlaufen werden → daher wenig Code
- IRAM_ATTR gibt an, dass die Funktion im RAM abgelegt werden soll und damit schneller geladen werden kann

Timer & Interrupts – Funktionen

HTL ST. PÖLTEN

- Timer aufsetzen:

```
hw_timer_t * timerBegin(uint8_t num, uint16_t divider,
bool countUp)
```

- Parameter:
  - num → Timernummer (0-3 bei ESP32)
  - divider → prescaler (2 bis 65536)
  - countUp → hoch, bzw. runterzählen
- Beispiel: `timer = timerBegin(0, 80, true);`

Timer & Interrupts – Funktionen

HTL ST. PÖLTEN

- Interrupt an Timer binden:

```
void timerAttachInterrupt(hw_timer_t *timer, void
(*fn)(void), bool edge)
```

- Parameter:
  - timer → zuvor erzeugter Timer
  - *fn → eigene ISR
  - edge → reagiert auf steigende Flanke
- Beispiel: `timerAttachInterrupt(timer, meineISR, true);`

Timer & Interrupts – Funktionen

HTL ST. PÖLTEN

- Wert einstellen bei dem Interrupt ausgelöst werden soll

```
void timerAlarmWrite(hw_timer_t *timer, uint64_t
alarm_value, bool autoreload)
```

- Parameter:
  - timer → zuvor erzeugter Timer
  - alarm_value → Timerwert bei dem Interrupt ausgelöst werden soll
  - autoreload → Timer wird auf 0 zurückgesetzt
- Beispiel: `timerAlarmWrite(timer, 1000000, true);`

Timer & Interrupts – Funktionen

HTL ST. PÖLTEN

- Timer aktivieren

```
void timerAlarmEnable(hw_timer_t *timer)
```

- Parameter:
  - timer → zuvor erzeugter Timer
- Beispiel: `timerAlarmEnable(timer);`

Beispiel Blinkprogramm

HTL ST. PÖLTEN

```
01. #include <Arduino.h>
02.
03. #define ON_BOARD_LED                    5
04. #define LED_ON                          LOW
05. #define LED_OFF                         HIGH
06.
07. bool ledState = false;
08. hw_timer_t* timer = NULL;
09.
10. volatile bool onesec = false;
11.
12. void IRAM_ATTR myISR();
13.
14. void setup() {
15.   // put your setup code here, to run once:
16.   pinMode(ON_BOARD_LED, OUTPUT);
17.   digitalWrite(ON_BOARD_LED, LED_ON);
18.
19.   Serial.begin(115200);
20.   Serial.println("- starting up -");
21.
22.   timer = timerBegin(0, 80, true);
23.   timerAttachInterrupt(timer, myISR, true);
24.   timerAlarmWrite(timer, 1000000, true);
25.   timerAlarmEnable(timer);
26. }
```

```
27.
28. void loop() {
29.   if (onesec) {
30.     onesec = false;
31.     if (ledState)
32.       digitalWrite(ON_BOARD_LED, LED_ON);
33.     else
34.       digitalWrite(ON_BOARD_LED, LED_OFF);
35.
36.     ledState = !ledState;
37.     Serial.println("One second passed");
38.   }
39. }
40.
41. void IRAM_ATTR myISR() {
42.   onesec = true;
43. }
```