

A. Neyenhuys, J. Verrieth, J. Scholz

# Kommunikation über die USB-Schnittstelle



## Projekte mit dem IOW28 Evaluation-Board:

Grundlagen der USB-Schnittstelle, Entwurf eines Entwicklungsboards und Anwendungen in Visual C#



Name:	
Klasse:	

Berufskolleg Kleve  
Stand 02-2021

# Inhalt

Allgemeine Informationen .....	4
IO-Warrior - Steht für "Universal" in USB.....	4
Überblick der IO Warrior28 Features .....	5
Grundlagen der USB Schnittstelle .....	6
Die USB - Schnittstelle.....	6
Kommunikation parallel / serielle Schnittstellen .....	6
Prinzip der seriellen Schnittstelle .....	6
Prinzip der parallelen Schnittstelle.....	7
<i>Arbeitsauftrag 1.0:</i> .....	7
Der USB Hub .....	7
Aufbau der Verbindung / USB Kabel .....	8
<i>Arbeitsauftrag 1.1:</i> .....	8
<i>USB Kabel Grundprinzip</i> .....	8
USB – Typen von Steckern/Buchsen.....	9
<i>Arbeitsauftrag 1.2:</i> .....	9
Das BUS USB Protokoll.....	10
Aufbau eines Entwicklungsboards .....	11
Das Pinout vom IOW28 Evaluation-Board .....	11
Was ist ein Mikrocontroller?.....	11
Erweiterung der Grundschialtung: Einlesen von Schaltzuständen und die Ansteuerung der Leuchtdioden .....	12
Ansteuern von Leuchtdioden: .....	14
Ermitteln Sie die Vorwiderstände zu den jeweiligen LEDs selbständig, indem sie die Datenblätter der LEDs studieren!.....	14
Bewertung und Kontrolle:.....	15
Aufbau des Entwicklungsboards .....	15
Einbau/Einsetzen des ICs (IOW28 Evaluation-Board).....	15
Elektrischer Test des Boards .....	15
Der IO-Warrior 28.....	16
Anwendungen mit dem IO-Warrior 28.....	21
Bewertung und Kontrolle:.....	29
Anwendungen mit dem IO-Warrior 28.....	29

# Vorwort

Dieses Arbeitsskript fasst die wesentlichen Inhalte des IO-Warrior Projektes zusammen, dass in der Oberstufe der Höheren Berufsfachschule für Informationstechnik am Berufskolleg Kleve durchgeführt wird.

Es gliedert sich in drei wesentliche Bereiche:

1. Grundlagen der USB-Schnittstelle (IT-Systemtechnik)
2. Entwurf eines Entwicklungsboards (Labormesstechnik, Fachpraxis)
3. Anwendungen mit dem IOW28 Evaluation-Board (System- und Anwendungssoftware, IT-Systemtechnik)

Jeder Bereich wird bewertet und auf einem Bewertungsbogen dokumentiert.

Tragen Sie Ihren Namen und Ihre Klasse ein und bewahren Sie dieses Skript, das Entwicklungsboard und Ihre Programme sorgfältig auf!

## **Das Skript kann, soll und wird nicht die Arbeit im Unterricht ersetzen!**

Darüber hinaus kann es durch Anpassungen der Unterrichtsinhalte aufgrund neuer Entwicklungen zu Verschiebungen kommen. Obwohl versucht wird, dieses Skript aktuell zu halten, kann das allerdings nur mit zeitlichen Verzögerungen garantiert werden.

Die Autoren sind natürlich über Hinweise auf Fehler und sonstige Anregungen dankbar.

Kleve, im Februar 2021

## Allgemeine Informationen

Kennen Sie das? Nur mal ein paar einfache Sachen an den Computer anschließen, vielleicht ein paar Relais, ein paar Schalter, ein kleines Display und.... Einfach?

Spätestens wenn der Computer, wie immer mehr neue Modelle, nicht mehr über serielle und parallele Schnittstellen verfügt, ist das alles andere als einfach. USB ist der logische Weg, aber der Aufwand ist für viele Anwendungen einfach zu groß. Ein Microcontroller mit USB-Schnittstelle, hunderte Seiten

Dokumentation, Entwicklungssysteme und dann erst die eigentliche Entwicklung.

USB-zu-Seriell-Adapter sind auch keine gute Lösung, da hier wenig Intelligenz in den Chips steckt, Schnittstellen wie SPI oder IIC müssen umständlich programmiert werden.

## IO-Warrior - Steht für "Universal" in USB

Der IO-Warrior28/IOW28 Evaluation-Board ist die fertige Lösung für viele dieser Probleme. Der IO-Warrior28 verfügt über 19 I/O-Pins die frei benutzbar sind. Jeder einzelne Pin kann wahlweise Ein- oder Ausgang sein. Um anspruchsvollere Dinge zu tun als nur ein paar Port Pins zu setzen oder zu lesen, verfügt der IO-Warrior über die "Special Mode Functions". Mit einem einfachen Befehl schaltet man eine dieser Funktionen ein, der IO-Warrior übernimmt dann direkt die Kontrolle über einige der I/O-Pins, um die gewünschte Funktion anzusteuern. Der IO-Warrior28 unterstützen den IIC-Bus, die Ansteuerung alphanumerischer LCD-Module und eine LED-Matrix mit bis zu 8x32 LEDs.

Zusätzlich hat der IO-Warrior28 einen 4 Channel 12 bit A/D Converter und einen IIC master up to 1 MHz.

Den IO-Warrior in eigene Software einzubinden ist leicht.

Unter Windows (7/8/10) ermöglicht eine Library einfachen Zugriff von vielen Programmiersprachen. Beispiele für C#, C++, VisualBasic und Delphi sind im SDK enthalten. Die IO-Warrior werden von Windows (ab 2000), Linux (ab Kernel 2.6) und MacOSX (ab MacOS 9) unterstützt. Für die gängigen Programmiersprachen (C/C++, C#, Delphi, Java) existieren Beispiele und API-Schnittstellen für die schnelle und einfache Anwendung. Für MacOS X bieten wir zusätzlich die Unterstützung von "AppleEvents". Kassenschublade von FileMaker aus öffnen? - Kein Problem! Für ein möglichst sorgenfreies Entwickeln mit dem IO-Warrior-Chip werden wir ein Starterkit entwickeln, dass bereits alle zum Betrieb notwendigen Komponenten und ein paar externe Bauteile für

👉 Link:

<https://www.codemerics.com/de/>

Hier finden Sie Informationen zum IO  
Warrior und kostenfreie Downloads  
von Dokumentationen etc. ....

[illegible]

erste Funktionstests auf einer Platine mit Lochrasterfeld vereinen. Das ist in den nächsten Kapiteln beschrieben.

## Überblick der IO Warrior28 Features

Überblick der Funktionen des IO Warrior28 Bausteins.



### *IOW28 Evaluation-Board*

- *Full Speed USB*
- *19 I/O Pins, typ. 1000Hz Leserate*
- *I2C Master Funktion, 10, 50, 100, 400 oder 1000kHz*
- *A/D-Wandler, 4 Kanäle, 12 Bit, interne Referenz für  $\pm 20$  mV Genauigkeit*
- *Ansteuerung von diversen Displaymodulen mit HD44780 kompatibelem Protokoll*
- *Erweiterter Temperaturbereich: -40 bis +85C*
- *SO24 Modul für begrenzte Rückwärtskompatibilität mit IOW24*
- *Kann mit Stiftleisten im DIL28 Formfaktor versehen werden*
- *Benutzt Systemtreiber von Windows, MacOSX. und Linuxtreiber*



---

---

---

---

---

---

---

---

---

---

## Grundlagen der USB Schnittstelle

In diesem Kapitel werden die Grundlagen der USB (Universal Serial Bus) Schnittstelle beschrieben. Zusätzlich wird noch eine praktische Messübung bearbeitet, die aktiv durchgeführt wird.

### Die USB - Schnittstelle

Neben den PCI- Bus und PCI Express eignet sich die USB Schnittstelle sehr gut für den Anschluss von E/A Geräte. Der USB Bus ist serielles Bussystem zur Verbindung eines Computers mit externen Geräten. Heute sind sogar langsame E/A Geräte wie Tastatur oder Maus über USB am Computer angeschlossen. Die Spezifikationsgruppe die sich ab 1993 um die USB Schnittstelle bemüht, hat unter andere folgende 3 Hauptziele immer beachtet:

- Benutzer sollen keine Schalter und Brücken auf Platinen oder Geräten einstellen müssen.
- Benutzer sollen das Gerät ohne das Gehäuse zu öffnen betreiben können und ohne Neustart verwendet werden.
- Es soll nur eine Kabelart für Anschluss und Stromversorgung verwendet werden.

### Kommunikation parallel / serielle Schnittstellen

Um den PC für eigene Elektronikexperimente zu verwenden, wird eine Schnittstelle zur Außenwelt bzw. zur eigenen Elektronik benötigt:



Es gibt eine parallele und serielle Standard Schnittstelle. Bei der seriellen Schnittstelle werden die Daten seriell übermittelt, d. h., die kleinsten Informationseinheiten werden über eine Signalleitung nacheinander übertragen.



Prinzip der seriellen Schnittstelle

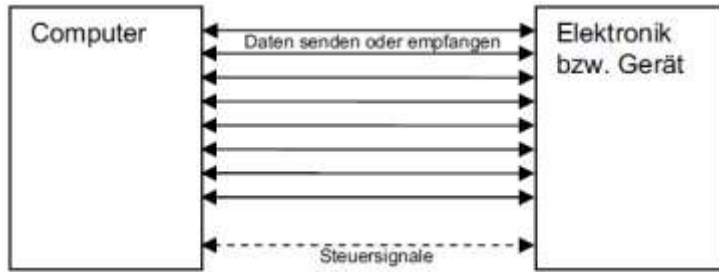
🔗 Link:

<http://www.usb.org/developers/docs/>

Hier finden Sie die Spezifikationen der USB Schnittstellen als kostenfreie Downloads etc. ....



Bei der parallelen Schnittstelle können bis zu 8 Zustände gleichzeitig auf 8 Datenleitungen übermittelt werden. Wie bei der seriellen, gibt es auch bei der parallelen Schnittstelle zusätzliche Steuersignale.



## Prinzip der parallelen Schnittstelle

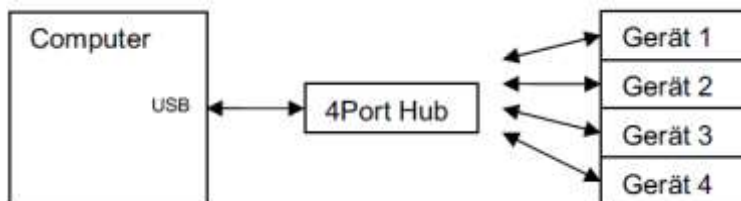
An die serielle oder parallele Schnittstelle eines PC wird üblicherweise ein Gerät angeschlossen.

## Arbeitsauftrag 1.0:

1. Benennen Sie je 3 Geräte die eine serielle Schnittstelle verwenden.
2. Benennen Sie je 3 Geräte die eine parallele Schnittstelle verwenden.
3. Klären Sie was für eine Art-Schnittstelle die USB Schnittstelle ist (seriell oder parallel)?

## Der USB Hub

Besitzt der PC nicht schon genügend USB-Schnittstellen, kann man USB einfach mit sogenanntem Hub auf die benötigte Anzahl erweitern.



## Prinzip USB Hub

Ausgehend vom USB-Host-Controller (in unserem Beispiel die USB-Schnittstelle des Computers) können an dem nachgeschalteten 4-Port-Hub weitere Geräte sternförmig angeschlossen werden. Ein Hub weist einen oder mehrere USB-Anschlüsse zum Anschluss weiterer USB-Geräte auf. Der Host-Controller im PC regelt die Kommunikation mit den am

[illegible]

USB-Host-Controller angeschlossenen USB-Geräten und dem Betriebssystem.

## Aufbau der Verbindung / USB Kabel

Die Verbindung zwischen dem Master und einem USB Gerät erfolgt durch eine 4 polige, geschirmte Leitung (wie unten in der Abbildung dargestellt).

Somit hat der physikalische USB-Anschluss vier Leitungen: zwei für die Stromleitungen (Masse und +5 Volt) und zwei Datenleitungen (D+ und D-).

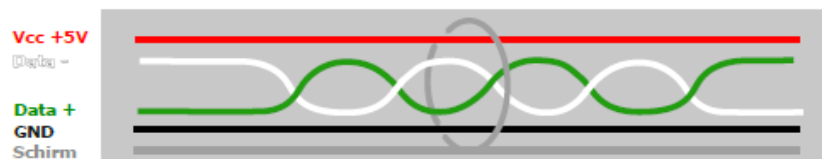


### Arbeitsauftrag 1.1:

1. Erstellen Sie eine Gegenüberstellung (wo die Anzahl der Geräte, Übertragungsrate und Max. Entfernung) häufig zu findender, serieller Schnittstellen. (mind. 4!)
2. Klären Sie den maximalen Stromverbrauch der USB Schnittstelle. Gibt es hier Unterschiede?
3. Gibt es eine Begrenzung der Leitungslänge? Wenn ja welche, und was kann man tun wenn die Leitung länger sein soll?
4. Klären Sie mit Hilfe der USB-Spezifikationen den Unterschied zwischen USB 1.0/1.1/2.0/3.0 und 4.0?
5. Was ist USB On-The-Go?

### USB Kabel Grundprinzip

Die beiden Datenleitungen sind verdreht um eingestrahlte Störungen weitestgehend zu verhindern bzw. zu eliminieren.



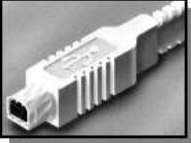



Aufbau einer USB Verbindung



## USB – Typen von Steckern/Buchsen

Für die Verbindungen von USB – Geräten kommen genormte Steckverbindungen zu Einsatz. Diese sind verpolungs- und vertauschsicher gestaltet. Es wird grundsätzlich zwischen zwei unterschiedlichen Bauarten unterschieden: in Richtung eines Hostcontrollers (USB Anschluss am PC) werden Stecker mit der Bezeichnung Typ-A verwendet, verbundene Geräte werden mit dem Stecker der Bezeichnung Typ-B angeschlossen.

Series "A" Connectors	Series "B" Connectors
<p>♦ Series "A" plugs are always oriented <b>upstream</b> towards the <i>Host System</i></p>  <p><b>"A" Plugs</b> (From the USB Device)</p>  <p><b>"A" Receptacles</b> (Downstream Output from the USB Host or Hub)</p>	<p>♦ Series "B" plugs are always oriented <b>downstream</b> towards the USB Device</p>  <p><b>"B" Plugs</b> (From the Host System)</p>  <p><b>"B" Receptacles</b> (Upstream Input to the USB Device or Hub)</p>

### Übersicht Typ-A / Typ-B

Dieser Vorgabe aus der Spezifikation wird allerdings nicht immer eingehalten. Neben den genormten Steckern und Buchsen gibt es auch Unterschiede bei den Übertragungsgeschwindigkeiten und den Übertragungsarten bei USB Geräten.

### Arbeitsauftrag 1.2:

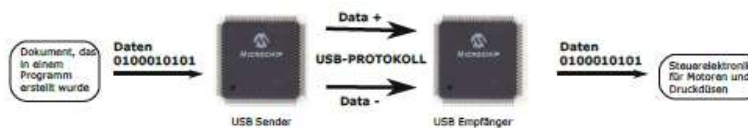
1. Erarbeiten Sie sich den Unterschied zwischen Lowspeed und Highspeed Übertragung bei USB Geräten.
2. Wie ist die USB Stromversorgung bei dem IOW28 gelöst worden? Wie kann dort Lowpower bzw. Highpower eingestellt werden?
3. Was sind USB Transfertypen? Erklären Sie die unterschiedlichen Typen.

## Das BUS USB Protokoll

Alle Bus-Transaktionen führen zur Übertragung von bis zu drei Paketen. Jede Transaktion beginnt, wenn der Host-Controller ein USB-Paket sendet, das die Art und Richtung der Transaktion sowie die Geräte-Adresse und Endpunkt-Nummer bestimmt. Dieses Paket wird als Token-Paket bezeichnet. Das angesprochene Gerät wird bestimmt, indem jedes angeschlossene Gerät die Adress-Felder dekodiert und sich im Falle der Übereinstimmung selbst selektiert. Bei einer bestimmten Transaktion können Daten entweder nur von Host an ein Gerät gesendet werden oder der Host kann nur Daten eines Gerätes empfangen.

Die Datenquelle sendet im Anschluss ein Datenpaket oder zeigt dem Empfänger an, dass keine Daten zu übertragen sind. Der Empfänger antwortet dann generell mit einem Handshake-Paket, welches anzeigt, ob die Übertragung erfolgreich war oder nicht.

Das USB-Transfer-Modell zwischen Datenquelle oder -ziel Host und einem Endpunkt am Empfänger wird als Pipe (Übertragungsstück) bezeichnet. Es gibt zwei Arten von Pipes: Stream- (Datenfluss-) und Message- (Nachrichten-) Pipes. Im Gegensatz zu Message-Pipes haben Stream-Pipes keine USB-definierte Datenstruktur. Außerdem hat eine Pipe weitere bezeichnende Eigenschaften wie Daten-Bandbreite, Transfer-Art und Endpunkt-Merkmale wie Puffer-Größe. Pipes entstehen, wenn ein USB-Gerät konfiguriert wird. Eine Message-Pipe -Kontroll-Pipe 0 - existiert, sobald das Gerät mit Strom versorgt wird und ermöglicht damit die Bereitstellung von Informationen zur weiteren Geräte-Konfiguration sowie Status- und Kontroll-Informationen.



### USB Protokoll (Sender/Empfänger)

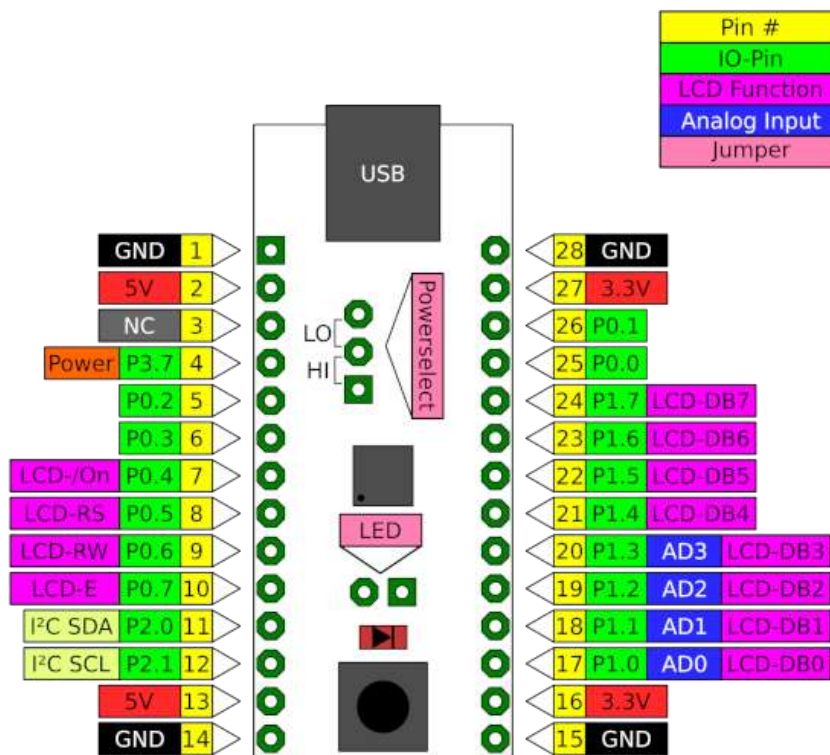
Bei der USB Schnittstelle werden die Daten eben nicht mehr direkt über die Schnittstellenelektronik an das angeschlossene Gerät übermittelt, mit Hilfe von einem Protokoll übertragen. Dies macht es nicht möglich, entsprechende Komponenten, wie Relais, LEDs, Leuchten usw. direkt über eine Steuerleitung mit dem PC zu verbinden bzw. anzusprechen. Das ist bei der parallelen Schnittstelle anders, hier ist das direkt möglich.



## Aufbau eines Entwicklungsboards

Nachfolgend wird Schritt für Schritt ein Entwicklungsboard für den IO-Warrior 28 entworfen. Der Aufbau der hier vorgestellten Schaltungen erfolgt auf einer Lochstreifenplatine. Die IC's werden alle gesockelt (also nicht direkt in die Platine eingelötet, sondern nach Fertigstellung der Lötarbeiten in die Sockel gesteckt).

### Das Pinout vom IOW28 Evaluation-Board



### Was ist ein Mikrocontroller?

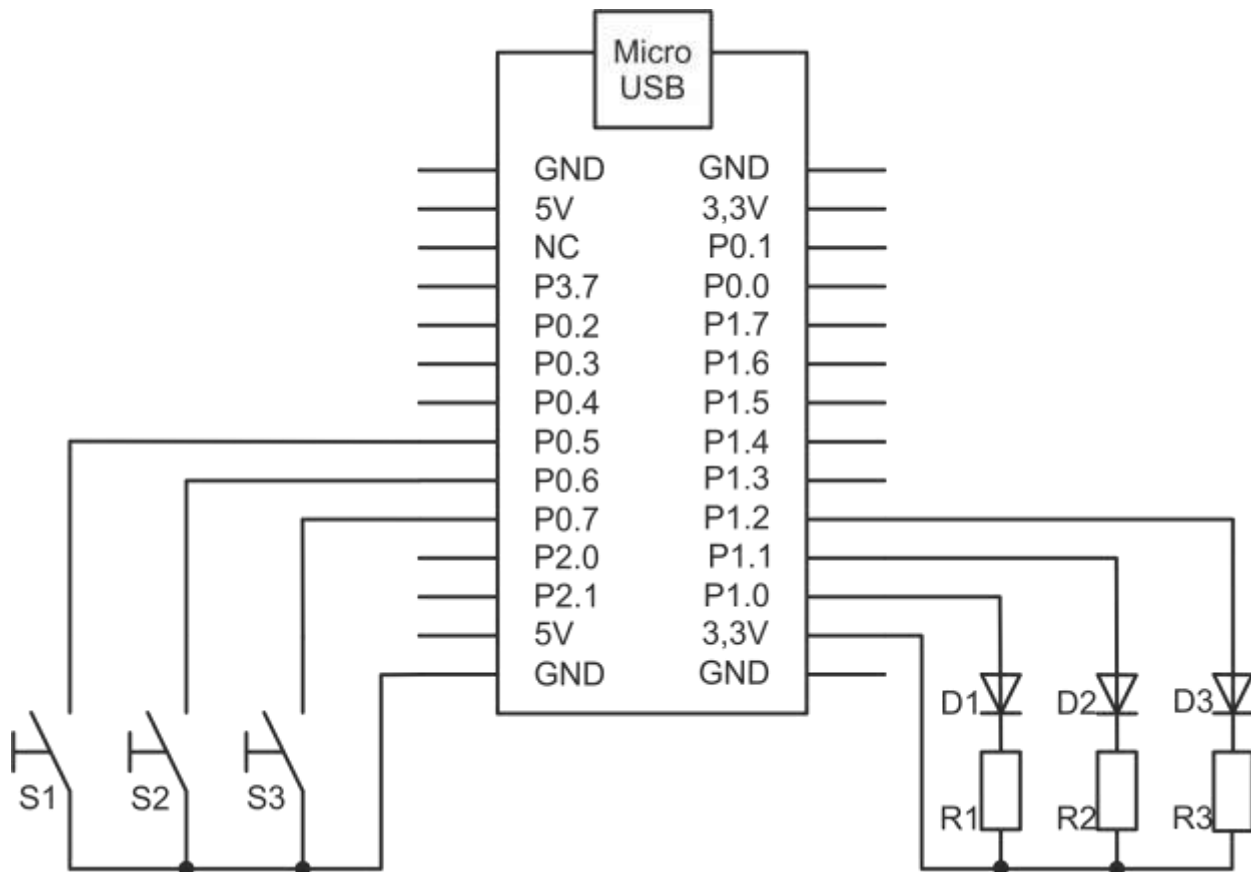
Ein Mikrocontroller (auch MC oder  $\mu C$ ) integriert mehrere Komponenten (Prozessor, Speicher,...) auf einem einzigen Baustein (Chip). Deshalb hat der Baustein nur wenige externe Zusatzkomponenten nötig, um genutzt zu werden. Er stellt dazu noch eine Reihe von Funktionen z.B I/O Pins oder BUS Systeme zur Verfügung. Der Mikrocontroller wird auch Einchip-Computer(Single-Computer).Mikrocontroller sind oft, z.B. im Smartphone einer Uhr, der Waschmaschine oder Autos im Einsatz.Ein Mikrocontroller besteht aus einem Kern und Peripheriekomponenten.Der Kern entspricht in etwa, was man bei einen PC als CPU bezeichnet. Enthält also u.a. ein Rechenwerk, Steuerwerk und Register.

Die Peripheriekomponenten des Controllers sind das Fenster zur Außenwelt. Durch Bereitstellung einer Reihe von Standardschnittstellen wird der Controller seiner eigentlichen Aufgabe gerecht. Nämlich Steuerungs-, Kontroll- und Kommunikationsaufgaben zu ermöglichen.

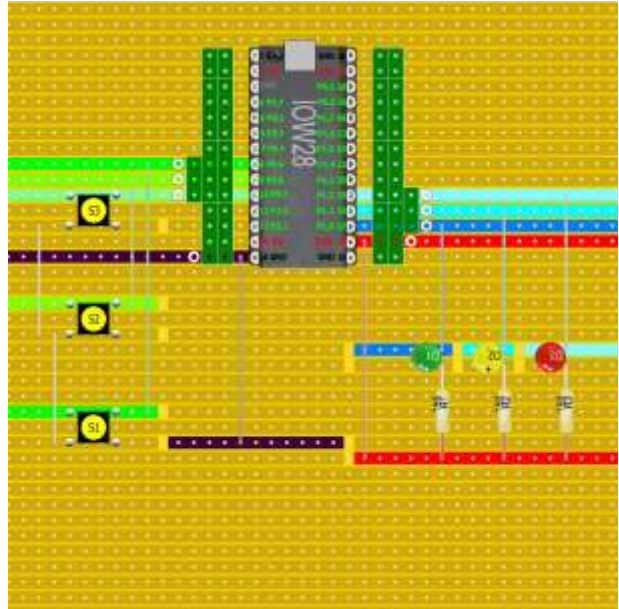
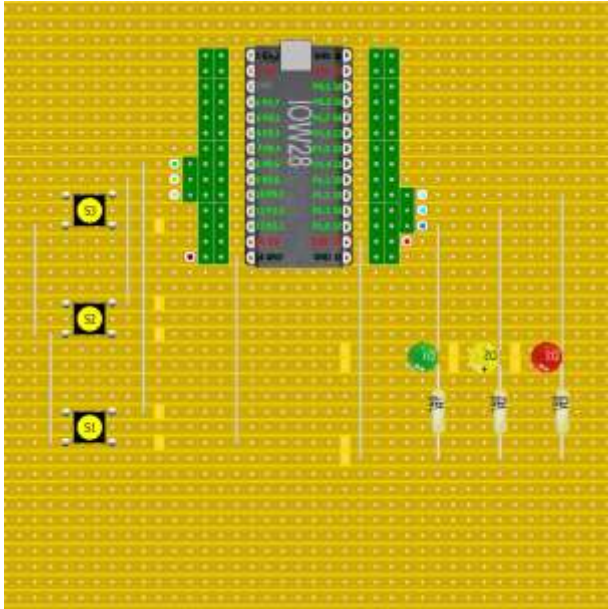
## Erweiterung der Grundschiung: Einlesen von Schaltzuständen und die Ansteuerung der Leuchtdioden

Folgende Bauteile werden nun zur Grundschiung des ergänzt:

- 3 Taster
- 3 Leuchtdioden (LEDs): rot, gelb, grün (5mm)
- 3 Widerstände 150  $\Omega$



So sieht die Bestückungsseite der Platine aus und der Aufbau- und Lötplan:



Der Plan ist folgendermaßen zu lesen:

- Sie schauen von der Bestückungsseite auf die Platine
- Die gelben Rechtecke sind die Leiterbahnunterbrechungen auf der Unterseite (Lötseite) der Platine
- Die unterschiedlichen Farben zeigen die Potentiale in der Schaltung.
- Achten Sie auf die korrekte Polung der Leucht-Dioden
- Beachten Sie die Einbaulage der Taster



## Ansteuern von Leuchtdioden:

Ermitteln Sie die Vorwiderstände zu den jeweiligen LEDs selbständig, indem sie die Datenblätter der LEDs studieren!

[illegible]

Denken Sie auch – sofern noch nicht geschehen – an die Abstandhalter an den Ecken der Platine!

## Bewertung und Kontrolle: Aufbau des Entwicklungsboards

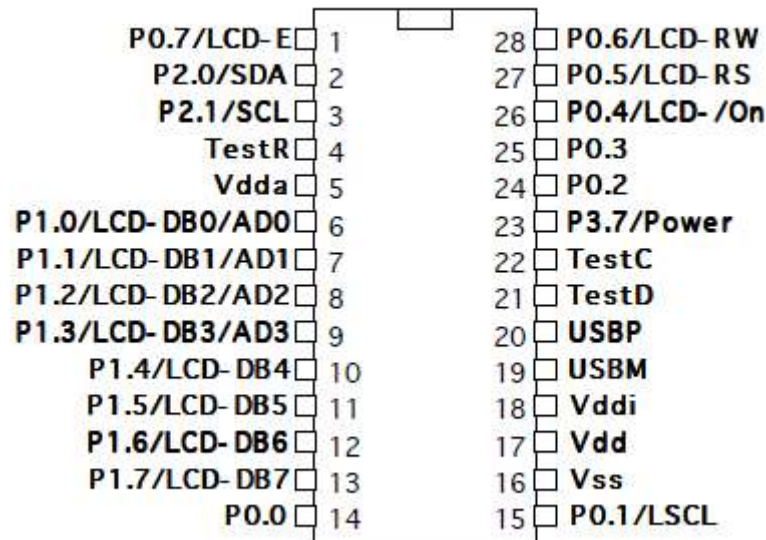
<b>Name</b>	
<b>Klasse</b>	



Teilauftrag	Punkte / Bewertung / Bemerkungen / Unterschrift
Die Grundsaltung Abgabetermin: _____	
Einbau/Einsetzen des ICs (IOW28 Evaluation-Board) Abgabetermin: _____	
Elektrischer Test des Boards  Abgabetermin: _____	

## Der IO-Warrior 28

Der für unsere Zwecke verwendete IO Warrior ist der IOW 28 (die 28 steht für die Anzahl der Pins des Bausteins). Hier die Pinbelegung des ICs:



Beschreibung der Pins:

Name	I/O	Type	Pins	Description
USBP, USBM	I/O	special	20, 19	USB differential data lines
P0.0, P0.1, P0.2, P0.3, P0.4, P0.5, P0.6, P0.7	I/O	I/O open drain, internal pullup	14, 15, 24, 25, 26, 27, 28, 1	First I/O port.
P1.0, P1.1, P1.2, P1.3, P1.4, P1.5, P1.6, P1.7	I/O	I/O open drain, internal pullup	6, 7, 8, 9, 10, 11, 12, 13	Second I/O port
P2.0, P2.1	I/O	I/O open drain, internal pullup	2, 3	Third I/O port. Also fast IIC port.
P3.7/Power	I/O	I/O open drain, internal pullup	23	Fourth I/O port P3.7 is used to determine power setting at power up.
Vss		power supply	16	Ground
Vdd, Vdda, Vddi		power supply	17, 5, 18	Supply voltage, connect to 3.3 V
TestR, TestD, TestC		special	4, 21, 22	Used during manufacturing, do not connect

Für die hier bearbeiteten Anwendungen werden die Pins 1, 6-15 und 24-28 (vorerst) nur als I/O-Ports behandelt.



## Erstinbetriebnahme des IOW28 Evaluation-Board

Um den Warrior in Betrieb zu nehmen, benötigen wir nur die USB-Buchse anzuschließen. Ab dann müssen wir entsprechend ein Programm mit C# entwickeln um einen ersten Test Software vs. Hardware durchzuführen.

Informationen zum Datenblatt finden Sie hier:

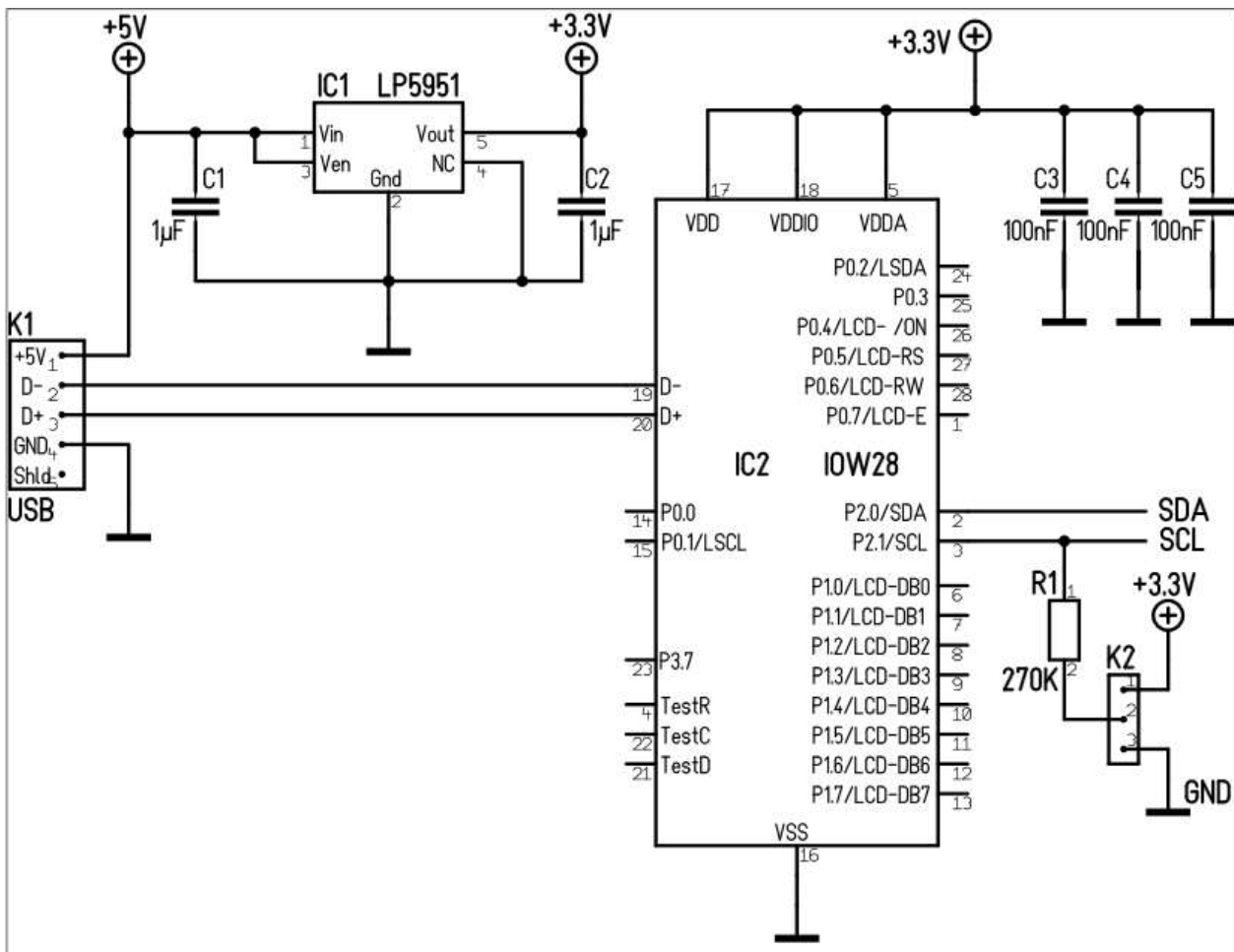
[https://www.codemercs.com/downloads/iowarrior/IOW28\\_Datasheet.pdf](https://www.codemercs.com/downloads/iowarrior/IOW28_Datasheet.pdf)

### IOW28 Evaluation-Board

#### Varianten

Typ	I/O Pins	LCD	I2C	SPI	ADC	Größe	Status
IO-Warrior28	19	✓	✓	—	4 Kanäle	DIL28 Wide	Active

Ebenso einen Basic Schaltplan wenn Sie kein IOW28 Evaluation-Board verwenden sondern nur den Chip des IOW28.



## Dynamische Laufzeitbibliotheken

Der Zweck von DLL-Dateien (Dynamic Link Library) ist, den von einer Anwendung (Programm) auf der Festplatte und im Hauptspeicher benötigten Speicherplatz zu reduzieren. Jeglicher Programmcode, der von *mehr als einer* Anwendung (*mehr als einem* Programm) benötigt werden könnte, wird deshalb in einer einzelnen Datei auf der Festplatte gespeichert und nur einmal in den Hauptspeicher geladen, wenn mehrere Programme dieselbe Programmbibliothek benötigen. So wird z.B. der „Datei Öffnen“-Dialog von nahezu allen Windowsprogrammen benutzt. Er ist in einer DLL-Datei abgelegt, der `comdlg32.dll`.

Beispielsweise ist folgender Code Inhalt eines Programms, das beim Compilieren eine DLL-Datei (`Addition.dll`) erzeugt:

```
DLL double ZahlenAddieren (double a, double b)
{
    return (a + b) ;
}
```

Wollen nun mehrere Anwendungsprogramme auf diese DLL zugreifen, muss sie in einem bestimmten Ordner des Betriebssystems liegen, nämlich `...\Windows\system32`. Alle Anwendungen, die auf eine DLL zugreifen, suchen automatisch dieses Verzeichnis. Der Zugriff auf die `Addition.dll` aus einem C#-Programm würde folgendermaßen erfolgen:

```
namespace DllImportBeispiel
{
    using System;
    using System.Windows.Forms;
    using System.Runtime.InteropServices;

    class Program
    {
        [DllImport("Addition.dll")]
        static extern double ZahlenAddieren(double a, double b);

        static void Main()
        {
            MessageBox.Show( ZahlenAddieren( 1, 2 ) );
        }
    }
}
```

In C# werden DLLs mithilfe des `DllImport`-Attributs eingebunden. Dazu ist der Namespace „`System.Runtime.InteropServices`“ nötig. Der Methodentyp wird als „`extern`“ angegeben und anschließend kann die Methode wie jede andere auch angesprochen werden.

## Die iowkit.dll

Die USB-Kommunikation des IO-Warriors wird vollständig von der vom Hersteller mitgelieferten DLL iowkit.dll übernommen. Einige in ihr enthaltenen Funktionen (Methoden) werden in der folgenden Tabelle erläutert:

Funktion	Erklärung
IowKitOpenDevice()	Die Funktion öffnet alle angeschlossenen IO-Warriors. Sie liefert einen Handle auf den ersten Warrior zurück.
IowKitCloseDevice(int iowHandle)	Die Funktion schließt den Warrior (in C# zwingend erforderlich!)
IowKitWrite(int iowHandle, int numPipe, ref byte buffer, int length)	Die Funktion schreibt Daten an den Warrior. Je nach Wert der Variablen numPipe werden IO-Pins oder Spezialfunktionen angesprochen.
IowKitRead(int iowHandle, int numPipe, ref byte buffer, int length)	Die Funktion liest Daten vom Warrior ein und speichert diese in einem Array. Wenn keine Änderungen der Eingänge vorliegen blockiert die Funktion das Programm. Möchte man Eingänge auslesen und das Programm nicht blockieren, so ist die Funktion IowKitReadNonBlocking zu verwenden.
IowKitReadNonBlocking(int iowHandle, int numPipe, ref byte buffer, int length)	Siehe IowKitRead
IowKitGetNumDevs()	Die Funktion gibt die Anzahl der angeschlossenen Warrior zurück. Vorher müssen diese natürlich über die entsprechende Funktion geöffnet werden.
IowKitGetDeviceHandle(int numDevice)	Die Funktion gibt eine Nummer (einen Handle) zurück, mit dem ein Baustein später identifiziert werden kann. Der Übergabeparameter entspricht der numerischen Reihenfolge des Bausteins.
IowKitGetProductId(int iowHandle)	Es wird die Produktnummer des Warriors zurückgegeben. Der dezimale Rückgabewert bei einem IO-Warrior 24 beträgt 5377, bei einem IO-Warrior 40 beträgt 5376 und bei einem IO-Warrior 56 beträgt er 5379.
IowKitGetSerialNumber(int iowHandle, ref int serialNumber)	Die Funktion übergibt den Handle und eine Referenz auf eine Integervariable. Die Seriennummer wird in serialNumber gespeichert und die Funktion gibt den Wert true zurück, sonst false.
IowKitSetTimeout(int iowHandle, int timeout)	Die Funktion setzt den Lese-Timeout des Warriors in Millisekunden. Wird diese Zeit überschritten, bricht der Lesevorgang ab.
IowKitCancelIo(int iowHandle, int numPipe)	Die Funktion bricht Lese- und Schreibvorgänge auf dem Warrior ab.

Die iowkit.dll, Datenblätter der jeweiligen Warrior und die Beschreibung der iowkit.dll stehen auf der Herstellerseite [www.codemerics.com](http://www.codemerics.com) zum Download bereit.



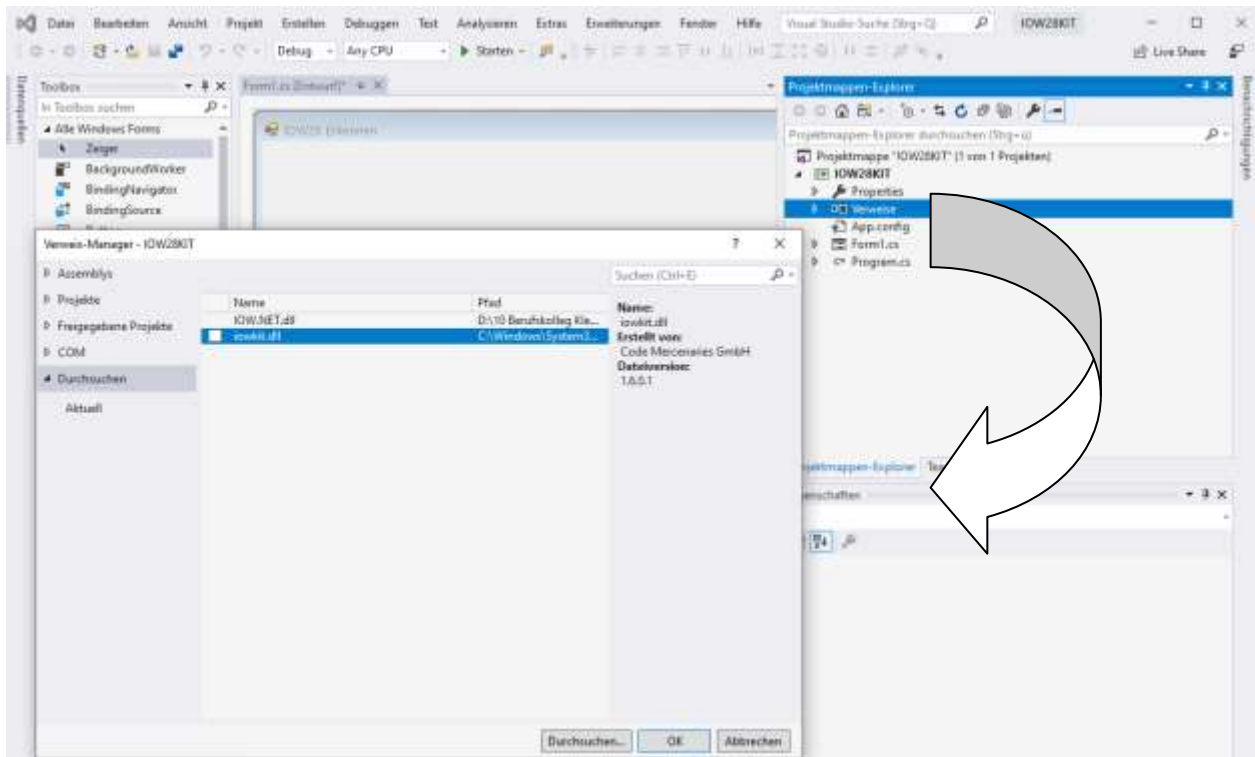
## Einbinden der iowkit.dll

Um nun das IO-Warrior Board als fertige Lösung für Parallel-I/O am USB-Bus zu nutzen müssen wir noch die iowkit.dll einbinden. Eine DLL kann nicht direkt ausgeführt werden. **Wir müssen für jedes C# Visual Studio Projekt im Debug- und Release Ordner die Datei iowkit.dll manuell dorthin kopieren.**

Stellen Sie sicher, dass die DLL am richtigen Speicherort liegt bevor Sie ihr Projekt ausführen.

Wenn Sie Administrator sind, dann können Sie auch den Verweis auf die DLL wie folgt durchführen:

- 1) Im Projektmappen- Explorer rechte Maustaste auf „Verweis“
- 2) Verweis hinzufügen auswählen
- 3) DLL im Verweis-Manager anklicken (ggf. durch „Durchsuchen“ auf dem Rechner auswählen)



### Hinweis

Eine DLL (Dynamic-Link Library) ist eine Bibliothek, die Code und Daten enthält, die von mehreren Apps verwendet werden können. Deshalb können Sie an Rechnern wo Sie Administrator sind auch die Datei hier ablegen ...Windows\system32.

## Anwendungen mit dem IO-Warrior 28

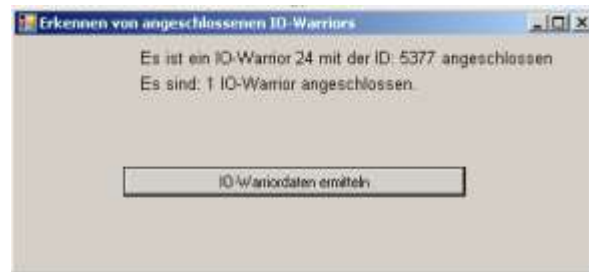
### Auftrag 1: Erkennen von angeschlossenen IO-Warriors

Es soll ein Programm geschrieben werden, dass einen oder mehrere angeschlossene IO-Warrior identifiziert.

Nach dem Programmstart soll folgendes Fenster angezeigt werden:



Nach dem Klick auf den Button sollen folgende Informationen ersichtlich sein:



Angezeigt werden sollen also:

- Art der oder des Warriors
- Die jeweilige ID
- Anzahl der Warrior

Das folgende Programmgerüst soll als Ausgangslage dienen:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Runtime.InteropServices;

namespace IOW_Identifizieren
{
    public partial class Form1 : Form
    {
        [DllImport("iowkit", SetLastError = true)]
        public static extern int IowKitOpenDevice();

        [DllImport("iowkit", SetLastError = true)]
        public static extern void IowKitCloseDevice(int iowHandle);

        [DllImport("iowkit", SetLastError = true)]
        public static extern int IowKitGetNumDevs();

        [DllImport("iowkit", SetLastError = true)]
        public static extern int IowKitGetDeviceHandle(int numDevice);
    }
}
```

```
[DllImport("iowkit", SetLastError = true)]
public static extern short IowKitGetProductId(int iowHandle);

public Form1()
{
    InitializeComponent();
}

public void Form1_Load(object sender, EventArgs e)
{
    // ... TO DO: Teilauftrag 6
}

private void button1_Click(object sender, EventArgs e)
{
    // ... TO DO: Teilauftrag 7
}

private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    // ... TO DO: Teilauftrag 8
}
}
```

Daraus ergeben sich folgende Teilaufträge:

1. Erstellen Sie ein neues C# -Windows-Forms-Projekt: „IOW\_Identifizieren\_Nachname“
2. Vergessen Sie nicht die iowkit.dll in das Debug Verzeichnis oder ...\\Windows\\system32 zu kopieren.
3. Fügen Sie den Namespace „System.Runtime.InteropServices“ hinzu.
4. Binden Sie die DLL`s ein, wie im Programmgerüst zu sehen.
5. Erstellen Sie das Formular: Sie benötigen nur zwei Labels und einen Button. Geben Sie den Labels und dem Button (über Eigenschaften) sinnvolle Namen (nicht wie im Programmgerüst oben)!
6. Nutzen Sie eine Funktion der iowkit.dll, die einen Handle auf den ersten Warrior zurückgibt und speichern Sie diesen Handle in einer Integervariablen int lowHandle. Öffnen Sie dann den angeschlossenen Warrior.
7. Ermitteln Sie die Anzahl der angeschlossenen Warriors. Ein Label kann nun fertig gestellt werden. Ermitteln Sie ein Handle zum Ansprechen des jeweilig angeschlossenen Warriors. Ermitteln Sie die Produktnummer(n) der Warrior. Entscheiden Sie nun, welcher Warriortyp angeschlossen ist. Das andere Label kann nun fertig gestellt werden.

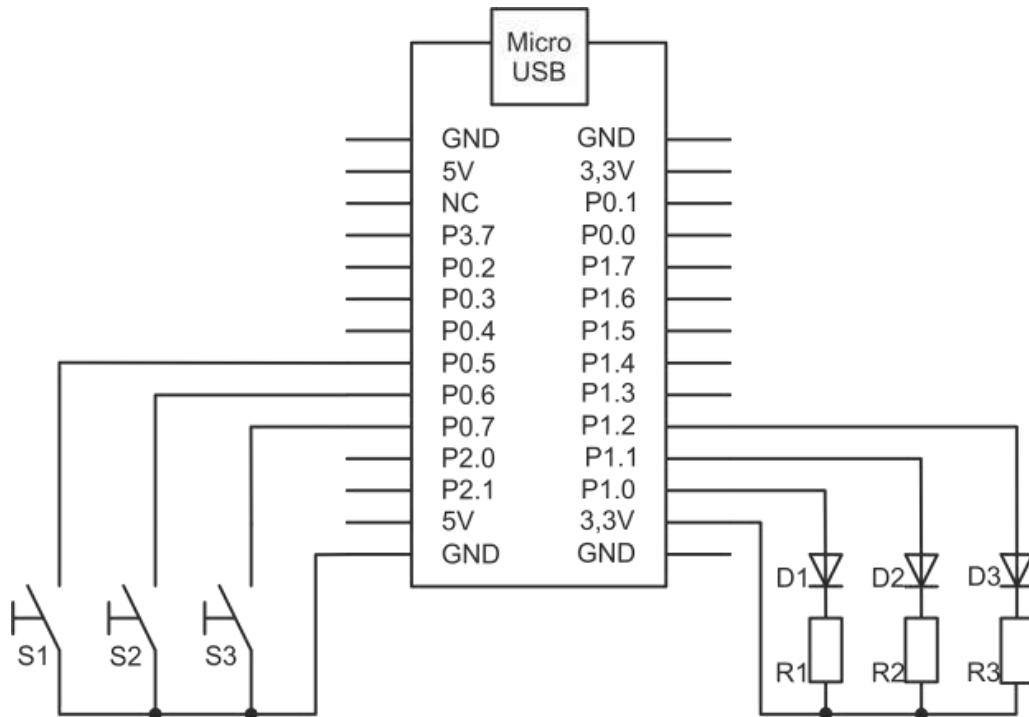
8. Über die Eigenschaften des Formulars kann man Ereignisse einstellen, auf die reagiert werden kann. Stellen Sie ein „Formular Schließen“-Ereignis ein.  
Schließen Sie alle angeschlossenen IO-Warrior dann, wenn auch das Formular geschlossen wird.
9. Überprüfen Sie ihr Programm und vergessen Sie nicht, Ihr Projekt abzuspeichern. Vielleicht möchten Sie es mit nach Hause nehmen (USB Stick)?

[illegible]

## Auftrag 2: Einlesen von Schaltzuständen

Die 19 Pins der Ports des IO-Warriors 28 können sowohl als Eingänge, als auch als Ausgänge genutzt werden. In diesem Auftrag sollen die Schaltzustände von zwei Tastern eingelesen und ausgewertet werden. Die drei Taster werden an die Ports P0.5, P0.6 und P0.7 des Warriors angeschlossen. Dies entspricht den Pins 8.9 und 10 des ICs. Also das sechste, siebte und achte Bit des Bytes von Port 1<sup>1</sup>.

Die Taster sind entsprechend nachfolgender Abbildung in die Schaltung zu integrieren:



Über die Funktion

**IOWrite(int iowHandle, int numPipe, ref byte buffer, int length)**

muss deklariert werden, welche Ports Ein- und welche Ports Ausgänge sein sollen.

Im Einzelnen:

`int iowHandle` ist das Handle (die „Erkennungsnummer“) des angeschlossenen Warriors.  
`int numPipe` entscheidet, ob die Ports als I/O Ports oder als Sonderfunktionsports genutzt werden. „0“ heißt, dass sie I/O-Ports sein sollen.  
`ref byte buffer` ist eine Referenz auf das erste Byte, das geschrieben werden soll.  
`int length` ist die Anzahl der Bytes, die auf das erste Byte folgen.

Der Wert von `int numPipe` muss also „0“ (Null) sein, damit wir die Ports als Ein-, bzw. Ausgänge nutzen können.

Beim IO-Warrior 28 müssen für I/O-Anwendungen fünf Bytes gesendet werden:

1. Byte: Ein sogenanntes „Report-Byte“, das für den Warrior 28 keine Rolle spielt, aber von Windows verlangt wird.
2. Byte: es beinhaltet den Zustand von Port 1, also Bit P0.0 bis Bit P0.7.
3. Byte: es beinhaltet den Zustand von Port 2, also Bit P1.0 bis Bit P1.7.
4. Byte: es beinhaltet den Zustand von Port 3 und I2C Funktion.

<sup>1</sup> Siehe Seite 4 im Datenblatt des IOW28\_Datasheet



5. Byte: es beinhaltet den Zustand von Port 4 und Power Setting.

Für die fünf Byte erstellen wir ein Byte-Array:

```
byte[] Data = new byte[5];
```

Die fünf Elemente des Byte-Arrays heißen also Data[0], Data[1], Data[2], Data[3] und Data[4]. Für uns ist zum Einlesen der Zustände nur Data[1] wichtig Data 0,2,3 und 4 werden mit „0“ initialisiert:

1. Byte Data[0]	2. Byte Data[1]	3. Byte Data[2]
Report-Byte	Zustände Port 1	Zustände Port 2
	P0.7 P0.6 P0.5 P0.4 P0.3 P0.2 P0.1 P0.0	P1.7 P1.6 P1.5 P1.4 P1.3 P1.2 P1.1 P1.0

Mit `int numPipe` haben wir festgelegt, dass die Ports 0 und 1 als Ein-/Ausgänge genutzt werden können. Nun müssen wir noch deklarieren, welche Portadressen Eingänge und welche Ausgänge werden sollen. Da unsere drei Taster an P0.5, P0.6 und P0.7 angeschlossen sind, müssen diese drei Adressen Eingänge werden (wir wollen darüber ja etwas einlesen).

Es wird also an den Warrior geschrieben, dass P0.5, P0.6 und P0.7 als Eingang genutzt werden soll. Das geschieht durch den Wert von Data[1] und Data[2] wird auf NULL gestetzt:

1. Byte Data[0]	2. Byte Data[1]	3. Byte Data[2]
Report-Byte	Zustände Port 1	Zustände Port 2
	P0.7 P0.6 P0.5 P0.4 P0.3 P0.2 P0.1 P0.0	P1.7 P1.6 P1.5 P1.4 P1.3 P1.2 P1.1 P1.0
0 0 0 0 0 0 0 0	1 1 1 0 0 0 0 0	0 0 0 0 0 0 0 0

Ist also eine Portadresse mit „0“ beschrieben, soll sie als Ausgang dienen, ist sie mit „1“ beschrieben, dient sie als Eingang.

Eselsbrücke: 1 -> I -> Input, 0 -> O -> Output

Zur Erinnerung: Data[0] ist notwendig, aber für uns nicht von Belang.

In unserem Fall haben wir alle Bits von Port 2 als Ausgänge deklariert. Bei Port 1 sind nur P0.5, P0.6 und P0.7 als Eingang deklariert worden.

Einem C#-Programm kann man keine Dualzahlen übergeben, man muss sie also umrechnen. Es ergibt sich für:

Data[0]: 00000000 = 0 (= 0x00, hexadezimale Schreibweise in C#)

Data[1]: 11100000 = 224 (= 0xE0, hexadezimale Schreibweise in C#)

Data[2]: 00000000 = 0 (= 0x00, hexadezimale Schreibweise in C#)

Diese Werte werden nun an den Warrior gesendet:

```
IOWKitWrite(IowHandle, 0, ref Data[0], 5)
```

1. Erstellen Sie ein neues Windows Forms Projekt (Tasterabfrage). Erstellen Sie die abgebildete Form mit zwei Label und drei Panels.



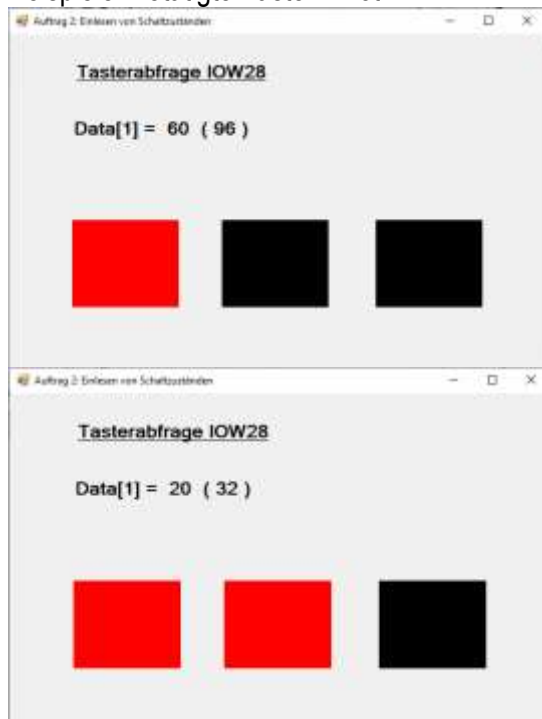
Bereiten Sie alles notwendige vor, um auf die Funktionen der iowkit.dll zugreifen zu können.

- Erzeugen Sie ein neues Byte-Array mit fünf Byte-Elementen und füllen Sie die fünf Elemente mit den entsprechenden Daten. Wenn das Formular geladen wird, laden Sie das Handle des angeschlossenen Warriors in die Variable `lowHandle`, öffnen sie den Warrior und senden Sie alle notwendigen Daten über die `lowKitWrite`-Funktion an den Warrior.
- Nun sollen alle 50ms die Eingänge abgefragt werden. Dazu benötigen wir einen Timer. Wird das Formular geladen, soll der Timer aktiviert werden (z.B. `timer1.Enabled=true;`), wird das Formular geschlossen, so muss neben dem Schließen des Warriors auch der Timer wieder deaktiviert werden (z.B. `timer1.Enabled=false;`).
- Bei jedem Timer-Tick-Event müssen nun die Eingänge abgefragt werden, welchen Zustand sie haben. Dies soll, wie gesagt, alle 50ms geschehen und mit der `IowKitReadNonBlocking(int iowHandle, int numPipe, ref byte buffer, int length)` realisiert werden. Lassen Sie sich den Zustand der Eingänge über das Label anzeigen (Dezimal und Hexadezimal):

Data[1] = E0 ( 224 )

5. Lassen Sie das linke Panel rot werden, wenn der linke Taster gedrückt wird, das rechte beim rechten Taster und wenn der mittlere Taster gedrückt wird soll das Panel ebenfalls in der Mitte rot werden. Ist der jeweilige Taster nicht gedrückt, wird das Label schwarz:

Beispiele: Betätigte Taster in rot

[illegible]

## Auftrag 3: Ansteuern von Ausgängen

Möchte man die Ausgänge des IO-Warriors 28 ansteuern, so kann man deren Status zum Beispiel durch Leuchtdioden anzeigen. In unserem Fall schließen wir eine rote, eine gelbe und eine grüne LED an die entsprechenden Ports des Warriors an:

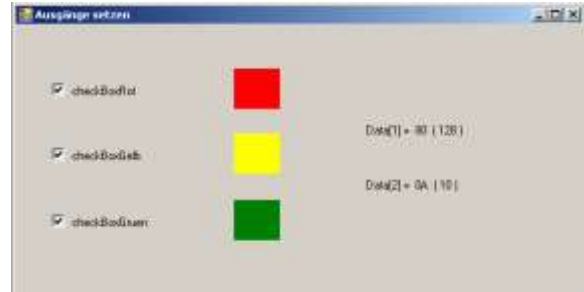


Pin 19: rote LED an P1.2, Pin 18: gelbe LED an P1.1, Pin 17: grüne LED an P1.0

1. Erstellen Sie folgendes Formular und bereiten Sie alles Notwendige für die Kommunikation des IO-Warriors über die USB Schnittstelle vor.

Nennen Sie die Elemente des Formulars so, wie in der Abbildung zu sehen ist: labelData1, labelData2, checkBoxRot, checkBoxGelb, checkBoxGruen, panelRot, panelGelb, panelGruen.

2. Folgende Funktion soll implementiert werden: Klickt man auf eine Checkbox (Haken gesetzt), so soll das entsprechende Panel in der entsprechenden Farbe leuchten. Ist die Checkbox nicht aktiviert, ist das Panel schwarz. Außerdem soll die entsprechende LED auf dem Board leuchten. Z.B.: Haken bei checkBoxRot gesetzt: panelRot ist rot, rote LED auf dem Board leuchtet. Darüber hinaus sollen die Werte von Data[1] und Data[2] in den jeweiligen Labels angezeigt werden, z.B. so:



Dies soll am Beispiel der grünen LED verdeutlicht werden:

Die grüne LED liegt an P1.0, analog zum vorigen Auftrag würde das das vierte Bit des Bytes von Data[2] sein.

Ist `checkBoxGruen.checked==true`, muss folgendes Bitmuster vorliegen:

Port 1, Data[2]							
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
0	0	0	0	0	0	0	1

Das entspricht hexadezimal dem Wert 8, in C# Schreibweise: `00x01`.

Ist `checkBoxGruen.checked==false`, muss folgendes Bitmuster vorliegen:

Port 1, Data[2]							
P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0
0	0	0	0	0	0	0	0

Das entspricht hexadezimal dem Wert 0, in C# Schreibweise: `00x00`.

Analog zum ersten Auftrag muss eine Bitmaskierung vorgenommen werden, um den gewünschten Zustand zu erreichen:

Portadresse	P1.7	P1.6	P1.5	P1.4	P1.3	P1.2	P1.1	P1.0	
Wert von Data[2]	0	0	0	0	0	0	0	0	0x00
Maskierung durch ODER ( )	0	0	0	0	0	0	0	1	0x01
<b>Ergebnis</b>	0	0	0	0	0	0	0	1	0x01

Programmiert man nun folgendes:

```
private void checkBoxGruen_CheckedChanged(object sender, EventArgs e)
{
    int WertNeu;
    if (checkBoxGruen.Checked == true)
    {
        panelGruen.BackColor = Color.Green;
        WertNeu = Data[2] | 0x01;
        Data[2] = Convert.ToByte(WertNeu);
        IowKitWrite(IowHandle, 0, ref Data[0], 3);
    }
    else
    {
        panelGruen.BackColor = Color.Black;
    }
}
```

```

    }
}

```

würde das für die grüne Checkbox funktionieren, allerdings würde die LED nach einmaligem Einschalten dauerhaft leuchten. Auch im `else` - Fall muss somit eine Bitmaskierung vorgenommen werden.

Programmieren Sie die geforderte Funktion für alle drei LEDs und kontrollieren Sie Ihr Ergebnis. Vergessen Sie nicht die Statusanzeigen durch die beiden Labels.

3. Programmieren Sie ein Lauflicht mit den drei LEDs. Dabei sollen nur die LEDs am Lauflicht beteiligt sein, deren Checkboxes auch angehakt sind.

## Bewertung und Kontrolle: Anwendungen mit dem IO-Warrior 28

<b>Name</b>	
<b>Klasse</b>	



Teilauftrag	Punkte / Bewertung / Bemerkungen / Unterschrift
<b>Auftrag 1:</b> Erkennen von angeschlossenen IO-Warriors Abgabetermin: _____	
<b>Auftrag 2:</b> Einlesen von Schaltzuständen Abgabetermin: _____	
<b>Auftrag 3:</b> Ansteuern von Ausgängen Abgabetermin: _____	
<b>ANWENDUNGEN MIT DEM IO-WARRIOR 28</b>	