1. Installation PSoC Creator 3 auf Windows XP

1. CyUSB-Serial_SDK installieren **USB-Serial Software Development Kit** 2. KIT einstecken und Treiber installieren lassen Assistent für das Suchen neuer Hardware Mit diesem Assistenten können Sie Software für die folgende Hardwarekomponente installieren: USB-Serial (Single Channel) Falls die Hardwarekomponente mit einer CD oder Diskette geliefert wurde, legen Sie diese 0 jetzt ein. Wie möchten Sie vorgehen? O Software automatisch installieren (empfohlen) Software von einer Liste oder bestimmten Quelle installieren (für fortgeschrittene Benutzer) Klicken Sie auf "Weiter", um den Vorgang fortzusetzen. Weiter > < Zurück Abbrechen

Bild 1: Hardware-Assistent beim ersten Einstecken des KIT

Verzeichnis wählen,



Bild 2: Manuelle Treiber-Wahl

Hadin adi ino	
Rechtsklick	Systemeigenschaften
	Sustemwiederherstellung Ammalische Undates Bemote
Arbeitsplatz	Algemein Computersame Hardware Enweitet
Hibbiopiate	
1	Gerate-Manager
	Der Gerafer-Manager istet alle auf dem Computer installeiten Handwarenzieha auf Verwenden Sie den Geräferten Man oper um
Netzwerkumgebung	die Eigenschaften eines Geräts zu ändern.
	Geräte-Manager
1	
Papierkorb	Durch die Treibersignierung kann sichergeste 🖳 Gorzite Manager
	Installerte Treber mt Windows kompablet an en der uter hanningen und der uter hanningen und der können Sie ferstenen wie Treber üb
	aktualisiert werden sollen. Datei Aktion Ansicht /
	Trebersignierung Windo 🗢 → 💷 🖀 🎒 🔡 💌 🧏 🕿 🗶
Internet Explorer	
	Hardwareprofile
	Ober Hardwareprofile können Sie verschiede USB Serial Port (COM3)
	Kontigurationen einrichten und speichern.
	Figenschaften von USB Serial Port (COM3)
	Haidv B CypressUsbConsoleWindov
	Allgemein Port Settings Treiber Details
	Subjecternaurwerke
	Ca Engaberate (Human Inte
	DK Abbre DE ATA/ATAPI-Controller Data bits: 8
	A Mirror and andres Zeinene
	House and a noise and a noise and gauge Parity: None
	Betzwerkadapter
	Prozessoren
	Systemger ate Flow control: None
	i Goodaan
	Advanced Hestore Defaults
	OK Abbrechen

Nach der Installation die Portgeschwindigkeit auf 115200 setzen.

Bild 3: Einstellung der COM-Port-Parameter

Kann auch auf 9600 belassen werden, aber dann muss die Geschwindigkeit im PSoC Creator 3 angepasset werden (kommt später)

3. PSoc Creator 3 installieren (Typical ohne Änderungen) <u>PSoCCreatorSetup 3.0 sp1</u>

2. CY8CKIT-049-4xxx Testprogrammierung

Das CY8CKIT-049-4xxx Kit ist vorprogrammiert mit einer Firmware, die einen Bootloader enthält. Deshalb gibt es 2 Möglichkeiten, das Board zu programmieren.

- 1. Die UART bootloader Programmierung des PSoC 4 Devices benutzt das USB-Serial-Device als USB-UART Bridge.
- 2. Man kann den Programmer <u>CY8CKIT-002 PSoC® MiniProg3 Program and Debug Kit</u> benutzen um das Kit direkt zu programmieren und zu debuggen.

Achtung! Solange das PSoC4-Device auf dem Kit einen Bootloader enthält, können beide Methoden benutzt werden.

Seite 2 von 25

Wenn ein Projekt ohne Bootloader auf dem PSoC4-Device programmiert wurde, kann nur der MiniProg3 zum Programmieren benutzt werden!

Oder anders ausgedrückt:

Jedes Projekt kann mit dem MiniProg3 auf das Kit geschrieben werden- Aber nur bootloadable Projekte können mit der erstem Methode auf das Kit geschrieben werden.

Siehe auch <u>AN73854 - PSoC® 3, PSoC 4, and PSoC 5LP Introduction To Bootloaders</u> für weitere Informationen zum **bootloading**.

3. Schreiben eines CY8CKIT-049-4xxx Projektes mittels Bootloader

Das folgende Beispiel zeigt, wie ein Bootload-Projekt auf das PSoC4-Device mittels USB-Serial-Device und dem Bootloader-Host geschrieben wird.

Dafür muss das PSoC4-Device einen Bootloader enthalten und das Projekt muss als **bootloadable** konfiguriert sein.

Dies ist die Default Programmiermethode für neue User.

Die folgenden Schritte benutzen ein <u>Beispielprojekt von der KIT-Webseite</u>. (CY8CKIT-049-42xx Example Projects.zip)

- 1. PSoC Creator starten.
- 2. Beim ersten Start am Besten erst mal ein Arbeitsverzeichnis festlegen:



Bild 4: Wahl des Projekt-Verzeichniss

3. Das heruntergeladene Beispielprojekt (CY8CKIT-049-42xx Example Projects.zip) im Projektverzeichnis entpacken.

😂 PSoc4 RW auf "psf\Hom	me\Documents" (P:)	
Datei Bearbeiten Ansicht	Favoriten Extras ?	
📀 Zurück 🝷 🕥 🕤 🏂	👂 🔎 Suchen 😥 Ordner 💷 🕶	
Adresse 💥 P:\		
Ordner	× Name 🔺	
🗆 🧝 PSoc4 RW auf "psf\Ho	iome\Documents" (P:) 🔼 🚞 SCB_Bootloader_42xx	
🖃 🚞 SCB_Bootloader_4	_42xx	
🗉 🚞 Bootloadable I	Blinking LED.cydsn	
🗄 🚞 Dependencies	es e	
🗉 🚞 UART_Bootloa	pader.cydsn 🧹	
<		

Bild 5: im Projekt-Verzeichniss entpacktes Beispielprojekt.

4. Das Beispiel-Projekt öffnen

⊞ P	SoC Creator 3.0						l
Eile	<u>E</u> dit <u>V</u> iew Project B	<u>J</u> uild	<u>D</u> ebug	<u>T</u> ools	<u>W</u> indow	<u>H</u> elp	
	<u>N</u> ew	•			50		Ţ
	<u>O</u> pen	•	đ.	Project/W	'orkspace		_
	Example Project		2	Eile	Ctrl+O		
	Add	×.					

Bild 6: Menüpunkt Projekt/Workspace öffnen

Öffnen				? 🛐
Suchen in:	🗀 SCB_Bootload	der_42xx 💌	G 🤌 📂 🖽-	
Zuletzt verwendete D Desktop	Backup Bootloadable Bl Dependencies UART Bootload	linking LED.cydsn der.cydsn r		
igene Dateien		\backslash		
Sin Arbeitsplatz				
	Dateiname:	SCB_Bootloader	*	Öffnen
Netzwerkumgeb	Dateityp:	All Project Files (*.cyprj;*.cywrk)	~	Abbrechen

Bild 7: Projekt-Datei zum Öffnen wählen

Der PSoC Creator überprüft beim Öffnen auf evtl. vorhandene Updates der Komponenten und schlägt vor, diese zu laden.

Component Update Tool		? 🗙
The project you are loading may contain components that are you check for and apply any available component updates. D defects or incompatibilities that could affect your design.	e out of date. We strongly recomme Jut of date components may contai	nd n
Name	Available Versions	^
E UART_Bootloader		
E TopDesign		
Bootloader	~	
Bootloader [v 1.20]	1.20 🗸	
⊨ 🦣 <u>SCB P4</u>	×	
UART (v 1.10)	1.20 🗸	
e ev pins	×	
💽 P1_6 [v 1.90]	2.0 🗸	
Boot_P0_7 [v 1.90]	2.0 🗸	~
Damit werden die alten Versionen archiviert		
Archive before committing	Next Cancel	

Bild 8: evtl. Update Dialog beim Öffnen von Projekten

Workspace/Pr	oject Archiver
Source:	Workspace 'SCB_Bootloader'
Destination:	P:\SCB_Bootloader_42xx\Archive
	Compress Archive (zip)

Bild 9: Archivier-Dialog beim Öffnen von aktualisierten Projektdateien

Nach evtl. Archivierung werden die Projektdateien geöffnet und im Workspace Explorer gezeigt.

5. Mit Rechts-Klick im Workspace-Explorer auf **Bootloadable Blinking LED..** klicken und als **Aktives Projekt** setzen.



Das bootloadable Projekt mit Bootloader HEX und ELF Projektdateien verknüpft sein. Dies stellt sicher, dass die Firmware mit dem Kit-Device Code verknüpft wird.

Doppelklicken der Datei '**TopDesign.cysch'** unter '**Bootloadable Blinking LED**' öffnet die schematic view dieser Datei.



Bild 10: TopDesign datei des aktiven Projekts

Die Verknüpfung zu den Bootloaderdateien HEX und ELF ist in den **Dependencies** der **Bootloadable Komponente** hinterlegt.

Diese Dateien werden generiert, wenn das Bootloader-Projekt generiert wird. Für das 42xx Kit befinden diese ich in den Ordnern:

```
P:\SCB_Bootloader_42xx\UART_Bootloader.cydsn
\CortexM0\ARM_GCC_473\Debug\UART_Bootloader.hex
```

P:\SCB_Bootloader_42xx\UART_Bootloader.cydsn
\CortexM0\ARM_GCC_473\Debug\UART_Bootloader.elf



Bild 11: Einstellen der Booloader Pfade

 Mit Build > Build Bootloadable Blinking LED oder Shift+F6 wird nun das Projekt übersetzt.

SCB_Bootloader - PSoC Creation	eator 3.0 [P:\\Bootloadable Blinking LED.cydsn\TopDes
File Edit <u>V</u> iew Project Buil	Debug Iools Window Help
🕄 🔁 👌 💕 🖵 🖉 📇 🛗	Build All Projects F6 8% 🗸 🕄 🤤 🖕
i 🎬 • 👗 🗇 💕 👹 🔆 🗖	Clean All Projects
Workspace Explorer (2 projects)	Clean and Build All Projects
🖫 Ca 🔛	Build Bootloadable Blinking LED Shift+F6
🔯 Workspace 'SCB_Bootloade 🗖	Clean Bootloadable Blinking LED
😑 🔁 Project 'Bootloadable 🎬	Clean and Build Bootloadable Blinking LED
	Cancel Build Ctrl+Break
🖻 🙆 Source Files 🛛 🔌	Compile File Ctrl+F6
🛄 🖸 main.c 👔	Generate Application
E Project UART_Bootloa	Generate Project Datasheet
Bild 12: Build Menü	

Der Build-Prozess sollte ohne Fehler erfolgreich durchlaufen.

Output				
Show output from: All	• 🖹			L
cyelftool.exe -B	"P:\SCB_Bootloader	_42xx\Bootloadable	Blinking	LED.cydsn\Co
cyelftool.exe -S	"P:\SCB_Bootloader	42xx\Bootloadable	Blinking	LED.cydsn\Co
Flash used: 5734	of 32768 bytes (17	,5%).		
SRAM used: 1536 o	of 4096 bytes (37,5%	8).		
H	Build Succeeded: 06,	/21/2014 23:45:35 -		

Bild 13: Build Log

Ein Bootloadable-Projekt enthält neben der Bootloadable-Komponente immer auch das zusätzliche eingebundene Project "UART_Bootloader" welches den eigentlichen Bootloadprozess handelt.



Bild 14: Project "UART_Bootloader"

Es beinhaltet die Komponenten **Bootloader** und **UART** sowie etwas Logic zur Auslösung und Signalisierung des Bootload-Prozesses. In diesem Fall die auf dem KIT befindliche LED, welche über die PWM Komponente blinkt solange der Bootloader aktiv ist, sie der auf dem KIT befindliche Taster, welcher den Bootvorgang auslöst, wenn er beim Einstecken des Kits gedrückt wird.

Beachte!

Der Bootloader wird aktiviert, wenn der Taster auf dem Board beim Einstecken gedrückt wird.

7. Das Kit muss nun mit gedrückter Taste eingesteckt werden. Die blaue LED sollte jetzt in einem schnellen Frequenz (etwa 2 Hz) blinken, was anzeigt, dass der Bootloader aktiv und bereit zur übernahme der Firmware ist.

Achtung!

Dies muss jedes mal gemacht werden wenn generierte Firmware in das PSoC4-Device per Bootloader übertragen werden soll.

8. Jetzt muss das Bootloader-Host-Tool geöffnet werden (Tools > Bootloader Host)

SCB_Bootloader - PSoC Creator 3.0 [P:\SCB_Bootloader_42xx\UAF]
 File Edit View Project Build Debug Tools Window Help
 Tools Window Help

Bild 15: Menü Bootloader Host

Bootloader Host					
File Actions Help					
🖆 泽 BB 🥎 🛞					
File:					
Ports: Filters	Port Config	uration	UART 🔽	Port Information	
S USB Serial Port (COM3)	Baud	115200	~	Generic Serial Port	
	Data Bits	8	*		
	Stop Bits	One	~		
	Parity	None	~		
Log:					
12:15:09 AM - Selected device: USB Serial Port (0	COM3)				

Bild 16: Bootloader Host Einstellung des COM Ports

An dieser Stelle muss überprüft werden, ob der richtige COM-Port angezeigt und aktiviert ist und ob die Portparameter korrekt sind (siehe Installation des USB-Treiber Bild 3.)

Die Port-Filter Einstellungen sollten wie folgt geändert werden.

	Filters Port Cor	nfiguration	6
ŋ	Port Filters		0
	Show I2C Devices Show SPI Devices Show UART Devices Show USB Devices	;	
	VID: 0x04B4]	
	PID: 0xB71D]	
	Cancel 0	к	

Bild 17: Bootloader Host Port Filter

Das Bootloader-Host Tool zeigt nun nur noch den verfügbaren UART basierenden COM Ports.

 Nun muss das zu programmierende File über File > Open gewählt werden. Die Datei heisst "Bootloadable Blinking LED.cyacd" und befindet sich im CortexM0 Verzeichnis des Projekts.

불 Bo	ootloader Host	
File	Actions Help	
File:	P:\SCB_Bootloader_42xx\Bootloadable Blinking LED.cydsn\CortexM0\ARM_GCC_473\Debug\Bootloadable Blinking LED.cyacd	
Ports:	s: Filters Port Configuration UART V Port Information Generic Serial Port	

Bild 18: zu übertragende Firmware Datei

10. Anschließend kann das Kit mit der Programm-Schaltfläche geflasht werden.

불 Bo	otloader Host 📃 🗖 🏹
File	Actions Help
File:	P:\SCB_Bootloader_42xx\Bootloadable Blinking LED.cydsn\CortexM0\ARM_GCC_473\Debug\Bootloadable Blinking LED.cyacd
Ports:	: Filters Port Configuration UART V Port Information Generic Serial Port

Bild 19: Bootloader Host Flash Button

	Log:		C
	12:15:09 AM - 12:33:10 AM -	- Selected device: USB Serial Port (COM3) - Programming Started	
	12:33:12 AM	Programming Finished Successfully	
	Programming	completed in 1783ms.	
F	leadv		1

Bild 20: Flash Log

Wenn die Programmierung erfolgreich abgeschlossen wurde, startet das Kit auch gleich neu. Die LED sollte nun etwas langsamer Blinken.

Im Originalbeispiel "Bootloadable Blinking LED" ist der Unterschied recht gering, deshalb bietet es sich an, das Blink verhalten bei aktivem Bootloader so zu ändern, dass es eindeutig erkennbar ist.

Achtung!

Der Bootloader selbst kann nur mit JTAG oder SWD mittels PSoC Creator oder PSoC Programmer auf das Kit geflasht werden! (z.B. MiniProg3) Erst wenn der Bootloader auf dem Board installiert und aktiv ist, können bootloadable Projecte per Bootload-Operation übertragen werden.

4. Das Bootloader Basis Beispielprojekt (Bild 14)

Das CY8CKIT-049-4xxx Prototyping Kit ist vorprogrammiert mit einem einfachen LED-Blink Beispielprojekt. Dieses Beispiel nutzt die PWM Komponente um eine LED langsam blinken zu lassen. In diesem Projekt ist das BootLoader Base Projekt integriert. Der Bootloader Code ist im Detail im <u>UART Bootloader Projekt</u> auf der Cypress Webpage zu finden.

Das Bootloader Beispiel lässt LED schnell blinken wenn der Bootloader aktiv ist, stellt die UART Communikation für den Bootloadprozess sicher und ließt den Status des Schalters SW1 beim Einstecken des Moduls um den Bootprozess zu starten.

Ist der Schalter beim Einstecken (während der power-up time) nicht gedrückt, springt der Bootloader zur User-Application und startet diese.

Ist sie hingegen gedrückt, wartet der Bootloader auf den Transfer einer neuen User-Application, welche durch das Bootloader-Host-Tool eingeleitet werden kann. Während der Bootloader auf den Transfer wartet und während des Transfers blinkt die LED im ca. 3Hz Takt.

Das Verhalten des Bootloadprozessen kann komplett angepasst werden. Er muss dann aber z.B. per MiniProg3 übertragen werden.

5. Ein Bootloadable Beispielprojekt (User - Applikation)

Das mitgelieferte Bespielprjekt **Bootloadable Blinking LED Project** zeigt, wie eine Bootloadabler Applikation aufgebaut sein muss um per USB-Serial Controller ins Kit übertragen werden zu können.

Im der Bootloadablen Applikation werden folgende Komponenten benutzt:

- Bootloadable
- PMW
- Clock
- Digital Output Pin
- Digital Constants (logic HIGH/LOW)
- Off-Chip Komponenten (externer Widerstand, LED und VSS)

In diesem Beispiel wird die PWM Komponente dazu benutzt, eine LED, die an einem Output-Pin über einen Widerstand angeschlossen ist, zu treiben.

Die Bootloadable Komponente stellt sicher, dass der Applikationscode korrekt mit dem Flash Speicherbereich des Bootloaders auf dem Kit korrespondiert und es da keine Überschneidungen gibt.

Doolloadal		eci
	Bootloadable Bootloadable	
The Bootloadable compon	ent links the application to the Bootloader project	s .hex file
<u> </u>	PWM ov E count un E start cc E	
Clock III	line Ine Resistor	1 Vss
The PSoC 4 pin P1[6] is contained this pin to blink the LED at	onnected to LED1 on the board. The PWM compo regular intervals.	nent drives

Bild 18: Das Bootloadabel Blinking LED Projekt

6. Erstellen eines neuen Bootloadablen Projektes

Um ein neues Bootloadables Projekt zu erstellen, geht man folgendermaßen vor:

1. Auf der Startseite des PSoC Creators klicken wir auf Create New Project.



Bild 19: Create New Project

2. Auf dem erscheinenden New Project Fenster wählen wir Empty PSoC 4 Design.

N	ew Project		?	
	Design Other		⊲	Þ
	Empty Templates			^
	Empty PSoC 3 Design	Creates a PSoC 3, 8 bit, design project.		
	Empty PSoC 4 Design	Creates a PSoC 4, 32 bit, design project.		
	🔲 Empty PSoC 5LP Design 🔪	Creates a PSoC 5LP, 32 bit, design project.		≡
	🖃 PSoC 3 Starter Designs			
	ADC_DMA_VDAC	Shows how to transfer data from an ADC to a DAC using DMA with no CPU intervention.		
	🔁 DelSig_16Channel	Shows a 16-channel, 12-bit Delta Sigma ADC in PSoC 3 sequenced in hardware; samples are transferred from ADC to SRAM using DMA - without processor intervention.		
	🔁 DelSig_I2CM	Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.		
	DelSig_12CS	Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over I2C.		
	🔁 DelSig_SPIM	Shows the 16-bit differential ADC, hardware multiplexed into 8 channels and transported over		
	🔁 Filter_ADC_VDAC	Shows how to filter an analog input all in hardware, and provides all the DMA setup to transfer the data from the ADC directly to the Digital Filter Block, bypassing the processor.		
	😼 HW_Fan_Control_with_Alert	Shows how performing fan control in hardware completely frees up the CPU.		
	🖃 PSoC 4 Starter Designs 🛛 🗡			~
ſ	Name: Test1 new Bootloadable			
	Location: P:\			
l	+ Advanced -			_

Bild 20: New Project Fenster

3. Mit dem + Button öffnen wir die erweiterten Einstellungen. Hier können wir weitere Angaben wie:

Anlegen im aktuellen Workspace oder in einem neuen Workspace, evt. anderes Device -

Achtung! CY8CKIT- 049-42XX kit benötigt das CY8C4245AXI-483 Device. die Grösse des Sheet Templates (da kann es evtl besser gedruckt werden)

4. Als Applications Type setzen wir hier Bootloadable!

5. Ein Klick auf Ok zeigt das leere Projekt

Advanced		
Workspace:	Create New Workspace	✓
Workspace Name:	Test1 new Bootloadable	
Device:	CY8C4245AXI-483 - (Default PSoC 4 Device)	×
Sheet Template:	A4 (11.7" x 8.3")	×
Application Type	Bootloadable	
		OK Cancel

Bild 21: erweiterte Einstellungen des New Project Fenster

6. Falls die Schematic Ansicht nicht geöffnet ist (Das Blatt aus dem die Komponenten plaziert werden), öffnen sie diese durch einen Doppelklick auf die Datei mit der Endung **.cysch** im Workspace Explorer.

Klicken sie auf den Page 1 Tab in der Schematic Ansicht, falls es sich um mehrere Seiten handelt und die Seite 1 nicht aktiv ist.

Die Schlüsselkomponente, die als erstes zugefügt werden muss, ist die Bootloadable Komponente, welche dazu da ist, den Bootloadablen Applicationscode zu generieren. Per Drag and Drop ziehen sie diese auf die leere Seite.



Bild 22: Die Bootloadable Komponente wird zugefügt.

7. Doppelklicken sie die Bootloadable Komponente um diese zu konfigurieren.

Sie wird genauso wie im Beispiel-Projekt konfiguriert.

Configure 'Bootloadab	le'	
Name: Bootloadable		
General Depend	encies Built-in	4 Þ
Application version:	0x0000	
Application ID:	0x0000	
Application custom ID:	0x0000000	
Manual application in	age placement	
Placement address:	0x0000000	
Datasheet	ОК	Apply Cancel

Bild 23 Configuration Bootloadable Komponente

Wichtig ist nun die Pfadangabe des HEX und ELF Files vom **UART Bootloader project**, welches zum Kit gehört. (Bild 6)

Configure 'Bootloadable'	? 🛛		
Name: Bootloadable			
General Dependencies Built-in	4 0		
Bootloadable projects require a reference to the associated Bootloader project's HEX and EI files. The HEX files extension is *.hex. The ELF files extension depends on IDE and can be *.elf, *.out, *.axf, or other.	_F		
Bootloader HEX file:			
P:\SCB_Bootloader_42xx\UART_Bootloader.cydsn\CortexM0\ARM_GCC_473\Debug\UART			
Browse			
Bootloader ELF file:			
P:\SCB_Bootloader_42xx\UART_Bootloader.cydsn\CortexM0\ARM_GCC_473\Debug\UA	\RT		
Browse			
Datasheet OK Apply Cano	el		

Bild 24: Bootloadable Komponente Pfade zu den HEX und ELF Dateien

Die Pfade sind:

```
P:\SCB_Bootloader_42xx\UART_Bootloader.cydsn\CortexM0\ARM_GCC_473\Debug
\UART_Bootloader.hex
P:\SCB_Bootloader_42xx\UART_Bootloader.cydsn\CortexM0\ARM_GCC_473\Debug
\UART_Bootloader.elf
```

Das so erzeugte leere Projekt ist noch nicht mit Leben erfüllt, es besitzt noch keine Code. Es soll der Einfacheit Halber mit einem simplen Code ausgerüstet werden.

Die LED auf dem Board soll mit Tastendruck des internen Tasters ein und aus geschaltet werden könne. (Togglefunktion)

Das Ganze soll zuerst in ,Software' implementiert werden. Dazu brauchen wir:

- einen Digital Output Pin
- eine LED Hier nehmen wir die LED auf dem Kit
- einen Vorwiderstand für die LED befindet sich ebenfalls auf dem Board

- einen Digital Input Pin
- einen Taster Der Einfachheithalber nehmen wir den auf dem Board befindlichen Taster
- und dem GDN (VSS) Signal

🗄 🔯 CapSense 5 🗄 🔯 Communications Configure 'cy_pins' **?** × 🕀 🔯 Digital 🗄 📷 Display Name: P1_6_LED Istanzname 🖻 🔯 Ports and Pins 4 Þ Pins Mapping Clocking Built-in 🔉 Analog Pin (v2.0) Number of Pins: 1 X 🗛 🕈 🕈 Digital Bidirectional Pin [v2.0] 🚯 Digital Input Pin [v2.0] [All Fins] Type General Output Input 🐉 Digital Output Pin (v2.0) P1_6_LED_0 Analog Preview: Beotloadable [v1.20] Pin 0 von n falls 📃 Digital Input F mehrere angelegt Bootloader [v1.20] werden Clock [v2.20] 🗹 Digital Output Die Temperature [v1.0] HW Connection Emulated EEPROM [v1.10] 🔲 Output Enable Bolobal Signal Reference [v2.0] 🔠 ILO Trim [v1.0] Bidirectional Interrupt [v1.70] 🗹 Show External Termina 🗄 📷 Thermal Management Fan Controller [v3.0] RTD Calculator [v1.20] ΟK Datasheet Cancel Thermistor Calculator [v1.20] Thermocouple Calculator [v1.20] TMP05 Interface Macro [v1.10] äußerst umfangreiche P1_6 LED Hilfetext zur Pin -Komponente

Wir kommen also für unser erstes Projekt ohne externe Bauelemente aus.

Bild 25: Eine DigitalOutputPin Komponente wird angelegt

Wir ziehen eine DigitalOutputPin Komponente auf das TopDesign-Sheet und erzeugen dadurch eine Instanz dieser Komponente.

Diese wird nun durch Doppelklick auf die Instanz entsprechend konfiguriert:

Die Komponente erhält einen Namen - In unserem Fall P1_6_LED. Zur besseren Orientierung verwenden wir hier folgende Namenskonvention:

P1_6 -> der physikalische Pin P1.6 - Das ist der Pin, an dem Laut Unterlagen die LED angeschlossen ist.



Eine Notwendigkeit dieser Benennung gibt es nicht. Die Zuordnung, welcher physikalische Chip - Pin zu unserer Pin-Komponente gehört MUSS ohnehin noch separat erfolgen. Die gewählte Namenskonvention ist wollkürlich und dient hier nur unserer Übersicht.

Bild 26: Auszug aus der Kit-Pinbelegung

Wichtig im Tab Type sind hier noch die folgenden Checkboxen:

- **Digital Output** legt fest, dass es sich um einen Ausgang handelt. Der Signalfluss wird durch einen kleinen Pfeil links vom Pin-Symbol sichtbar gemacht.
- **HW Connection** Es erscheint eine Connect-Box links neben dem Pin-Symbol. Ist diese Box NICHT aktiviert, lässt sich der Pin NUR per Software API programmatisch ansprechen. Eine direkt "Verdrahtung" mit anderen Hardware Komponenten ist dann

nicht möglich. Ist die Box aktiviert, kann der Pin sowohl softwaretechnisch als auch per Hardware-Komponente angesprochen werden.

• Show External Terminal - rechts neben dem Pin-Symbol erscheint eine Connect-Box die die mit einem blauen "Draht" mit dem Pin verbunden ist. Über diese Box können EXTERNE Bauteile zur Dokumentation mit dem Pin verdrahtet werden. Diese Verdrahtung muss sich nicht auf dem Kit befinden! Für die Arbeitsweise des Pins hat sie keinerlei Bedeutung! Alle hier evtl. angeschlossenen Bauteile dienen lediglich der Dokumentation. Alle externen Bauteile werden in blauer Farbe dargestellt.

Type General Input Output			
C Drive Mode	Initial State:		
Strong Drive 😽	High (1) 🛛 🔽		
	Minimum Supply Voltage:		

Bild 27: DigitalOutputPin Komponente Tab General

Im Tab **General** kontrollieren wir noch die Einstellungen: Drive Mode -> **Strong Drive** Initial State: -> **High (1)**

Type G	eneral Input	Output			
Slew Rate:	Fast	~			
Drive Level:	Vddio	~			
Current:	4mA source, 8mA sink 💌				
Output Mode: Transparent 💌 🗖 OE Synchronized					

Bild 28: DigitalOutputPin Komponente Tab Output

Die Einstellungen im Tab Output mussen ebenfalls kontrolliert werden und ggf. gemäß Bild 28 eingestellt werden.

Die einzelnen Optionen sollen hier erst einmal nicht weiter betrachtet werden. Dazu sei auf die hervorragende und umfangreiche Dokumentation verwiesen, die man direkt aus dem Configurations-Fenster erreichen kann (Datasheet - ganz links unten) siehe Bild 25.

Ok, damit wir es nicht vergessen und beim Build-Prozess einen Fehler bekommen, wollen wir auch gleich unserem Pin den physikalischen Pin des Kits zuweisen.



Bild 29: physikalische Pin Zuweisung

Die Pinzuweisung geschieht in der Datei mit der Endung .cydwr im Reiter Pins.

Wir sehen rechts eine Tabelle mit unseren generierten Pin(s), die beim ersten Aufruf noch keinem Port und keinem Pin zugeordnet sind.

Wir weisen unserer LED dem **Port P1[6]** zu. Dies ist ja laut Dokumentation der Port für die auf dem Kit befindliche LED. Die LED ist auf der Platine direkt mit dem Pin 43 des Schaltkreises verbunden, was sich ohne Lötkolben auch nicht ändern lässt, weshalb der PIN automatisch eingestellt wird und die Checkbox **Lock** aktiviert ist.

Nach erfolgter Zuweisung wird der entsprechende Pin links am Schaltkreis mit dem Namen unserer LED beschriftet (P1_6_LED) und bekommt eine graue Farbe.

Hinweis!

Notice List				
🕉 4 Errors 🚺 0 Warnings 🔯 0 Notes Go To Error View Details				
Description				
⊕ 😵 sdb.M0025:No input on Instance "P1_6_LED", terminal "y_0".				
⊕ 😵 sdb.M0050:Terminal "y_0" requires connection when it is visible.				
B Sdb.M0025:No input on Instance "P1_6_LED", terminal "y_0".				
⊕ 😵 sdb.M0050:Terminal "y_0" requires connection when it is visible.				
Output Notice List				

Bild 30: Notice List Unsere Notice List - standardmäßig am unteren Rand des Programmfensters dargestellt, zeigt uns noch eine Reihe von Fehlern an. Im Wesentlichen besagen diese, dass da wohl noch Verbindungen fehlen. Da hat er ja recht, unser Pin schwebt da noch recht sinnfrei in der Gegend rum. Also sollten wir den erst einmal weiter verdrahten.

Beginnen wir mit den externen Bauteilen, einem Widerstand und einer LED, sowie dem GDN (VSS) Signal, oder schlicht der Masse.

				Cypress Off-Chip
		· · · · · · · · · · · · ·		🔲 Off-Chip Component Catalog
				🖻 🐼 Active
				🗄 🐼 Diodes
<u></u>			<u></u>	🗄 🔯 Electro-Mechanical
Configure 'Resistor'			2 🗙	🛱 🔯 Passive
				Capacitor [v1.0]
Name: R_1				🗹 Crystal [v1.0]
				Fuse (v1.0)
Basic Built-in				Inductor [v1.0]
Parameter	Val	lue		Potentiometer [v1.0]
Include_In_Netlist	true			Resistor [v1.0]
Instance_Name_Visible	true			Transformer [v1.0]
Malua	500		· · · · /	e e e e e e e e e e e e e e e e e e e
value	300			
				🛨 🐼 Transistors
Parameter Information				
Value: Sets the resistance	of the resistor used			
Value: Tune: string				
.,,,,				
Datasheet	ОК	Apply Cance	el 🔰	
	, a cest Rul	🔆 🖌 🖌 🖌 🕹 🖌 🕹 🖌 🕹 🖌		
· · · · · · · · · · · · · · · · · · ·	1_6_LED	A		
	10K			

Bild 31: Off-Chip - Komponente Resistor

Aus den **Off-Chip Komponenten** fischen wir uns die **passive Komponente Resistor** raus und nachdem wir ihn platziert haben, öffnen wir mit einem Doppelklick das zugehörige Configurations-Fenster.

Im Reiter Basic ändern wir den Widerstandswert (Value) auf 560, weil der auf dem Kit verbaute LED Vorwiderstand eine Größe von 560 Ohm hat.

Seinen Namen dar er behalten und alle anderen Einstellungen auch. Nach bestätigen mit OK sollte sich der Widerstandswert von 10K auf 560 geändert haben.

CY8CKIT-049-4xxx PSoC 4 Prototyping Kit Guide Tipps



Achtung! Unsere LED wird gegen Masse geschaltet. Es fließt also "Source Current" und davon kann der Port nur 4mA bereitstellen. Würde die LED gegen Vdd geschalten, fließt Sink-Current. Und davon kann der Pin 8mA bereitstellen.

Ok, die Bauteile sind plaziert. Nun müssen sie noch korrekt verdrahtet werden. Dazu wird

das Wire - Symbol am rechten Fensterrand verwendet.



Bild 34: Mit dem Wire - Tool verbundene Elemente

Ist das Wire - Tool aktiv, erfolgt die Verbindung indem man bei den Start-Connector klickt und anscließend den End-Connector.

Die Verbindung erscheint als gestrichelte blaue Linie und die Connectoren verschwinden.

Wir könne bis hierhin schon mal einen Test durchführen. Allerdings müssen wir unserem Pin noch ein Eingangsignal zuweisen, weil es sonst in der Luft hängt. Später soll dises Signal ja von unserm Schalter kommen.



Bild 35: Die Pin-Komponente erhält ein Eingangssignal Seite 21 von 25

Hinweis!

Das Eingangssignal ist nötig, weil wir bei der Configuration der Pin-Komponente die Checkbox **HW Connection** aktiviert hatten (Bild 25). Soll der Pin ausschließlich per Software angesprochen werden, braucht die Checkbox **HW Connection** NICHT aktiviert zu werden. Dann ist auch das **Logical High(1)** Signal nicht nötig

Ohne weitere Maßnahmen würde die LED würde die LED einfach an sein, wenn wir das Programm an da Kit übertragen.

Deshalb platzieren wir noch ein wenig Code zum Blinken in die main.c dem C-Startprogramm.



Bild 36: Eine kleine Blink Sequence im Startprogramm

Hinweis!

Aber halt, vorher sollten wir das Projekt schon mal "Builden", denn erst danach sind dem C-Compiler alle erforderlichen Projekt-Komponenten bekannt. Unterlässt man dies, dann arbeitet insbesondere Intellisense noch nicht korrekt und es werden z.B. spezifischen Komponentenfunktionen z.B. zum Beschreiben der P1_6_LED Komponente nicht angezeigt, da dieser erst nach dem ersten Build-Prozess bekannt sind.

Ok, nach dem ersten Bild schreiben wir also den Code wie in Bild 36 innerhalb der for() Schleife ein.

Der Code ist recht unspektakulär und entsprechend im Quellcode dokumentiert. Anschließend "**Builden**" wir des Projekt und wenn wir keine Fehler erhalten, können wir das Erzeugte Projekt mittels Bootloader in das Kit schreiben. Beginnend bei Bild 18.

Achtung!

Vergessen sie nicht im Bootloader Host Tool den Pfad zur korrekten **.cyacd Datei** des aktuellen Projektes einzugeben. Sie befindet sich üblicherweise im **...ARM_GCC_473\Debug\..** Verzeichnis zu finden.

Also:

- A. Bootloader Host Tool aufrufen
- B. Kit mit gedrückter Taste einstecken (Led sollte schnell blinken)
- C. Pfad der .cyacd Datei anpassen (Bild 18)
- D. Firmware übertragen (Bild 19)

Nach erfolgreicher Übertragung sollte die LED sofort unsymmetrisch (lange On-Zeit/kurze OFF-Zeit) blinken.

Hat alles wie gewünscht funktioniert, können wir den Schalter implementieren.

Achtung!

Der auf dem Kit verbaute Schalter ist zwischen dem Kit-Anschluss P0.7 und GND angeschlossen.

Da die LED ebenfalls auf GND liegt, muss das Schaltersignal invertiert werden, wenn die LED bei Tasterdruck leuchten soll.

In diesem Fall benötigen wir noch einen Inverter, der natürlich als Inverter-Komponente zur Verfügung steht.

Wir ändern das TopDesign.sysch wie folgt:



Bild 37: Die LED per Schalter ein/aus schalten komplett in Hardware

Am DigitalInputPin hängt der auf dem Kit verbaute Schalter dran. Wie beim DigitalOutputPin müssen wir diesen erst noch dem richtigen Chip-Pin zuordnen, was wir wieder in der .cydwr Datei erledigen.



Bild 38: Den DigitalInputPin "verdrahten"

Der DigitalInputPin muss noch konfiguriert werden:

	P1_6_LED 560
Configure 'cy_pins'	
Pins Mapping Clocking Built-in Number of Pins: 1 2 + [All Pins] Type G [All Pins] Type G [All Pins] Type G [Analog ✓ Digital Input ✓ HW Cor Digital Output ✓ HW Cor Output E	4 b VSS eneral Input Output Input Preview: Besistive Pull Up Image: State High (1) Instance Image: State Image: State Image: State Image: State Image: State Image: State Image: State
Datasheet	Anoly Cancel

Bild 39: Konfigurationseinstellungen des DigitalInputPin

Im Tab **Typ** gemachten Einstellungen sind nötig um den Pin mit Hardware "verdrahten" zu können.

Auch die im Tab General vorgenommenen Einstellungen sind wichtig.

Der Schalter schaltet gegen GND(Masse). Im offenen Zustand würde der DigitalInputPin also offen sein. Also einen undefinierten Zustand einnehmen, was vermieden werden sollte.

Er bekommt also einen PullUp (Einen Widerstand gegen Vdd) - hat also im offenen Zustand einen sicheren High-Pegel. Außerdem bekommt er den Initialwert High (Schalter offen)

Nach einem erneuten Build - Lauf kann die neue Firmware mit dem Host-Tool auf das Kit übertragen werden.

Seite 24 von 25

Die LED sollte nun leuchten, wenn der Schalter gedrückt ist, sonst nicht.