Karlsruher Institut für Technologie

**Institut für Mikro- und Nanoelektronische Systeme**
Leiter: Prof. Dr. rer. nat. habil. Michael Siegel

# Praktikum ASE / PSoC

# Programmierbare Mixed-Signal “Systeme on Chip“

September 2013

# Inhalt

# 1 Einleitung

Um den Bedienkomfort, die Sicherheit und die Funktionalität von elektrischen Geräten (Haushalt, Multimedia, Messtechnik, usw.) immer weiter zu verbessern, wird ein immer größerer Einsatz von elektronischer Hardware notwendig. Zu solcher Hardware zählen z.B. Mikrocontroller, AD/DA-Wandler, Counter, Timer, Operationsverstärker, Filter, Schnittstellen und vieles mehr. Diese wurden in der Vergangenheit meist als separate integrierte Schaltungen (ICs) auf einem Silizium-Chip realisiert und mit einem Gehäuse versehen.

Mit steigender Anzahl solcher Funktionsbausteine (ICs) steigen auch die Herstellungskosten. Abhilfe schafft hier die Integration möglichst vieler Funktionen in einen einzelnen Baustein. Dabei werden zunehmend analoge und digitale Funktionen gemeinsam auf einem Chip realisiert. Man spricht dann von Mixed-Signal-Chips oder System-on-Chip (SoC). So gibt es viele Mikrocontroller, welche bereits einige der oben genannten Komponenten enthalten. Meist sind diese nicht ausreichend flexibel konfigurierbar und/oder die Anzahl der zur Verfügung stehenden Funktionen ist zu gering.

Die Firma Cypress bietet ebenfalls Mikrocontroller an, die solche Funktionen beinhalten. Neu an deren Entwicklung sind zum einen die Vielzahl der analogen und digitalen Funktionen, sowie die flexible Konfiguration. Die Bausteine können mit einem Programmiergerät mindestens 10000-mal neu programmiert werden. Der Mikrocontroller selbst kann programmgesteuert die analogen und digitalen Funktionen beliebig oft dynamisch rekonfiguriert. Cypress nennt seine SoCs deshalb PSoCs (Programmable System on Chip).

Durch diese dynamische Rekonfiguration sind z.B. die zwei folgenden Konfigurationen auf ein und demselben Baustein möglich, zwischen denen per Schalter oder softwaregesteuert umgeschaltet werden kann. Somit können die Chipresourcen mehrfach genutzt werden.

| Konfiguration 1 | Konfiguration 2 |
|---|---|
| Ein 8-Bit Counter | Ein 16-Bit Counter |
| Ein 16-Bit Timer | Ein 8-Bit PWM |
| Ein Full-Duplex UART mit Baudratengenerator | Ein Half-Duplex UART |
| Ein SPI Slave (Full Duplex) | Ein SPI Master |
| Ein 4-Kanal 8 Bit Delta-Sigma AD-Wandler | Ein 12-Bit AD-Wandler (inkremental) |
| Ein 6-Bit DA-Wandler | Ein Tiefpass-Filter |
| Ein 8-Bit DA-Wandler | Ein 8-Bit DA-Wandler |
| Zwei Tiefpass-Filter | Zwei Instrumentationsverstärker |

# 2 PSoCs

PSoC steht für Programmable System on Chip und wird von Cypress auch als Mixed-Signal-Array bezeichnet.

Ein PSoC-Baustein besteht aus verschiedenen Subsystemen auf einem Chip. Die Subsysteme selbst und deren Verbindungen untereinander können flexibel konfiguriert werden, so dass ein SoC (System on Chip) entsteht.

Die Erstellung einer PSoC-Applikation erfolgt mit der integrierten Entwicklungsumgebung (IDE). Cypress nennt diese IDE PSoC-Creator.

Die PSoC-Familie beinhaltet eine Vielzahl unterschiedlicher PSoC Bausteine. Die wesentlichen Unterscheidungskriterien sind:

- Programmspeichergröße (Flash-PROM)
- Anzahl der Analog-Digital Wandler
- Gehäuseform und Anzahl der Ein-/Ausgabe-Pins

Für das Praktikum kommt der CY8C5568-AXI060, ein Baustein der PSoC-Familie, zum Einsatz. Eine Übersicht aller Typen finden Sie im Anhang PSoC-Auswahlbaum.

## 2.1 PSoC CY8C5568-AXI060

Das nachfolgende Blockschaltbild zeigt die einzelnen Subsysteme des PSoC, die über den Systembus miteinander verbunden sind. Dieses Blockschaltbild deckt die Funktionalität aller PSoC Bausteintypen ab.

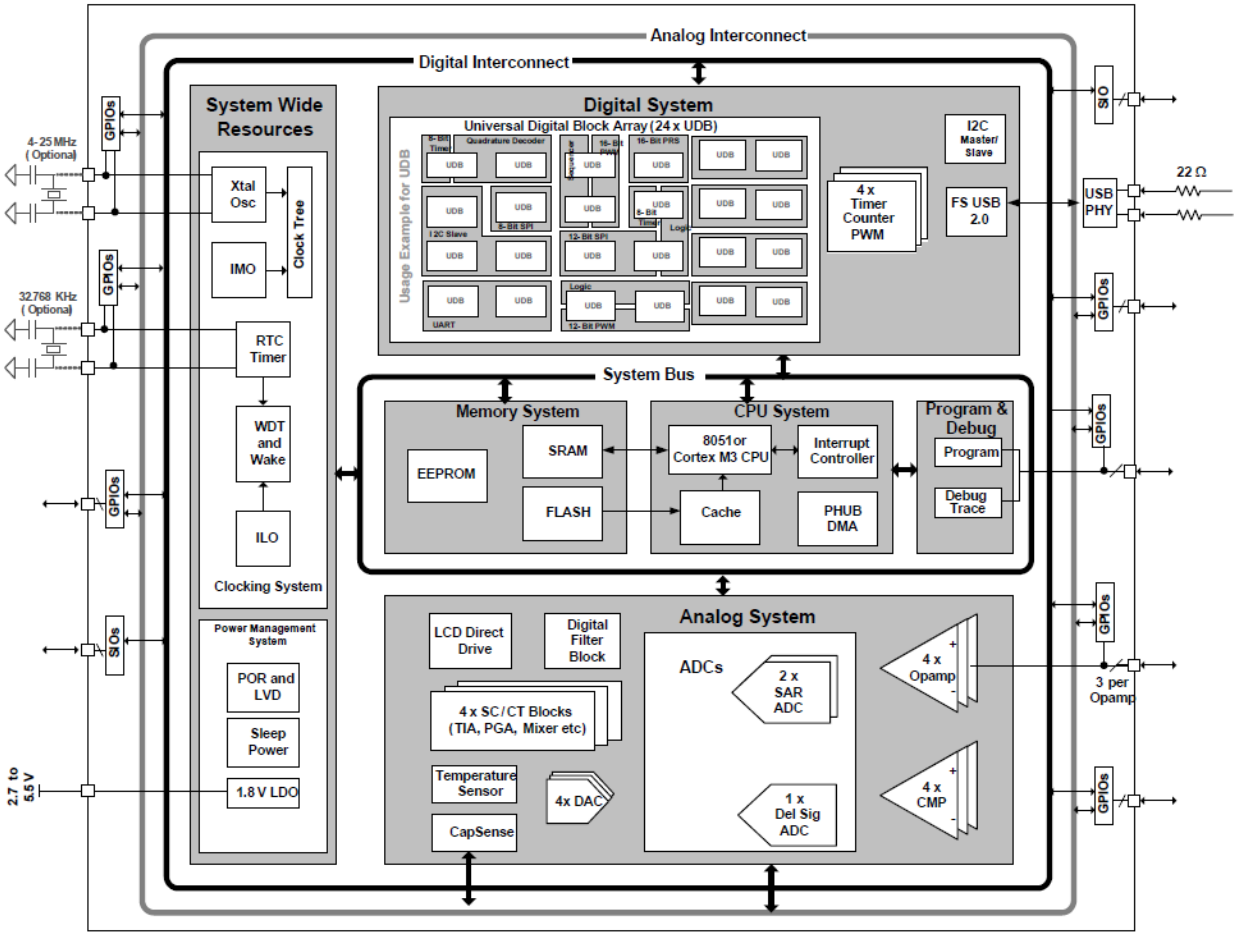


Abbildung 2-1: Vereinfachtes Blockschaltbild

## ***Subsystem PSoC Core***

Der PSoC ist ein leistungsstarkes CPU Subsystem mit einem ARM Cortex-M3 Prozessor. Das *SRAM* dient dabei als Datenspeicher und der *Flash*-Speicher als Programmspeicher.

Mit dem *Interruptcontroller* können durch externe oder interne Ereignisse (z.B. Signalflanke an IO-Pins, Ablauf von Timern, usw.), unabhängig vom eigentlichen Hauptprogramm, sogenannte ISR (Interrupt-Service-Routinen), ausgeführt werden.

*GPIOs* sind die Schnittstellen zwischen den PSoC-Pins (8 Pins pro Port) und der CPU. Sie beinhalten für jeden Pin einen konfigurierbaren *GPIO* –Block (General Purpose IO), mit dem sie zur digitalen und/oder analogen Signalein-/ausgabe eingestellt werden. Über interne Register für *Digital Interconnect* und *Analog Interconnect* werden die analogen und digitalen Blöcke mit den GPIOs verbunden.

## ***Subsystem Analoge Blöcke***

Die analoge Peripherie bietet eine Vielzahl an Funktionen. Eine wichtige Funktion sind die A/D-Wandler. Der PsoC5 bietet einen Delta-Sigma- und zwei Successive-Approximation-ADW. Ersterer bietet bis zu 20bit Genauigkeit bei maximal 192ksps, die SAR-ADW sind mit 700ksps deutlich schneller, wandeln dafür aber auch nur mit 12bit. Über vier sogenannte Switched Capacitor Blöcke können diverse Verstärker und Mixer

implementiert werden. Weitere Komponenten sind je vier Op-Amps, Komparatoren, D/A-Wandler, ein Block für digitale Filter, ein LCD-Treiber und weitere spezialisierte Blöcke.

***Subsystem Digitale Blöcke***

Kernstück der digitalen Peripherie ist das Universal Digital Block Array mit 24 UDBs, auf denen eine Vielzahl digitaler Funktionen abgebildet werden kann. Für Timer, Zähler und PWMs gibt es zusätzlich 4 sogenannte Fixed-Function-Blocks, sodass für diese drei Komponenten zwischen UDB und FF gewählt werden kann. Die FF-Blocks sind allerdings in ihrer Funktionalität eingeschränkt. Ebenfalls losgelöst von UDB und FF ist die USB-Engine, mit der die USB2.0-Schnittstelle realisiert wird.

***Subsystem Resources***

Zusätzlich zu den analogen und digitalen Blöcken sind auf dem Chip noch folgende Systemresourcen integriert:

- Programmierbare Taktquellen (Digital Clocks)
- Dezimator (Decimator)
- $I^2C$ Controller (Master oder Slave)
- Überwachung der Spannungsversorgung (POR/LVD)
- Interne Referenzspannung (Internal Voltage Reference)

Weitere Informationen hierzu finden Sie im Datenblatt oder im Referenzhandbuch

### 2.1.1 Analog-Teil

Ausführliche Informationen finden Sie im Referenzhandbuch!

### 2.1.2 Digital-Teil

Ausführliche Informationen finden Sie im Referenzhandbuch!

### 2.1.3 GPIO

Die GPIO-Pins (General Purpose InOut) können sehr flexibel konfiguriert werden. Für die digitale sowie analoge Ein-/Ausgabe von Signalen stehen 8 sogenannte ***Drive Modes*** zur Verfügung. Nicht jeder Pin unterstützt alle Drive Modes. Die Drive Modes legen fest, ob ein GPIO z.B. als analoger Ausgang dienen soll oder als digitaler Eingang. Über die IDE sowie programmgesteuert kann dies eingestellt werden.

| Drive Mode | High | Low |
|---|---|---|
| High impedance analog | High-Z | High-Z |
| High impedance digital | High-Z | High-Z |
| Resistive pull-up | Res High(5k) | Strong low |
| Resistive pull-down | Strong High | Res low (5k) |
| Open drain, drives low | High-Z | Strong low |

| | | |
|---|---|---|
| Open drain, drives high | Strong high | High-Z |
| Strong drive | Strong high | Strong low |
| Resistive pull-up and pull-down | Res High(5k) | Res low (5k) |

Tabelle 2-1:GPIO Drive Modes

Die beiden ***high impedance*** *Drive Modes* dienen zur analogen Ein-/Ausgabe. Der erste davon dient auch zur digitalen Eingabe. Mit ***strong drive*** werden digitale Signale ausgegeben, ebenso mit ***slow strong drive***, nur mit dem Unterschied, das hier die Treiberleistung reduziert ist und auch eine geringere Bandbreite unterstützt wird.

### 2.1.4 Systemressourcen

Ausführliche Informationen finden Sie im technischen Referenzhandbuch (*trm.pdf*).

### 2.1.5 Mikrocontroller Cortex M3

Der Kern des PSoC5 ist eine Cortex M3 CPU von ARM, welche eine hohe Rechenleistung bereitstellt. Die 32-bit-CPU taktet mit maximal 67MHz. Im CPU-Subsystem sind außerdem noch der DMA- und der Interrupt-Controller enthalten.

Der Cortex M-3 ist ein energieeffizienter 32-bit-Prozessor, basierend auf Harvard-Architektur, mit einer dreistufigen Pipeline. Die Rechenleistung wurde mit dem Dhrystone-Benchmark auf 1,25MIPS / MHz beziffert. Er verwendet den Thumb2-Instruktionssatz, welcher hohe Performance bei geringer Code-Dichte liefert.

Das folgende Bild zeigt die Architektur des Mikrokontrollers.

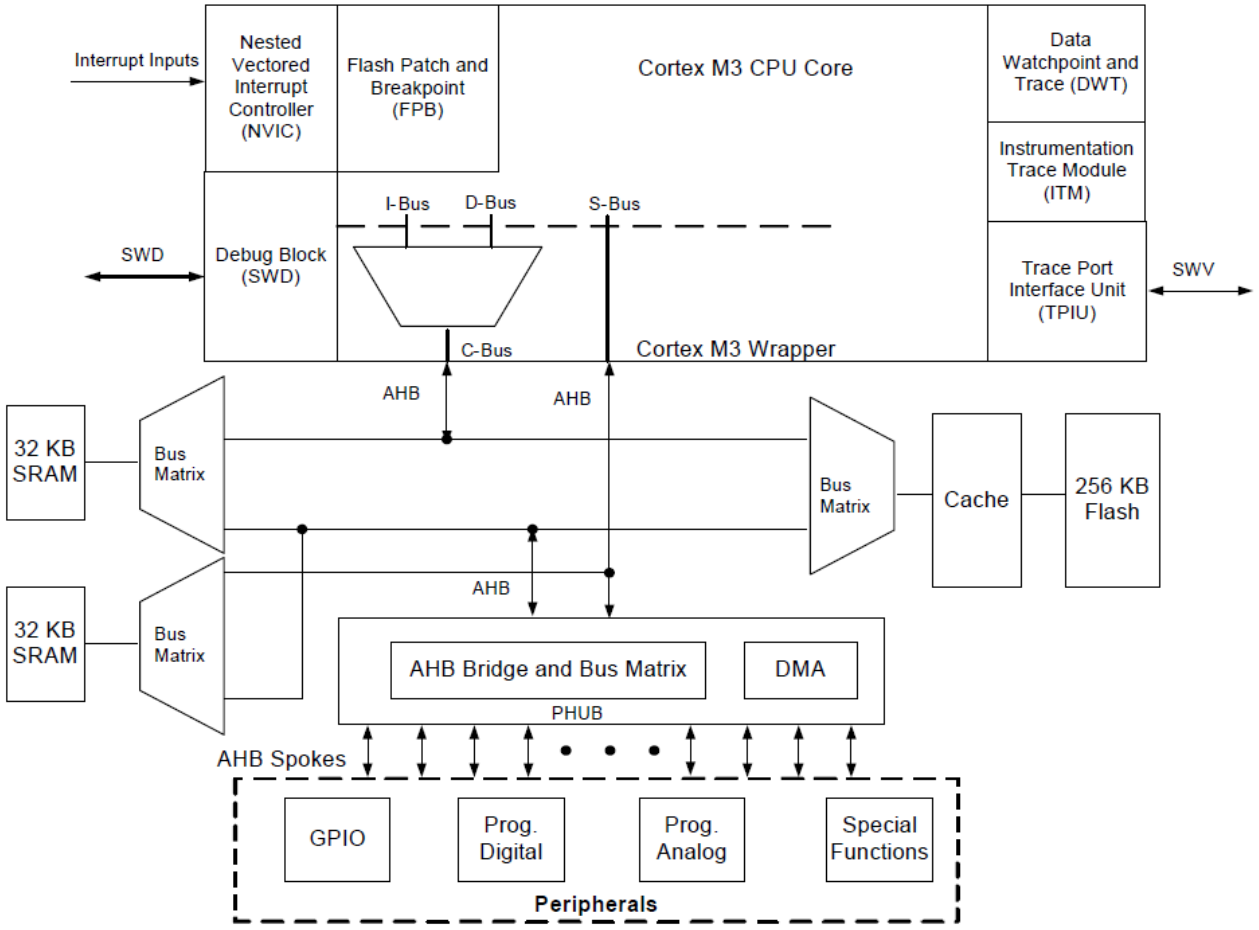


Abbildung 2-2: Cortex M3 Blockdiagramm

### 2.1.5.1 Interne Register

Die Cortex M3 CPU verwendet folgende interne 32-bit-Register:

- **R0 – R12 General Purpose Register**
  Für die allgemeine Verwendung durch Instruktionen vorgesehen. R8-R12 können nur von einigen 16-bit- und den 32-bit-Instruktionen verwendet werden.
- **R13 Stack Pointer Register(SP)**
  Beinhaltet entwedern den Main Stack Pointer oder den Process Stack Pointer. Per Adressspeicherumschaltung (Bank-Switching) kann zwischen beiden gewechselt werden. Der Stack Pointer beinhaltet die Adresse des obersten Elements vom Stack.
- **R14 Link Register(LR)**
  Speichert die Programmadresse, zu der nach Ausführung einer Subroutine zurückgesprungen wird.
- **R15 Program Counter(PC)**
  Der PC beinhaltet die Adresse der nächsten Anweisung.
- **xPSR Program Status Register**
  Unterteilt in Application Program Status Register(APSR), Interrupt Program Status Register(IPSR) und Execution Program Status Register(EPSR) über welche Systemzustände und Betriebsmodi auslesbar sind.

Auf den ersten Blick fehlen im Vergleich zu einem klassischen Mikrokontroller der Accumulator und das Index-Register. Der Cortex M3 hat kein spezielles Register wie den

Accumulator mehr, in das Ergebnisse geschrieben werden – diese Funktion kann im M3 jedes General Purpose Register erfüllen, die Wahl wird dabei der Instruktion überlassen. Das Index-Register ist im Grunde genommen ein General Purpose Register.
Zusätzlich sind auf dem M3 128Byte Cache untergebracht, aus dem Instruktionen über Direct Mapping aus dem Flash geladen werden können. Dadurch wird das Laden der Instruktionen erheblich beschleunigt.

Der DMA-Controller unterstützt bis zu 24 Kanäle, über die Daten ohne Beteiligung der CPU übertragen werden können. Auf diese Weise ist zum Beispiel eine gewisse Parallelisierung möglich, außerdem ist ein DMA-Tranfer schneller als ein CPU-Transfer: Bei einem CPU-Transfer werden die Daten erst von Datenquelle in den SRAM geschoben und anschließend vom SRAM zur Datensenke; ein DMA-Transfer findet direkt zwischen Quelle und Senke statt.

Der Interrupt-Controller unterstützt bis zu 32 Interrupts auf 8 unterschiedlichen, dynamisch anpassbaren Prioritäten. Wird ein Interrupt ausgelöst, wird der normale Programmfluss unterbrochen, die zugehörige Interrupt Service Routine (ISR) durchgeführt und der unterbrochene Programmfluss wieder aufgenommen. Interrupts können nur durch Hardware ausgelöst werden. Interruptquellen können alle UDB- und „Fixed Function“-Blöcke sowie der DMA-Controller sein. Die Interrupts können verschachtelt werden („nested interrupts“), d.h. ein Interrupt hoher Priorität kann die Ausführung einer Interrupt Service Routine niedrigerer Priorität unterbrechen. Wenn die ISR der höheren Priorität abgearbeitet ist, wird der Rest der ISR niedrigerer Priorität abgeschlossen.

### 2.1.5.2 Speicher

Der PSoC5 stellt 256kB Flashspeicher und 2kB EEPROM als Programmspeicher, 64kB static RAM(SRAM) als Datenspeicher zur Verfügung. Während der SRAM flüchtig ist und daher nur zur temporären Speicherung von Daten geeignet ist, halten EEPROM und Flashspeicher Daten Spezifikation bis zu 20 Jahre. Der eigentliche Programmcode wird in den Flashspeicher geschrieben. Um das Auslesen oder Verändern des Flashspeichers zu verhindern, kann externer Zugriff auf den Flashspeicher unterbunden werden. Dies ist zum Beispiel bei proprietärem Code zum Schutz vor Industriespionage wünschenswert. Um den Debugger verwenden zu können, muss allerdings wenigstens Lesezugriff gewährt werden.
Der EEPROM dient zur nichtflüchtigen Speicherung von Benutzerdaten und wird nur auf explizite Anweisung verwendet. Im Unterschied zum Flashspeicher kann der EEPROM $10^6$x wiederbeschrieben werden (Flash: $10^4$x).

Der durch die 32-bit-CPU adressierbare Speicherbereich von 4GB ist wie folgt aufgeteilt:

| Adressbereich | Größe | Verwendung |
|---|---|---|
| 0x00000000 – 0x1FFFFFFF | 0,5GB | Programmcode |
| 0x20000000 – 0x3FFFFFFF | 0,5GB | Static RAM(Datenspeicher) |
| 0x40000000 – 0x5FFFFFFF | 0,5GB | Peripherie |
| 0x60000000 – 0x9FFFFFFF | 1GB | Externer RAM |
| 0xA0000000 – 0xDFFFFFFF | 1GB | Externe Peripherie |
| 0xE0000000 – 0xFFFFFFFF | 0,5GB | Interne Peripherie (Interrupt Controller, Debug/Trace Module) |

Tabelle 2-2: Speicherbereiche

Natürlich verfügt der PSoC allein nicht über genug Speicher und Peripherie, um den 32-bit-Adressraum komplett zu nutzen. So ist z.B. der für den Programmcode vorgesehene Speicherbereich mit dem Flashspeicher (0x00000000 bis 0x0003FFFF) und der Hälfte des SRAMs (0x1FFF8000 bis 0x1FFFFFFF, 32kB) nicht annähernd ausgefüllt.

Neben den internen Registern gibt es auch sogenannte externe Register. Diese befinden sich ebenfalls auf dem PSoC-Chip. Sie werden als externe Register bezeichnet, da sie nicht in den CPU-Kern integriert sind, sondern wie die Speicher, in einem separaten Bereich auf dem PSoC-Chip untergebracht sind.

Über die externen Register werden sämtliche analoge und digitale Blöcke konfiguriert. Somit sind die verfügbaren Usermodule nichts anderes als eine Sammlung vordefinierter Registereinstellungen.

Im technischen Referenzhandbuch (*trm.pdf*) finden Sie die komplette Auflistung der externen Register.

# 3 PSoC Entwicklungsumgebung

## 3.1 Integrated Development Environment (IDE)

Cypress stellt eine eigene IDE für den PSoC5 bereit, die sich PSoC Creator nennt. Mit dem PSoC Creator wird die Hardwarekonfiguration des PSoCs erstellt, das Programm geschrieben und der Chip selbst mit den so erzeugten Daten programmiert. Außerdem ermöglicht der PSoC Creator das Debuggen des Chips.

Allgemein wird zum Erstellen eines neuen Projektes ein neuer Workspace erstellt, in welchem zusammenhängende Projekte zusammengefasst sind und z.B. in einem Workspace Bundle gepackt werden können. Wird ein Projekt geöffnet, wird immer sein Workspace und damit alle zugehörigen Projekte geöffnet.
Im zentralen Unterfenster wird bei Programmstart eine Homepage mit Neuigkeiten und Quicklinks angezeigt. Nach dem Laden eines Projektes werden hier die Projektdaten im entsprechenden Editor angezeigt; zwischen geöffneten Projektdaten kann mittels der Reiter direkt über dem zentralen Unterfenster umgeschaltet werden.

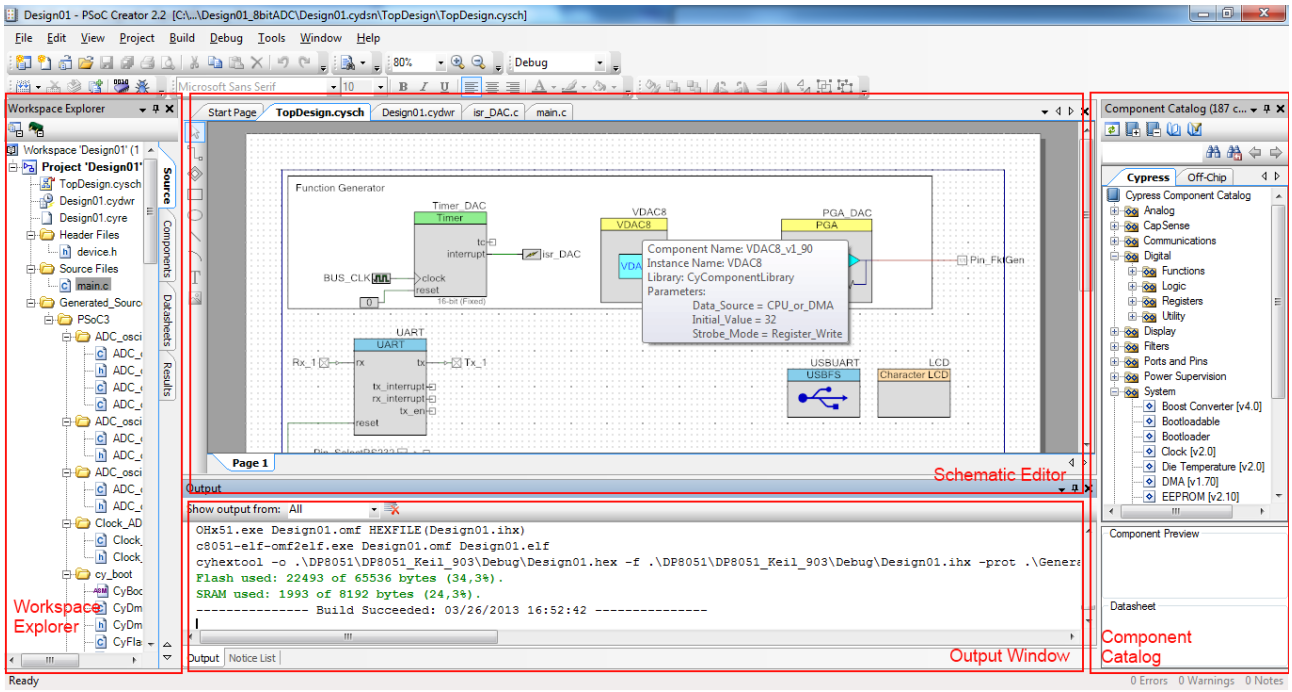


Abbildung 3-1: PSoC Creator Übersicht

a) **Workspace Explorer**

Der Workspace Explorer befindet sich gewöhnlich auf der linken Seite. In ihm werden alle Projektdaten, Datenblätter und Ergebnisse des Build-Prozesses wie Timing-Analysen etc. angezeigt. Hier sind z.B. alle Header- und Quellcodedateien, auch die der Interrupts, zu finden. Wenn er nicht angezeigt wird, kann er über View → Workspace Explorer angezeigt werden.

b) **Output Window**

An der Fensterunterseite befindet sich das Output Window, in dem Fehler, Warnungen und sonstige Informationen zu Build-, Compile- und Generate-Prozessen angezeigt werden. Es kann ebenfalls über das View-Menü aufgerufen werden.

### c) Schematic Editor

Im Schematic Editor werden die Hardwarekomponenten konfiguriert und verdrahtet. Es wird im mittleren Fensterbereich angezeigt, sobald eine .cysch-Datei aufgerufen wird. In der Standardanordnung befindet sich auf der rechten Seite der Komponentenkatalog, über den auf alle mitgelieferten und selbstdesignten Komponenten zugegriffen werden kann. Zusätzlich können off-Chip-Bauteile platziert werden. Diese dienen lediglich der Information und haben keine Auswirkung auf die Programmierung des PSoC.
Alle Komponenten lassen sich mittels simplen Drag-N-Drop platzieren.

Auf der linken Seite des Schematic Editors befinden sich Werkzeuge zum Verdrahten (Wire Tool, Hotkey: W), Texte schreiben (Annotation Tool, Hotkey: T) usw. Mit dem Verdrahtungswerkzeug können auch Busse angelegt werden.

Per Doppelklick oder Rechtsklick → Configure... wird der Konfigurationsdialog einer Komponente aufgerufen. Hier lassen sich grundlegende Einstellungen vornehmen, die beim Start oder Reset der Komponente geladen werden. Über Rechtsklick → Open Datasheet wird das Datenblatt in einem pdf-Reader geöffnet. Neben einer Beschreibung der Komponente und seiner Funktionsweise, Konfigurationsoptionen und Informationen zu Registern, Speicherbedarf und elektrischen Charakteristika sind hier auch die Instruktionen der API zu finden.

### d) Design Wide Ressources (DWR)

Über die DWR werden zentrale Komponenten des Chips konfiguriert. Dazu gehören zum Beispiel die Taktsignale, der DMA-Controller, Interrupts und vor allem das Pin Out.

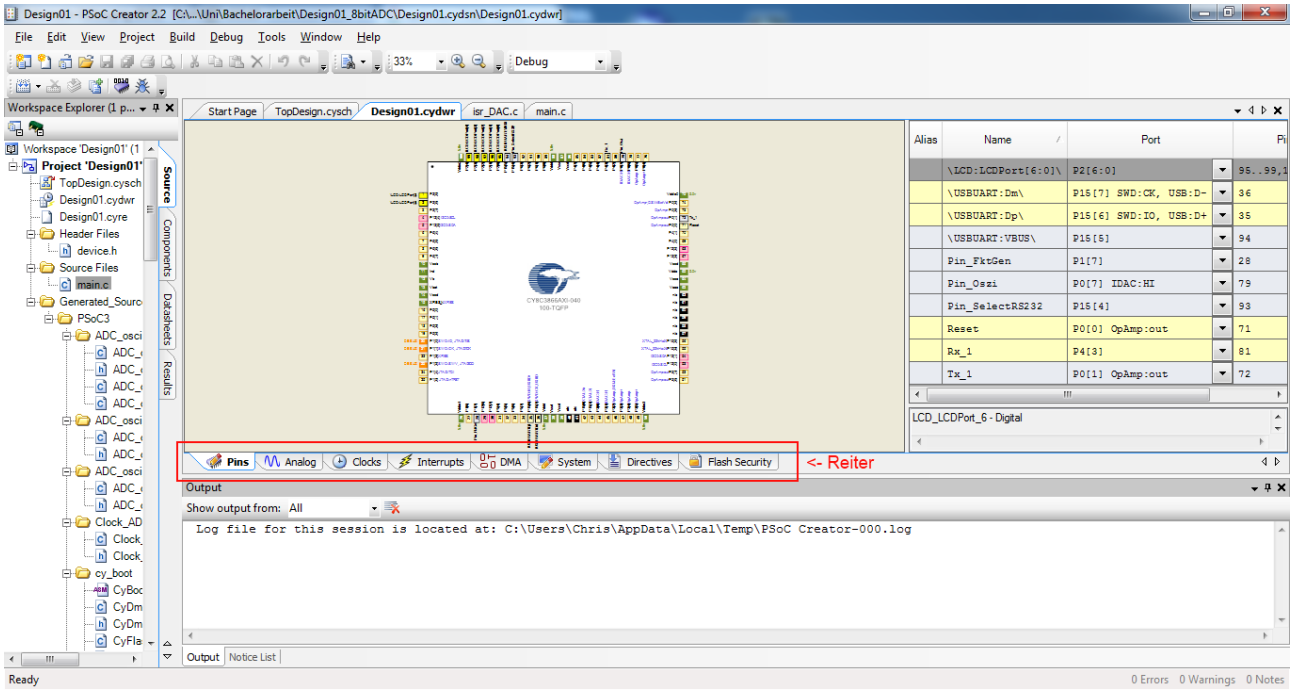


Abbildung 3-2: Design Wide Resources

Um die Pins zu konfigurieren muss zunächst ein Pin im Schematic Editor platziert werden. Dieser lässt sich dann im Pinout Editor mit einem Hardwarepin assoziieren. Dabei ist noch zu beachten, dass einige Pins spezielle Funktionen haben, z.B. Pins für den LCD-Treiber oder die USB-Schnittstelle. Sie lassen sich zwar auch für andere Signale verwenden, dann ist aber unter Umständen mit Einschränkungen zu rechnen.

Die Taktsignale werden vom PSoC Creator automatisch nach Bedarf generiert, können aber auch manuell konfiguriert werden, entweder im Reiter „Clocks“ des DWR-Editors oder bei Doppelklick auf eine Clock-Komponente im Schematic Editor. Über eine Doppelklick auf einen Takt im entsprechenden Reiter des DWR-Editors wird ein Überblick über die Taktquellen angezeigt. Diese können hier auch konfiguriert werden.

Im Reiter „Interrupts“ werden alle im Schematic Editor platzierten oder z.B. für die USB-Schnittstelle automatisch generierten Interrupts angezeigt. Hier kann die Priorität eines Interrupts geändert werden.

Weitere Reiter sind für DMA-Konfiguration, Schutz des Flashspeichers und weitere Systemeinstellungen und -informationen. Diese werden für das Praktikum allerdings nicht benötigt.

### e) Code Editor

Im Code Editor werden Header- und Quellcodedateien angezeigt. Code für PSoC3 und 5 kann in Assembler oder C geschrieben werden, allerdings stehen API-Instruktionen für die einzelnen Komponenten nur unter C zur Verfügung. Im Praktikum wird der PSoC5 daher nur in C programmiert. Die *main.c*-Datei ist hierbei die Hauptprogrammdatei

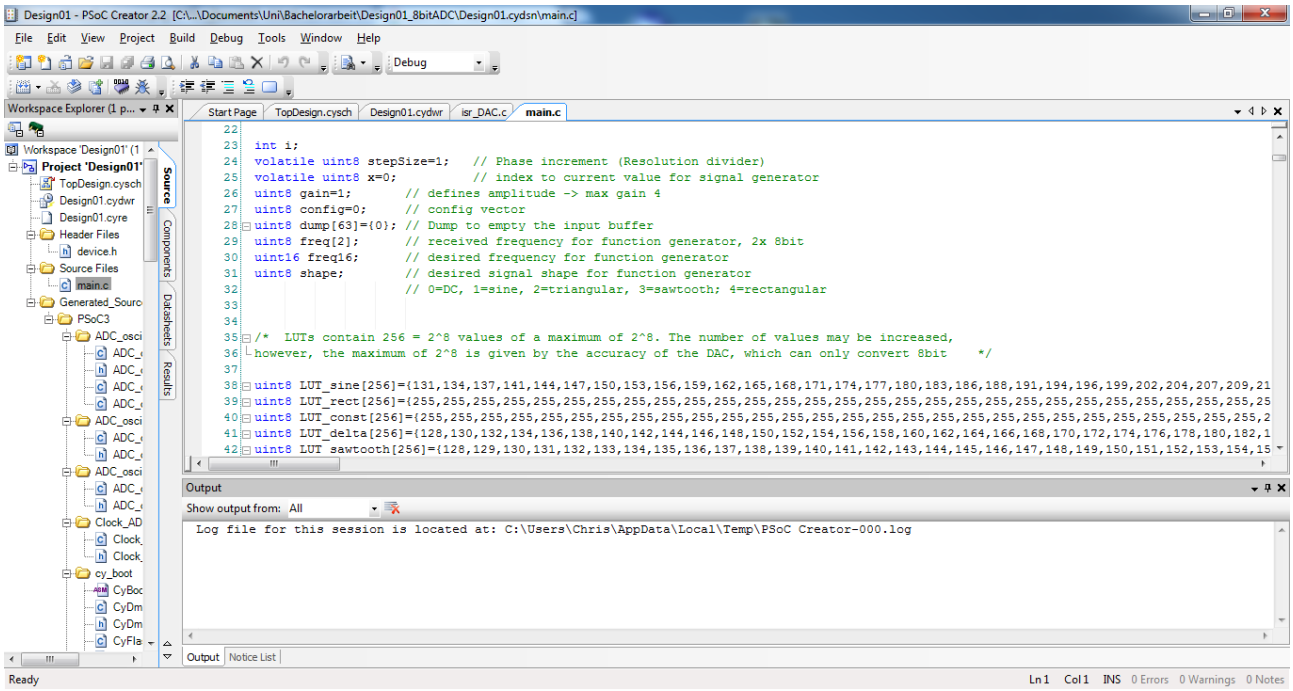


Abbildung 3-3: Code Editor

### f) Erstellung der Programmdateien

Die Erstellung der Programmdateien, welche letztlich auf den PSoC geschrieben werden, erfolgt über die folgenden Knöpfe im PSoC Creator:

Von links nach rechts sind dies:

- Build Design

- Cancel Build
- Compile File
- Generate Application
- Program
- Debug

Mit „Build Design“ wird der komplette Code erstellt, über den Drop-Down-Pfeil rechts daneben können „Clean and Build“ oder nur „Clean“ ausgewählt werden. „Clean“ löscht alle temporär erstellten Dateien, sodass diese beim nächsten Build neu erstellt werden. „Compile File“ kompiliert nur den angezeigten C-Code und kann daher auch nur geklickt werden, wenn C-Code angezeigt wird bzw. der ausgewählte Reiter ein „Code Editor“-Fenster ist.
Mit „Generate Application“ werden die Konfigurationsdaten für den PSoC erstellt und automatisch Header- und Quellcodedateien für die verwendeten Komponenten erstellt, damit diese verwenden werden können. Sobald die Schaltung verändert wurde, müssen die Konfigurationsdaten daher aktualisiert werden.. Der Button ist nur verfügbar, wenn der Schematic Editor aktiv ist und die Konfigurationsdaten nicht aktuell sind.
Mit Program werden die erzeugten Programmdateien auf den PSoC geschrieben. Dabei öffnet sich ein neues Fenster, in dem alle verfügbaren bzw. angeschlossenen PsoCs angezeigt werden und der Ziel-PSoC (das „Target“) ausgewählt werden kann.
Über den letzten Button „Debug“ wird in den Debug Modus gewechselt, in dem das Ändern der Projektdateien nicht möglich ist.

### *Hinweise*

**Verwendung von Bussen**
Busse können nicht nur mit dem Wire Tool erstellt werden, sondern ebenfalls über die Namensgebung von Signalen. Hierzu wird über Rechtsklick auf ein Wire → „Edit Name and Width“ ein Einstellungsfenster „Signal Name“ aufgerufen und „Use computed name and width“ deaktiviert. Dann sollte bei „Specify Name“ ein aussagekräftiger Name gewählt werden. Im Abschnitt „Indices“ wird die Option „Bus“ ausgewählt und seine Breite spezifiziert.

Um auf ein einzelnes Signal zuzugreifen, wird im selben Dialog „Bit“ ausgewählt und der entsprechende Index angegeben. Wichtig dabei ist, dass der korrekte Name des Busses angegeben wird.

Mit dem **Debugger** ist es möglich, den Programmcode schrittweise auszuführen und zu testen. Dazu wird eine spezielle Hardware, ein sogenannter In-Circuit-Emulator (ICE), eingesetzt. Da dieser im Praktikum nicht zum Einsatz kommt, wird auf das Debbuging nicht weiter eingegangen.

## 3.2 Evaluationsboard

Das Evaluationsboard stellt eine Plattform zum Testen der PSoC-Reihen 1, 3 und 5 bereit. Es verfügt dazu über etliche Peripheriekomponenten, wie USB- und serielle Schnittstelle, ein LC-Display, Breadboard, Knöpfe, LEDs, Varistor und sogar eine Schnittstelle für ein Funkkommunikationsmodul (welches aber nicht im Lieferumfang enthalten ist).

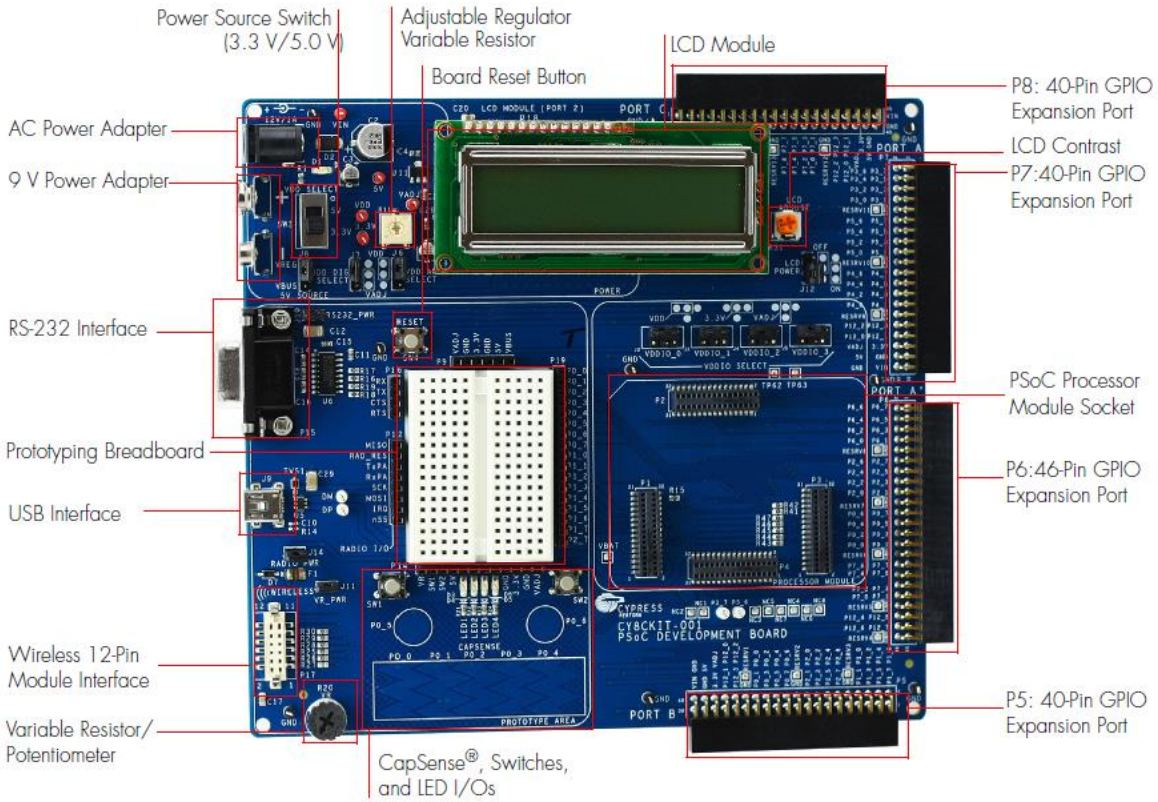


Abbildung 3-4: PSoC Developmentboard CY8CKIT-001

Das Board ist in drei Bereiche unterteilt: Power, Prototype Area und Processor Module. Im Powerbereich befindet sich der Anschluss für die 9V-Spannungsversorgung und einige Schalter, Jumper und Regler. Für das Praktikum wird die Spannungsversorgung auf 5V eingestellt und die drei Jumper befinden sich in der oberen Position. Die Spannung $V_{adj}$ und damit auch der Varistor werden für das Praktikum nicht benötigt.
In der Prototype Area findet sich ein Breadboard zum schnellen Testen eigener Schaltungen. Die Verdrahtung des Breadboards ist in der nebenstehenden Grafik zu sehen: Jeweils fünf horizontal nebeneinanderliegende Pins sind miteinander verbunden. Direkt über dem Breadboard findet sich ein Resetschalter, direkt unterhalb 4 LEDs, 2 Taster sowie ein Varistor
Außerdem befinden sich in diesem Bereich der USB-Port und die serielle Schnittstelle zu Verbindung mit dem PC.
Rechts der Prototype Area wird das Prozessormodul aufgesteckt. **Der Wechsel des Prozessormoduls wird vom Betreuer vorgenommen.**

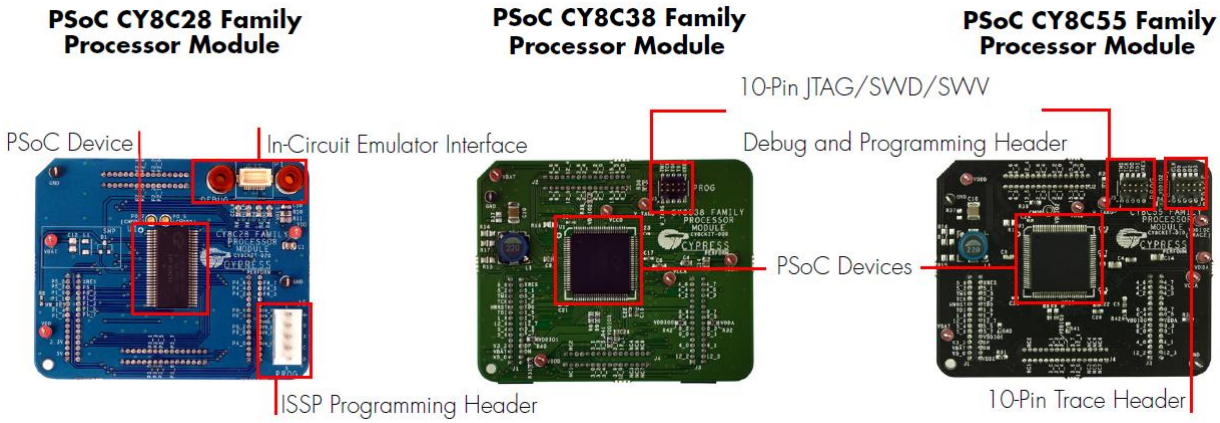


Abbildung 3-5: PSoC Prozessor Module

Über die oberhalb des Steckplatzes befindlichen Jumper werden die I/O-Spannungen $V_{DDIOx}$ eingestellt. Für das Praktikum sind alle I/O-Spannungen auf $V_{DD}$ zu stellen.

## 3.3 Programmer

Im Lieferumfang des Evalboards ist auch ein Programmer / Debugger enthalten: Der MiniProg3 wird zum schreiben der Programmdaten auf den PSoC verwendet.

Zur Programmierung des PSoC1 wird der MiniProg3 direkt auf den ISSP-Header (weißer 5-Pin Header) des Prozessormoduls gesteckt. Für PSoC3 und 5 wird eine 10-Pin JTAG Schnittstelle verwendet. Dazu wird das im Bild erkennbare, 10-adrige Flachbandkabel verwendet. Beim PSoC5 existiert ein zusätzlicher 10-Pin Trace Header, der im Praktikum aber keine Verwendung findet.
Der MiniProg3 wird per USB-Kabel mit dem Computer verbunden und dort automatisch erkannt.

## 3.4 PSoC5-Programmierung

Der PSoC5 kann in Assembler und C programmiert werden. Für das Praktikum wird der PSoC-Code in C geschrieben, da die APIs für die Komponenten nur in C verfügbar sind und diese das Schreiben von Quellcode erheblich beschleunigen. Projekte im PSoC Creator verwenden standardmäßig C, das hinzufügen von Assembler Code ist aber möglich.

C ist stark mit C++ verwand, daher sollte das C++-Kompendium oder ähnliche Dokumentation aus der Vorlesung Informationstechnik bereitgehalten werden. Im Folgenden werden Unterschiede zwischen C und C++ aufgezeigt, sowie einige Hinweise zur Programmierung auf PSoC5 aufgeführt.

### 3.4.1 Unterschiede zwischen C++ und C

C++ ist eine Erweiterung von C, d.h. jeder C++-Compiler kann normalen C-Code verarbeiten – umgekehrt funktioniert das nicht.

- Als Hauptunterschied ist die objektorientierte Programmierung zu nennen, welche mit C nicht möglich ist. Damit fallen Funktionen wie Vererbung, Datenkapselung, Polymorphie usw. weg. Für einfache Programme sind diese Funktionen meist nicht relevant, aber wenn mehrere Programmierer an einem größeren Projekt arbeiten, erleichtert OOP das Coden erheblich. Zudem wird die Wiederverwendbarkeit von Quellcode verbessert und das ganze Programm kann übersichtlicher strukturiert werden.
- Mit C++ wurde eine neue Bibliothek für Datenstreams, genannt `iostream.h`, eingeführt. Damit ist z.B. das Lesen und Schreiben in eine Datei oder in eine Konsole möglich. In C übernimmt diese Funktion die Bibliothek `stdio` (kann in C++ als `cstdio` verwendet werden). Der wesentlich Unterschied ist, dass iostreams zwar mehr Code erfordern, dafür aber für komplexere Anforderungen besser geeignet sind und allgemein als sicherer gelten.
- Der Datentyp `bool` existiert in C nicht. An seiner Stelle wird in C meist in

vorzeichenloser Integer mit möglichst geringer Größe verwendet (meist: `uint8`) und dieser auf „0“ für „false“ und „1“ für „true“ gesetzt. Viele API-Funktionen verwenden ein „non-zero“ Schema bei der Rückgabe, sodass mit „x==0“ oder „x!=0“ „false“ bzw. „true“ abgefragt werden können.

- C kennt keine Strings. Zeichenketten können in C nur als Character-Array (`char[]`) verwendet werden.

### 3.4.2 Hinweise

Im Allgemeinen ist das Programmieren in C im PSoC Creator aufgrund der umfangreichen API und zugehöriger Doku sehr einfach. Einige hilfreiche Hinweise sind im folgenden zu finden.

- Wenn das Hinzufügen von Code in automatisch generierten Dateien (außer main.c) erforderlich ist, muss darauf geachtet werden, diesen immer in den dafür bestimmten Bereich zu schreiben. Dieser ist mit Kommentaren wie z.B. „Place your Interrupt code here.“ gekennzeichnet. Benutzer-Code, der außerhalb dieser Bereiche steht, wird beim erneuten Erstellen der Konfigurationsdaten („Generate Application“) gelöscht!
- Um Interrupts zu verwenden müssen diese global aktiviert sein. Dies wird mittels `CyGlobalIntEnable;` erreicht. Dieser Befehlt ist in der automatisch generierten main.c bereits enthalten, aber auskommentiert.
- Variablendefinitionen müssen außerhalb von Dauerschleifen wie z.B. `for(;;)` oder `while(1)` stattfinden.
- Um eine Variable in zwei unterschiedlichen Quellcodedateien zu verwenden, muss diese in einer der beiden mit dem Modifier „volatile“ definiert und deklariert werden. In der anderen genügt eine Deklaration mit dem Modifier „extern“.
  Beispiel:

  Datei 1: volatile int i=0; //Zählvariable, Flag, o.ä.

  Datei 2: extern int i; //Quelldatei angeben

  „Volatile“ teilt dem Compiler mit, dass die so modifizierte Variable durch Ereignisse außerhalb der Kontrolle des Programms verändert werden kann. Damit diese Änderungen nicht verloren gehen, wird auf eine Optimierung dieser Variable und seiner Speicherzuweisung verzichtet.
  „Extern“ bedeutet schlicht, dass die Variable in einer anderen Datei bereits definiert wurde.

# 4 Weiterführende Ressourcen

## 4.1 Dokumentation von Cypress

- Technical Reference Manual PSoC5: http://www.cypress.com/?docID=39997
  Übersicht über die Architektur von PSoC5
- CY8C55 Family Data Sheet: http://www.cypress.com/?rID=37581
  Features und Ausstattung der CY8C55-Chips
- Component Data Sheets for PSoC5:
  http://www.cypress.com/?id=2233&rtID=377
- Alle Datenblätter der PsoC5-Komponenten. Auch ältere Versionen verfügbar.
- PSoC Knowledge Base: http://www.cypress.com/?id=4&catID=1353&source=header

## 4.2 Weitere Dokumentation

- C library reference: http://cplusplus.com/reference/clibrary/
  Übersicht über alle C-Standardbibliotheken.

# 5 Anhang

Im Anhang finden Sie wichtige Detailinformationen, auf welche Sie beim Erstellen von PSoC-Applikationen zurückgreifen können.

# 5.1 GPIO Blockschaltbild

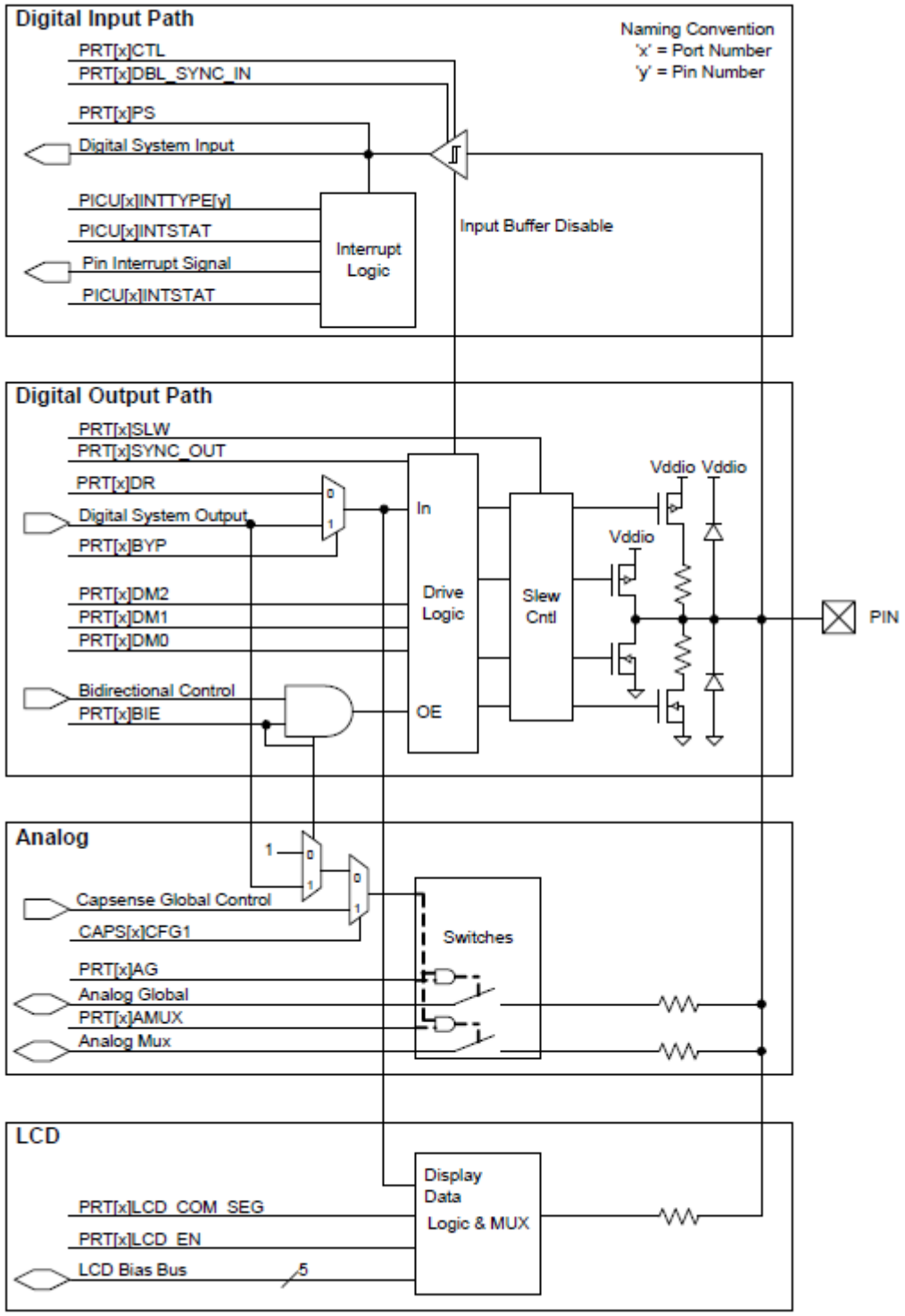


Abbildung 5-1: GPIO Blockdiagramm

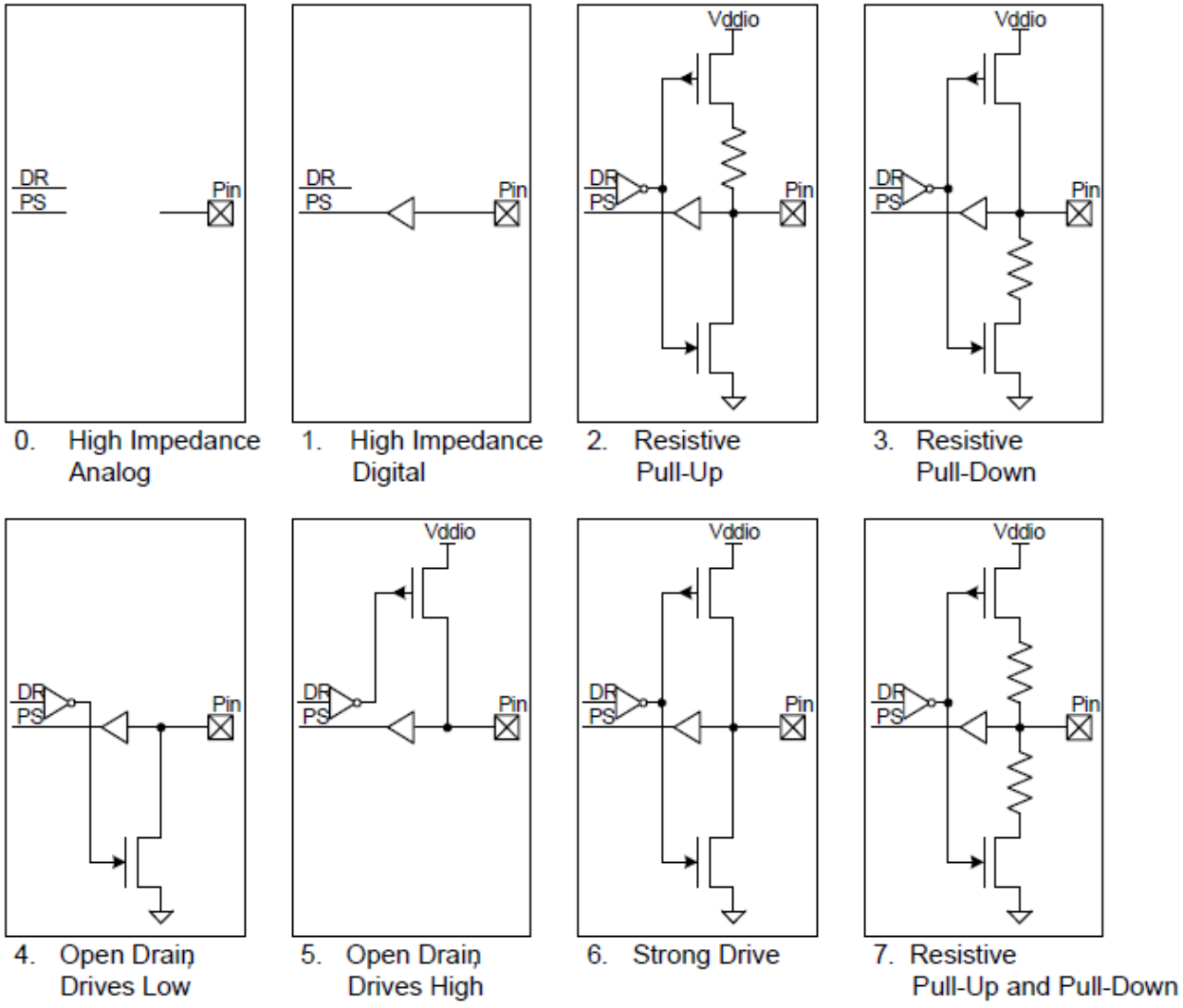


Abbildung 5-2: Drive Modes

## 5.2 Acronyme

Assembly Language User Guide (`asm.pdf` Seite 11)

Integrated Development Environment User Guide (`ide.pdf` Seite 13)

## 5.3 PSoC-Auswahlbaum

# 5.4 Usermodule

| Module | Analog | Digital | Flash | SRAM | Features |
|---|---|---|---|---|---|
| **ADCs** | | | | | |
| ADC8 | 2 | 1 | 209 | 2 | 8 bits, 8.8 ksps |
| ADCINC12 | 1 | 1 | 209 | 6 | 12 bits, 7.8 - 480 sps |
| ADCINC14 | 1 | 4 | 262 | 6 | 14 bits, 2 - 120 sps |
| ADCINCVR | 1 | 3 | 309 | 5 | 7 - 13 bits, 4 - 10 ksps |
| ADCINC | 1 | 1 | - | 8 | 6 - 15 bits, up to 46.8 ksps |
| DELSIG8 | 1 | 1 | 143 | 8 | 8 bits, 0.125 - 31.25 ksps |
| DELSIG11 | 1 | 1 | 208 | 12 | 11 bits, 125 - 7.8 ksps |
| DELSIG | 1 | 1 | 89 | 2 | 6 - 14 bits, up to 65.5 ksps |
| DUALADC | 2 | 4 | 455 | 8 | 7-13 bits, 4 - 10 ksps |
| DUALADC8 | 2 | 4 | 312 | 9 | 8 bits, 122 - 7600 sps |
| SAR6 | 1 | 0 | 58 | 0 | 6 bits, 40 ksps |
| TRIADC | 3 | 5 | 604 | 11 | 7 - 13 bits, 4 - 10 ksps |
| TRIADC8 | 3 | 5 | 416 | 11 | 8 bits, 4 - 10 ksps |
| **Amplifiers** | | | | | |
| AMPINV | 1CT | 0 | 52 | 0 | 18 gain options, max gain of -47 |
| CMP | 1CT | 0 | 17 | 0 | 2 input comparator |
| CMPPRG | 1CT | 0 | 52 | 0 | Programmable threshold and ref |
| INSAMP | | | | | |
| Two Opamp | 1CT | 0 | 57 | 0 | Programmable gain from 2 to 16 |
| Three Opamp | 1CT | 0 | 113 | 0 | Programmable gain up to 93 |
| PGA | 1CT | 0 | 52 | 0 | 30 gain options, max gain of 48 |
| **Analog Comm** | | | | | |
| DTMFDialer | | | | | |
| Background | 1 | 1 | 766 | 16 | System control while dialing |
| Foreground | 1 | 1 | 761 | 2 | Minimize use of RAM |
| **Counters** | | | | | |
| Counter8 | 0 | 1 | 67 | 0 | 8 bits, programmable pulse width |
| Counter16 | 0 | 2 | 88 | 0 | 16 bits, programmable pulse width |
| Counter24 | 0 | 3 | 129 | 0 | 24 bits, programmable pulse width |
| Counter32 | 0 | 4 | 147 | 0 | 32 bits, programmable pulse width |
| **DACs** | | | | | |
| DAC6 | 1 | 0 | 61 | 0 | 6 bits, update rates up to 250 ksps |
| DAC8 | 2 | 0 | 147 | 0 | 8 bits, update rates up to 125 ksps |
| DAC9 | 2 | 0 | 151 | 0 | 9 bits, update rates up to 125 ksps |
| MDAC6 | 1 | 0 | 112 | 0 | 6 bits, update rates up to 250 ksps |
| MDAC8 | 2 | 0 | 205 | 0 | 8 bits, update rates up to 125 ksps |
| **Digital Comm** | | | | | |
| CRC16 | 0 | 2 | 54 | 0 | 2 - 16 bits, data input clocking up to 48 MHz |
| EzI2Cs | 0 | 0 | 264 | 6 | Slave, 100/400 kbits/s |
| I2CHW | | | | | 50, 100, 400 kbits/s; 7- and 10-bit addressing |
| RAM Read/Write Buffers | 0 | 0 | 272 | 6 | |
| Additional for Flash Read Buffer Support | 0 | 0 | 89 | 2 | |
| I2Cm | 0 | 0 | 597 | 4 | 100 kbits/s |
| IrDARX | 0 | 2CB | 66 | 0 | Maximum receive rate of 115.2 kbits/sec |
| IrDATX | 0 | 2CB | 53 | 0 | Maximum transmit rate of 115.2 kbits/sec |
| RX8 | 0 | 1CB | 34 | 0 | Burst rates up to 6 Mbits/sec |
| SPIM | 0 | 1CB | 37 | 0 | 0, 1, 2, and 3 SPI clocking modes supported |
| SPIS | 0 | 1CB | 43 | 0 | 0, 1, 2, and 3 SPI clocking modes supported |
| TX8 | 0 | 1CB | 34 | 0 | 8 bits, clocking up to 48 MHz, 6 Mbits data rate max |
| UART | | | | | RS-232-compliant, burst rates up to 6 Mbits/sec |
| Low Level API | 0 | 2CB | 66 | 0 | Maximum supported data rate 115.2 kbits/second |

| Module | Analog | Digital | Flash | SRAM | Features |
|---|---|---|---|---|---|
| **Filters** | | | | | |
| BPF2 | 2 | 0 | 109 | 0 | Programmable mid-band gain, center frequency, and Q; sampling rates up to 1.0 MHz |
| LPF2 | 2 | 0 | 109 | 0 | Programmable gain, corner frequency, and damping ratio; sampling rates up to 1.0 MHz |
| **Generic** | | | | | |
| SCBLOCK | 1 | 0 | 20 | 0 | Fully parameterized for custom development |
| **Misc. Digital** | | | | | |
| DigBuf | 0 | 1 | 9 | 0 | Two Digital Buffers, Input1 can be inverted |
| DigInv | 0 | 1 | 36 | 0 | Output is digital inverted input |
| E2PROM | 0 | 0 | 439 + Size | 13 | Full byte-oriented EEPROM emulation |
| LCD | | | | | Uses HD44780, requires seven I/O pins |
| Bar Graph Enabled | 0 | 0 | 622 | 0 | |
| Bar Graph Disabled | 0 | 0 | 442 | 0 | |
| SleepTimer | 0 | 0 | 150 | 4 to 7 | 8, 16, or 32 bit tick counter, 3 types of timer functions |
| **MUXs** | | | | | |
| AMUX4 | 0 | 0 | 25 | 0 | Mux up to 4 inputs; high impedance, rail-to-rail input |
| AMUX8 | 0 | 0 | 44 | 0 | Mux up to 8 inputs; high impedance, rail-to-rail input |
| RefMux | 1CT | 0 | 32 | 0 | Allows routing internal references to external pins |
| **Protocols** | | | | | |
| USBFS | 0 | 0 | 500 | 10 | Optional HID class support |
| **PWMs** | | | | | Source clock rates up to 48 MHz |
| PWM8 | 0 | 1 | 67 | 0 | 8 bits |
| PWM16 | 0 | 2 | 89 | 0 | 16 bits |
| PWMDB8 | 0 | 2 | 35 | 0 | 8 bits, programmable dead band |
| PWMDB16 | 0 | 3 | 44 | 0 | 16 bits, programmable dead band |
| **Random Sequence** | | | | | Data input clocking up to 48 MHz, serial output bit stream |
| PRS8 | 0 | 1 | 41 | 0 | 2 - 8 bits |
| PRS16 | 0 | 2 | 54 | 0 | 2 - 16 bits |
| PRS24 | 0 | 3 | 82 | 0 | 2 - 24 bits |
| PRS32 | 0 | 4 | 93 | 0 | 2 - 32 bits |
| **Temperature** | | | | | |
| FlashTemp | 1 | 0 | 74 | 3 | Measures Flash temp, accuracy of ± 20C with no cal |
| **Timers** | | | | | Source clock rates up to 48 MHz, capture up to 24 MHz |
| Timer8 | 0 | 1 | 70 | 0 | 8 bits |
| Timer16 | 0 | 2 | 93 | 0 | 16 bits |
| Timer24 | 0 | 3 | 141 | 0 | 24 bits |
| Timer32 | 0 | 4 | 158 | 0 | 32 bits |

„CT“ steht für *continuous time block*, alle anderen analogen Blöcke sind *switched capacitor blocks*
“CB“ steht für *communication block*, alle anderen sind *basic blocks*

## 5.5 Weiterführende Dokumentation

| Titel | Beschreibung | Datei |
|---|---|---|
| IDE User Guide | Handbuch zum Programm PSoC-Designer | IDE User Guide.pdf |
| Technical Reference Manual | Referenzhandbuch zu den PSoC-Bausteinen | PSoC(R) Technical Reference Manual.pdf |
| Assembly Language User Guide | Handbuch zum Assembler | Assembly Language User Guide.pdf |

**Teletraining**

Cypress bietet auf seiner Webseite neben einer Vielzahl von Applikationsbeispielen auch englischsprachige Teletrainingskurse (Modul 1 bis 4) zum Download an. In diesen werden jeweils im ersten Teil auf anschauliche Weise die Komponenten und Funktionen der PSoC-Bausteine vorgestellt. Im zweiten Teile eines jeden Moduls wird die Programmierung mit dem PSoC-Designer Schritt für Schritt vorgeführt.

Die Teletrainingskurse Module 1 bis Module 4 sind als Video (englischsprachig) erhältlich. Der erste Teil ist jeweils zusätzlich als PDF- und Powerpoint-Datei verfügbar. Daneben gibt es noch Modul 5, welches z.Z. nur als PDF- und Powerpoint-Datei angeboten wird. Die Videos liegen in Form von selbstextrahierenden Zip-Archiven vor.

Dateien für Module 1:

Video: *psoc_pre_recorded_video_module_1__ introductory_module_11.exe*
PDF: *psoc_pre_recorded_video_module_1__introductory_module_12.pdf*
Powerpoint: *psoc_pre_recorded_video_module_1__introductory_module_13.zip*

Die verschiedenen Module behandeln folgende Themen:

Module 1: Introduction to PSoC
Module 2: Designing with PSOC
Module 3: Debugging
Module 4: Dynamic Reconfiguration
Module 5: Advanced Analog Design

Das Praktikum baut nicht auf den Teletrainingskursen auf. Diese werden nur als zusätzliche und vertiefende Informationsquellen genannt.