Bachelorarbeit

Sven Reimer

Erprobungsplattform für Mikrocontroller-FPGA-Kommunikationsinterface

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Sven Reimer

Erprobungsplattform für Mikrocontroller-FPGA-Kommunikationsinterface

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Angewandte Informatik
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Bernd Schwarz
Zweitgutachter : Prof. Dr. rer. nat. Thomas Canzler

Abgegeben am: 20. März 2011

Sven Reimer

Thema der Bachelorarbeit

Erprobungsplattform für Mikrocontroller-FPGA-Kommunikationsinterface.

Stichworte

4-Phasen-Handshake, asynchrone Kommunikation, Datensynchronisation, Master-Slave-Anwendung, FPGA Spartan3 XC3S1200E, µController ARM7 TDMI LPC2468FET208, NEXSELv2-Board, NEXYS2-Board, SECURITYv2.1-Board, LPC-Stick/TI-Board

Kurzzusammenfassung

Diese Bachelorarbeit ist in das Projekt: Fahrerlose Autonome Transportsysteme (*FAUST*) und die Studienlehrveranstaltungen: Computer Engeneering/Praktikum *(CE/P)* der HAW Hamburg integriert. Der Schwerpunkt ist die diskrete Kopplung einer FPGA Spartan3E basierten SoC-Plattform und eines ARM7 TDMI µController-Evaluationsboards, zum Aufbau einer Kommunikationserprobungsplattform. Diesbezüglich wurden die Hardware-Interfaces zwischen den FPGA- und µController-I/Os, sowie die jeweiligen VHDL- und Software-Module zur Steuerung des Kommunikations- und Datentransfers entwickelt. Das FPGA übernimmt in der Master-Slave-Konfiguration die Funktion eines Beschleunigers.

Sven Reimer

Title of the paper

Evaluation platform for a microcontroller-FPGA communications interface.

Keywords

4-phase handshake, asynchronous communication, data synchronisation, master-slave application, FPGA Spartan3 XC3S1200E, µController ARM7 TDMI LPC2468FET208, NEXSELv2 Board, NEXYS2 Board, SECURITYv2.1 Board, LPC-Stick/TI Board

Abstract

This bachelor thesis is in the project: autonomous driverless transport systems (*FAUST*) and the study courses: computer engineering/training *(CE/P)* of the HAW Hamburg. The focus is the discrete coupling an FPGA-based SoC platform Spartan3E and one ARM7 TDMI microcontroller evaluation boards, to build a communications test platform. This re-gard developed the hardware interface between the FPGA and µController-I/Os, and the respective VHDL and software modules to control the communication-ons and data transfers. The FPGA controls in the master-slave configuration, the function of an accelerator.

Vorwort

Die vorliegende Bachelorarbeit wurde an der Hochschule für Angewandte Wissenschaften Hamburg, im Fachgebiet Technische Informatik des Departments Informatik, erstellt. Die Betreuung erfolgte durch Herrn Prof. Dr.-Ing. Bernd Schwarz, bei dem ich mich an dieser Stelle recht herzlich für die sehr gute Zusammenarbeit und zwischenmenschliche Hilfe bedanken möchte. Desweiteren bedanke ich mich bei allen Mitarbeitern des Labors für Technische Informatik für die freundliche Unterstützung meiner ausgeführten Arbeit. Hervorzuheben ist die uneingeschränkte Hilfsbereitschaft von Herrn Bruno Carstensen.

Im Besonderen möchte mich bei meinen Eltern und Geschwistern für die großartige Unterstützung während des ganzen Studiums bedanken.

Die Bachelorarbeit widme ich meinem verstorbenen Vater.

Inhaltsverzeichnis

Anhang

1. Einleitung

Das Department Informatik der Hochschule für Angewandte Wissenschaften (HAW) Hamburg bietet in dem Forschungsprojekt FAUST (Fahrerassistenz- und Autonome Systeme) eine Umgebung für die Entwicklung von eingebetteten Systemen, die sich an den Vorgaben der derzeitigen Industriestandards orientiert.

Ziel dieser Arbeit und Hauptschwerpunkt ist die Hardware-Kopplung, zwischen einer FPGA-basierten System-on-Programmable-Chip Plattform mit einem Spartan3E™ und einem µController ARM7™ TDMI mit Evaluation-Board, auf einer Erprobungsplattform als Funktionsmuster. Zum Aufbau einer individual konfigurierbaren TI-Development-Plattform NEXSELv2, für die Ausbildung der Studenten im Studienfach CE/CEP. Dazu sind die Hardware-Interfaces zwischen FPGA- und µController-I/Os, sowie die jeweiligen VHDL- und Software-Module zur bidirektionalen Datenfluss- und Steuersignalkopplung zu entwickeln. Die SoC-Plattform übernimmt in dieser Peripheriekonfiguration die Funktion eines Coprozessor zur beschleunigten digitalen Signalverarbeitung. Den technischen Schutz der Hardware-I/Os und die Betriebssicherheit, vor fehlerhafter Ansteuerung und oder Beschaltung gewährleistet das SECURITYv2.1-Board. Dieses ist eine Komponente der Erprobungsplattform NEXSELv2.

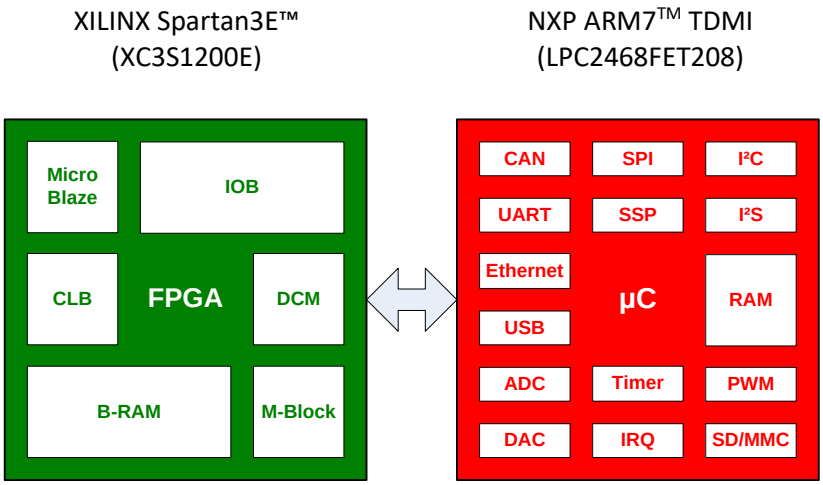


Abb. 1.1: Hardwarekomponenten der NEXSELv2-Plattform

Das Schaltungskonzept der in dieser Arbeit beschriebenen Erprobungsplattform erfüllt alle Anforderungen aktueller integrierter Systeme, welche einerseits eine Vielzahl an Sensor- und Kommunikationsschnittstellen und andererseits hohe Rechengeschwindigkeiten für die digitale Signalverarbeitung erfordern. FPGA-basierte SoC-Plattformen können die erforderlichen Datenraten mit parallelen DSP-Funktions- und integrierten Prozessorelementen bei niedrigen Taktfrequenzen erfüllen.

Typische µController bieten hingegen eine Vielzahl Funktionseinheiten und Schnittstellen, wie CAN, SPI, I^2C und ADC/DAC sowie verschiedene andere Peripherien zur Sensor- und Aktorkopplung.

Der Entwurf von Hardwareplattformen zur Realisierung von autonomen Regelanwendungen ist durch die breite Palette an leistungsfähigen Mikroprozessoren und FPGAs ein kostengünstiges Unterfangen. Viele Anwendungen benötigen nicht viel mehr zusätzliche Komponenten als die zur Erfassung von Eingangssignalen erforderlichen Sensoren, sowie einige elektronische Bauelemente. Sobald allerdings ein Bedarf an Filterfunktionen oder Visualisierungskomponenten zur Benutzerinteraktion besteht, deren Informationsgehalt über das Maß an Information hinausgeht, die durch ein kleines Display dargestellt werden können, wird ein leistungsfähigerer Chip benötigt. Hierdurch entstehen weitere Randbedingungen wie Mehrkosten und erhöhter Energiebedarf. Nicht zuletzt aus diesen Gründen sind Alternativen gefragt, welche diesen Bedarf erfüllen, ohne die Kosten oder den Energieverbrauch entscheidend zu beeinträchtigen. Bei vielen Systemen wird die erweiterte Benutzerinteraktion auf einen externen Rechner ausgelagert, der über ein Interface mit der Plattform verbunden wird. Eine bessere Alternative ist das FPGA.

Die in dieser Arbeit entstandene Bibliothek richtet sich an Entwickler von Software-Anwendungen für eine µController-FPGA-basierte Plattform, die einen schnellen Einstieg in die Entwicklung von Hardware-Software-Codesign-Konzepten wünschen. Viele konventionelle Erprobungsplattformen bieten nicht den Funktionsumfang und die Variabilität der externen Peripherie, wie die NEXSELv2-Plattform.

Die vorliegende Bachelorarbeit gestaltet sich folgendermaßen im Aufbau:

Das **2. Kapitel** umfasst eine Einführung und die konzeptionelle Systemübersicht der SoC-Plattform und derer integrierten Hardwarekomponenten, sowie eine Beschreibung des funktionalen Kommunikationstransfers des NEXSEL2-Gesamtsystems.

In **Kapitel 3** werden die verwendeten, bereits vorhandenen Hardwarekomponenten des NEXYS2-Boards [11] und LPC-2468-Sticks der NEXSEL2-Plattform vorgestellt. Diese Umfasst die externe Peripherie der Boards, die internen Module und I/O-Interface des FPGAs Spartan3E™ XCS1200E [10] und dem µC ARM7™ TDMI LPC2468FET208 [5]. Desweiteren werden die eingesetzten Entwicklungswerkzeuge kurz beschrieben.

Das Funktionskonzept des SECURITYv2.1-Boards, für die bidirektionale Kommunikation, wird im **4. Kapitel** detailliert erläutert. Zunächst werden der Aufbau und die Funktionsübersicht beschrieben. Nachfolgend werden die Partitionierung und Parametrisierung der Treiberstufen, zur statischen oder dynamischen Steuerung der Schutzschaltung durch den µController als Mastersystemeinheit, vorgestellt. Im Anschluss werden die messtechnischen Latenz- und Umschaltzeiten der Hardwarekomponenten zusammengefasst.

Das **5. Kapitel** umfasst die konkrete Realisierung des asynchronen Kommunikationsinterfaces. Neben einer Einleitung über die Anwendung, werden in diesem Teil die Protokoll- und Transferstruktur des 4-Phasen-Handshakes und die Steuerungs- und Kommunikationsmodi, sowie die Synchronisation der Steuer- und Datensignale beschrieben. Desweiteren wird der Zustandsautomat des SoC-Beschleunigers, mit Steuer- und Datenpfad, erläutert. Das verwendete, in der ISR gekapselte, Polling-Verfahren ermöglicht die Darstellung der Transferzeiten.

Die Zusammenfassung der Inhalte und Ergebnisse dieser Arbeit werden im **6. Kapitel** vorgestellt.

In **Kapitel 7** sind alle Verzeichnisse tabellarisch erfasst.

Zusätzliche detaillierte Informationen und technische Daten in tabellarischer oder grafischer Darstellung sind im **Anhang** erfasst.

2. Konzeptionelle Systemübersicht der NEXSELv2-Plattform

Die NEXSELv2-Plattform ist eine Hardware-Kopplung, bestehend aus zwei Basiseinheiten, den Development-Boards NEXYS2 [DIGI], TI-Board mit einem LPC-2468-Stick [HAW, HITEX] und der Schutzschaltung des SECURITYv2.1-Boards. Die Hardware-Schnittstellen mit jeweils 38 I/O-Leitungen des FPGAs (vgl. Abb. 2.1) und des µControllers sind über das SECURITYv2.1-Board diskret verbunden. Alle Komponenten des Funktionsmusters können somit auch separat betrieben werden.

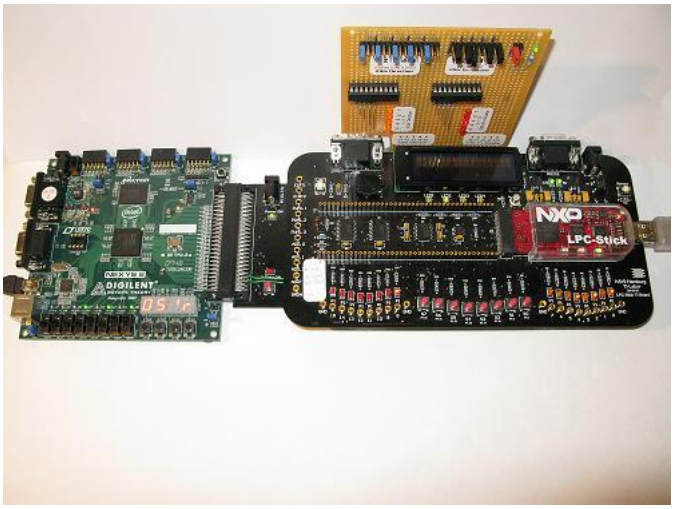


Abb. 2.1: Erprobungsplattform NEXSELv2

Die standardisierten Stecker und Buchsen des Funktionsmusters ermöglichen weitere externe Peripherie-Komponenten anzuschließen. Diese können aktiv über die Development-Boards versorgt werden, sofern der maximale Strom des Netzteils oder USB-Anschlusses nicht überschritten wird.

Das im FPGA eingebettete Prozessorelement zur Steuerung des 10bit Datentransfers als Slave-Request-Anwendung, arbeitet mit synchronisierten Signalen über ein 4-Phasen-Handshake (vgl. Kapitel 5). Die SoPC-Plattform übernimmt in dieser Peripheriekonfiguration die Funktion eines Coprozessor zur beschleunigten digitalen Signalverarbeitung. Auf dem µController läuft die Master-Anwendung, welche den bidirektionalen Datenfluss- mit dem 4-Phasen-Handshake-Protokoll und die Signalkopplung steuert.

Über die Adressierungsleitungen können Prozessorelemente im FPGA für die Signalverarbeitung ausgewählt werden.

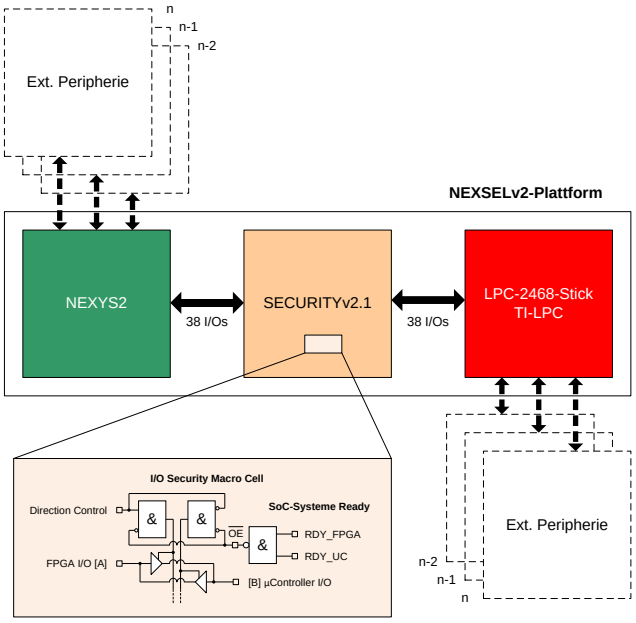


Abb. 2.2: NEXSELv2-Plattformkonzept

Die Erprobungsplattform NEXSELv2 hat nachfolgend aufgeführte Peripherie-Module und –Interfaces:

Mikrocontroller

- Chipsatz: ARM7™ TDMI-S LPC2468FET208
  16/32bit Core mit ISP und IAP
  512KB Fast Flash ROM und 98KB SRAM
  Variable Taktfrequenz 2..72MHz
- USB2 RW I/O – Programmdaten / Debugging
- 10bit ADC/DAC mit maximal VU 5V0
- 2x PWM I/Os – 32bit counter und prescaler
- 4x 32bit Timer – Soft- und Hardware gesteuert
- 3x I²C- und eine I²S-Schnittstelle mit max. 1MHz
- 32bit CAN nach ISO 11898-1 mit max. 1Mb/s
- 2x UART-Interfaces
- 2x SPI- und SSP-Schnittstellen
- 66 FastGPIO mit zwei ext. IRQ-Leitungen
- USB-A/B device/host/OTG mit 4KB RAM

FPGA

- Chipsatz: FPGA Spartan3E™ XC3S1200E
  Taktfrequenz 50MHz
- 16MB Flash-RAM (PSD) und Flash-ROM
- MicroBlaze – Embedded Prozessorsystem
- USB2 RW I/O – JTAG und Programmdaten
- UART EIA/TIA (RS-232 konform)
- VGA I/O
  8bit Farbtiefe mit HS/VS
- PS/2 I/O Mini DIN
  Mouse- und Keyboard-Ansteuerung
- 24x User I/Os
  LEDs, Taster und Schalter
- 8x 4fach und 4x 8fach Pmods Connectoren

Schutzschaltung

- Kontrolle der Netzversorgung, Boot-, Reset-Sequenz
- Steuerung des Datentransfers zwischen FPGA und µController
- Fünf 20MHz Bustreiberstufen mit 38 I/O-Leitungen
  Statisch oder dynamisch zur Laufzeit konfigurierbar

Die Software-Grundfunktionalitäten (vgl. Anhang E) des Kommunikationstransfers über das Interface, werden für das FPGA im VHDL-Template und für den µController über das Projekt-Template zur Verfügung gestellt [CD-ROM]. Die Pin-Belegungen der Hardware-Komponenten und sämtlicher Connectoren sind im Anhang A beschrieben.

3. FPGA und µController der NEXSELv2-Konfiguration

Dieser Abschnitt liefert einen Überblick über die Basiskomponenten des Gesamtsystems. Die Aufgaben und Funktionsweisen aller Komponenten werden erläutert.

3.1 NEXYS2-Board

Das NEXYS2-Board [DIGI] (vgl. Abb.3.1) bietet neben dem Spartan3E™ FPGA [10] eine Reihe von Peripherien, wie externen Speicher oder I/O-Schnittstellen, mit denen die Integration des FPGA in ein Gesamtsystem erleichtert wird. Die Aufteilung des NEXYS2-Boards ist nachfolgend beschrieben.

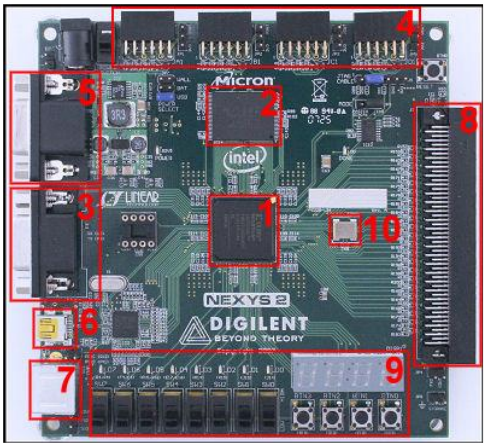


Abb. 3.1: NEXYS2-Board [XILINX]

3.1.1 I/O Interfaces und Pmods

[2] Als externes RAM ist auf dem NEXYS2-Board ein 128 MBit Micron pseudostatisches DRAM integriert, das, wie ein typisches SRAM, asynchrone Lese- und Schreibvorgänge in 70ns. durchführt. Daneben verfügt das NEXYS2-Board über ein 128MBit Intel StrataFlash ROM das Lesevorgänge innerhalb von 110ns und Schreibvorgänge in 70ns unterstützt.

[3] Eine serielle Schnittstelle ist in Verbindung mit einem Spannungswandler integriert, wobei dieser die -12V bis 12V RS-232 Pegel auf die 3,3V Betriebsspannung des FPGA umsetzt.

[**4**] Das NEXYS2-Board verfügt über 4 Pmod-Schnittstellen die direkt mit den I/O-Ports des FPGA verbunden sind. Ein Pmod besteht aus zwölf Pins, wobei acht Daten-, zwei Masse- und zwei Spannungsversorgungspins.

[**5**] Die VGA-Schnittstelle Farbinformationen und Standard für eine Auflösung von sie kann dabei sowohl mit CRT werden**.**

[**6**] Über eine PS2-Schnittstelle kann eine Maus oder eine Tastatur betrieben werden.

[**7**] Neben der USB2-Schnittstelle, die auch zum Download von und ins FPGA benutzt werden kann, besteht die Möglichkeit die Spannungsversorgung über eine Netzteilbuchse oder eine Batterie zu beziehen**.**

[**8**] Über ein FX-2-Kabel kann externe Peripherie angeschlossen werden, die in Taktraten von bis zu 100MHz laufen.

[**9**] Außerdem werden verschiedene I/O-Schnittstellen bereitgestellt; neben acht LEDs und einer vierstelligen Sieben Drucktasten und acht Schiebeschalter.

[**10**] Es verwendet einen 50 MHz Oszillator und hält einen Socket für einen zusätzlichen Oszillator bereit.

3.1.2 Hardwarekomponenten des FPGAs XC3S1200E

[**1**] Das XC3S1200E FPGA besteht aus fünf programmierbaren Elementen:

- Configuarble Logical Blocks (CLBs), diese enthalten flexible Look-Up Tabellen, welche sowohl als logische Einheit als auch zur Datenspeicherung
- Input/Output Blocks (IOBs), die die bidirektionale Verbindung von der internen Logik zu den I/O-Pins herstellen und 3-State Operationen unterstützen.
- Block RAM, zur Datenspeicherung.
- Multiplizierer, die zu zwei binären 18-Bit Zahlen das Produkt liefern.
- Digital Clock Manager Blocks (DCM), mit deren Hilfe Taktsignale verzögert, multipliziert, dividiert und verschoben werden können.
- Die Anordnung auf dem FPGA ist (siehe Kapitel. 3.1.2) so organisiert, dass die CLBs von den IOBs ringförmig umgeben sind.

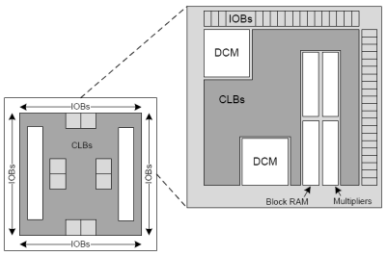


Abb. 3.2: Aufbau des FPGA [XILINX]

3.2 LPC-2468-Stick

Der LPC-2468-Stick [HITEX] ist ein kleines, modulares Evaluation Kit mit optionalen Erweiterungskarten. Die Leistung des USB-Sticks basiert auf dem ARM7™ TDMI-S LPC2468FET208 [NXP, 5] µController. Der LPC-2468-Stick bietet Zielhardware mit externem SRAM, User-Pins und LEDs für Anwendungen zu nutzen. Kombiniert mit dem bewährten USB-Debugger und die HiTOP53 Entwicklungs-Tools, erlaubt der Der LPC-2468-Stick vollen Zugriff auf alle Funktionen wie, Debugging und Programmierung. Zugang zu den externen Peripherien erfolgt über die PCM80-Schnittstelle des LPC2468-Stick.

3.2.1 Komponentenkonzept

Der Funktionsumfang an Interfaces des LPC-2468-Stick lässt sich über die Development-Boards LPC-COM und LPC-PROTO (siehe Anhang C.2) erweitert werden.

3.2.2 Peripherie des µControllers LPC2468FET208

Eine Übersicht aller verfügbaren Pin-Funktionen des LPC-2468-Sticks, über die PCM80-Schnittstelle, sind im Anhang A.3 beschrieben.

Pin-Connect-Block

Auf alle I/O-Pins des LPC-2468-Sticks sind jeweils vier interne Funktionen über einen Multiplexer angeschlossen. Diese können mit der PINSEL-Funktion [6] aktiviert werden. Die Standard-Konfiguration der Pins ist GPIO.

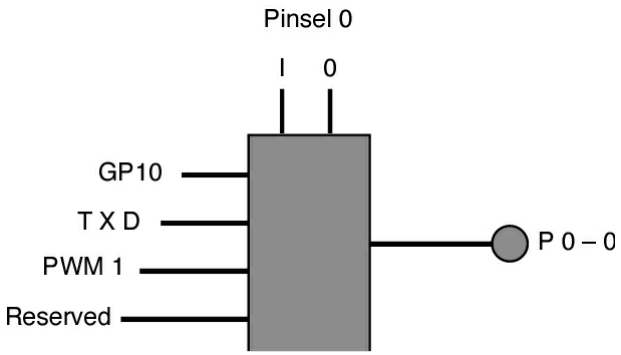


Abb. 3.3: LPC-2468-Stick Pin-Connect-Block

GPIO und FastGPIO

Der LPC-2468-Stick hat bis zu fünf Funktions-I/O-Ports, von diesen sind 3 Ports P0..2 (vgl. Anhang A.3) über den PCM80-Stecker des LPC-2468-Sticks zur Verfügung gestellt. Die jeweils 32 I/O-Leitungen ergeben so ein Maximum von 160 Pins (66 beschaltbare Pins). Desweiteren ist eine zweite Gruppe von GPIO-Steuerregistern auf dem lokalen AHB-Bus, die sogenannten FastGPIO-Steuerregister. Darüber hinaus haben PORT0 und PORT2 je einen Interrupt-Eingang Die Aktivierung kann mit steigender oder fallender Flanke am Pin codiert werden.

In den früheren LPC-Versionen [5] wurden die GPIO-Control-Register auf den APB-Bus zusammen mit allen anderen Peripherie-Modulen geschaltet. Allerdings betrug die Adressierung dieser Register eine große Anzahl von Taktzyklen für die Datenübergabe vom AHB- auf den APB-Bus. Das Umschalten eines Port-Pins dauerte 14 Taktzyklen, bei einer Oszillatorfrequenz von 60MHz, somit war die maximale Toggle-Frequenz eines Port-Pins 4,3MHz. Die Fast-GPIO-Register ermöglichen einen Port-Pin-Zugriff in nur zwei Taktzyklen. Dies entspricht einer Impulsfrequenz von 30MHz (vgl. Abb. 3.4). Mit dem Maskierungsregister für die Fast-GPIO-Pins, werden Bit-Änderungen von jedem Pin registriert.

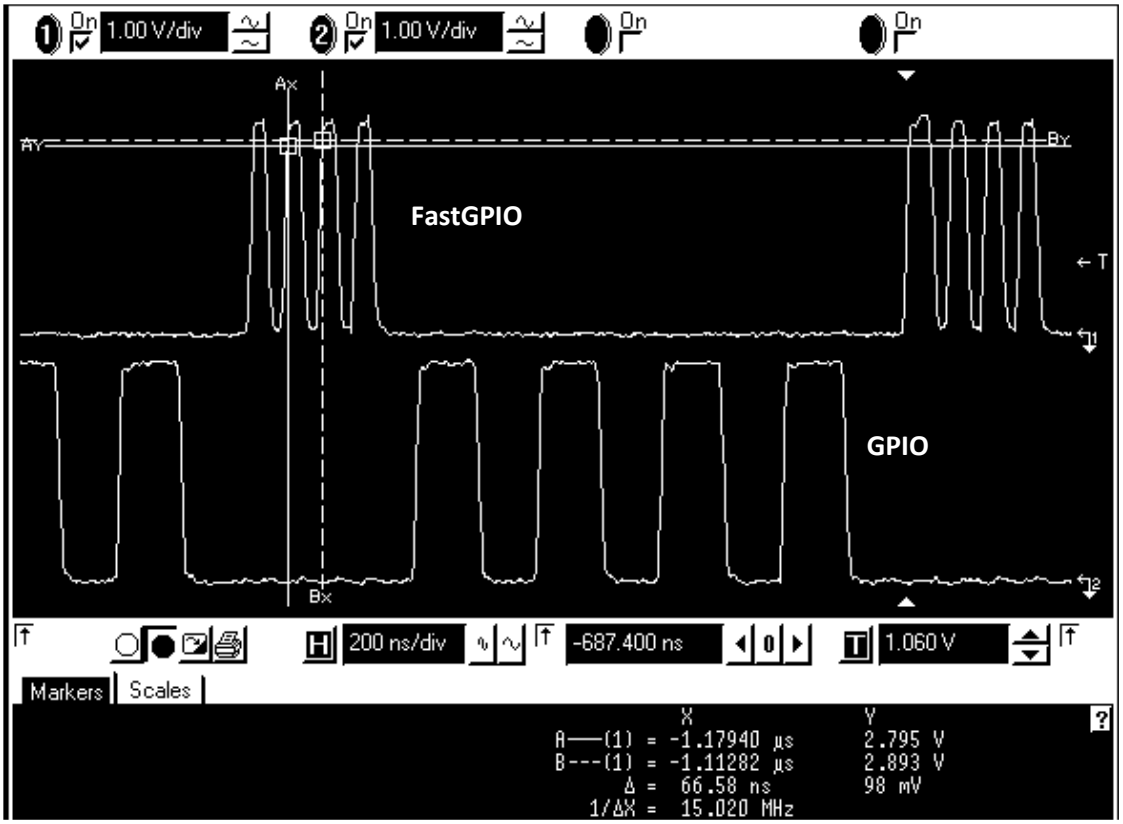


Abb. 3.4: Toggle-Frequenz von GPIO und FastGPIO [www.mikrocontroller.net]

Nach einem Systemreset des LPC-2468-Sticks, werden über den Pin-Connector-Block PCB alle Pins auf GPIO voreingestellt.

```
int main(void){
   unsigned int delay;
   unsigned int toggle = 0x00010000; // Startbit
   IODIR1 = 0x00FF0000; // Port-Pins als Output

   while(1){ // Endlosschleife
      for(delay = 0;delay<0x10000;delay++){
         ;
      }
      IOCLR1 = ~toggle; // Port-Pins auf low setzen
      IOSET1 = toggle; // Port setzen
      toggle = toggle << 1; //Bit multiplizieren mit 2
      if(toggle&0x01000000) toggle = 0x00010000; // Zurücksetzen
   }
}
```

Die FIODIR-Pins können als Eingang (0) oder Ausgang (1) konfiguriert werden. Wenn der Pin ein Ausgang ist, kann der Zustand mit den Registern FIOSET und FIOCLR gesteuert werden. Pins werden auf 0 gesetzt, wenn entsprechend eine 1 im FIOCLR-Register gesetzt wurde. Der Zustand der GPIO-Pins kann jederzeit durch das Lesen der FIOPIN-Register ermittelt werden. Das FIOMASK Register dient als Maske, um einzelne Bits der FIOSET-, FIOCLR-Register zu setzen. Über das FIOMASK-Register werden beim Schreiben einer 0, das zugehörige Bit im FIOSET- und FIOCLR-Register gesetzt. Diese Funktion beschleunigt I/O-Bit-Manipulation.

DAC

Der LPC-2468-Stick [5] hat einen 10bit Digital-Analog-Wandler. Diese Peripherie besteht aus einem einzigen Register. Der DAC ist aktiv beim Setzen der Bits 20 und 21 im PINSEL1-Register. Die umgewandelten analogen Signalwerte liegen am Pin P0.26 (AOUT) an. Es ist zu beachten, dass ein Kanal dieses Analog-Digital-Konverters den Pin mitverwendet.

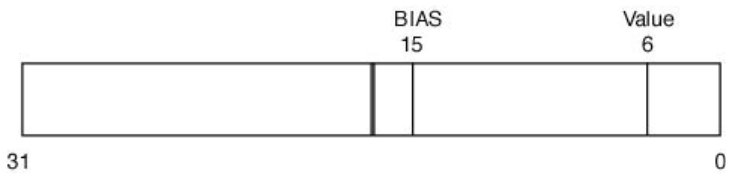


Abb. 3.5: BIAS des DAC

Die Konvertierung wird gestartet beim Schreiben in das Steuerregister. Die Umwandlungszeit ist abhängig vom gesetzten BIAS-Bit. Wenn dieses auf 1 gesetzt ist, beträgt die Zeit der Konvertierung der 2,5µs, mit einem maximalen Ausgangsstrom von 700µA. Wird das BIAS-Bit auf 0 gesetzt, ist die Umwandlung in 1µSec fertig, aber der maximale Ausgangsstrom des DACs beträgt nur 350µA. Diese Werte gelten für bei einer Kapazität von 100pF.

GPIO-Timer

Der LPC-2468-Stick hat vier Mehrzweck-Timer. Alle Timer sind identisch aufgebaut. Die Timer sind aufgebaut mit einen 32bit counter und einem 32bit prescaler. Der Timer erhält seinen Takt von der APB-Peripherie (PCLK).

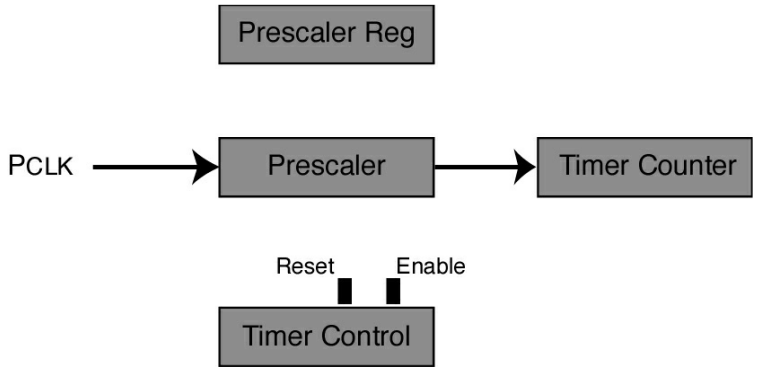


Abb. 3.6: Aufbau der GPIO-Timer

Die Taktrate des Timers wird gesteuert durch den gespeicherten Wert im Prescaler-Register. Der Prescaler inkrementiert bei jeder Taktflanke des PCLK den gespeicherten Wert im Prescaler-Register. Wenn der Skalierungsfaktor mit dem Wert des Prescaler-Registers identisch ist, wird das Counter-Register des Timers um eins erhöht und das prescaler-Register wird auf 0 gesetzt und beginnt wieder mit dem Zählvorgang. Die Timer-Control-Register werden über zwei Bits, die aktiviert oder deaktivieren und auf den Startwert zurückgesetzt (Reset).

Zusätzlich haben die Timer vier capture-Eingänge. Über diese Funktion kann der Wert eines Timer-Counter erfasst werden, wenn das Eingangssignal sich ändert.

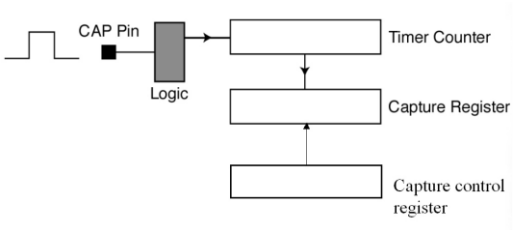


Abb. 3.7: Capture-Funktion der Timer

Das Capture-Control-Register kann so konfiguriert werden, dass bei steigender oder fallender Flanke am Eingangs-Pin des Timers ein Capture-Ereignis auslöst wird. Beim eintreten des Ereignis, wird der aktuelle Wert des Timer-Counters in das Capture-Register übertragen und ggf. ein Interrupt erzeugt. Der folgende Code veranschaulicht die Konfiguration für ein Timer mit Capture-Funktion. Wenn eine steigende Flanke an P0.2 anliegt, wird ein Interrupt ausgelöst.

```
int main(void){
   VPBDIV = 0x00000002; // PCLK auf 30MHz
   PINSEL0 = 0x00000020; // Aktivere P0.2 als capture channel0
   T0PR = 0x00007530; // Prescaler voreingestellt mit 1ms
   T0TCR = 0x00000002; // Resete counter und prescaler
   T0CCR = 0x00000005; // Capture bei steigender Flanke auf channel0
   T0TCR = 0x00000001; // Starte Timer
   VICVectAddr4 = (unsigned)T0isr; // Setzte Timer-ISR Vectoradresse
   VICVectCntl4 = 0x00000024; // Vectorchannel 4
   VICIntEnable = 0x00000010; // Aktiviere Interrupt

   while(1);
}

void T0isr (void) __irq{
  static int value;
  value = T0CR0; // Einlesen des capture Wertes
  T0IR |= 0x00000001; // Interrupt zurücksetzen über match0
  VICVectAddr = 0x00000000; // Dummy-Aufruf!! Interruptbehandlung beendet
}
```

4. Schutzschaltung der bidirektionalen Kommunikation zwischen µC und FPGA

In dieser Ausarbeitung wird die Funktion des SECURITYv2.1-Boards beschrieben. Welche den korrekten Datenstrom zwischen dem FPGA und dem µController sicherstellt. Der Schwerpunkt dieses Kapitels ist die Entwicklung einer Schutzschaltung für 38 I/O-Leitungen.

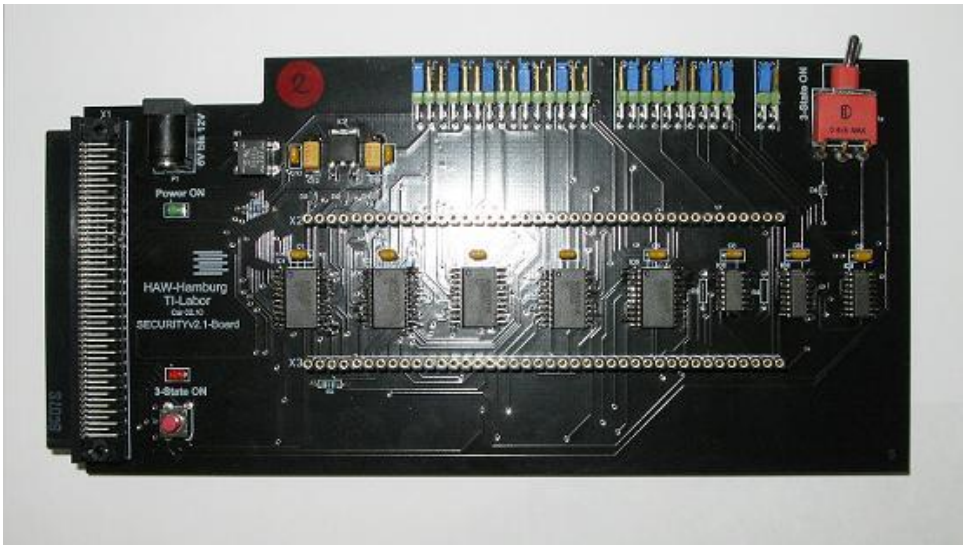


Abb. 4.1: SECURITYv2.1-Board

4.1 Schaltungskonzept des SECURITYv2.1-Boards

Der Transfer diskreter Steuer- und Datensignale (vgl. Abb. 4.2) zwischen µController und dem FPGA erfolgt über die Schutzschaltung des SECURITYv2.1-Boards (vgl. Kapitel B.2). Hierfür regelt der µController, dynamisch zur Laufzeit sequenziell vor jeder bidirektionalen Übertragung eines Datums die Transferrichtung (s. Kap. 4.4), beider Bustreiber IOB1 und IOB2, im GPIO- oder FastGPIO-Modus, über zwei separate Steuerleitungen. Die Signalflussrichtung IOB3..5 (Steuer- und Adresssignale) sind statisch über Jumper (vgl. Abb. 4.2) voreingestellt.

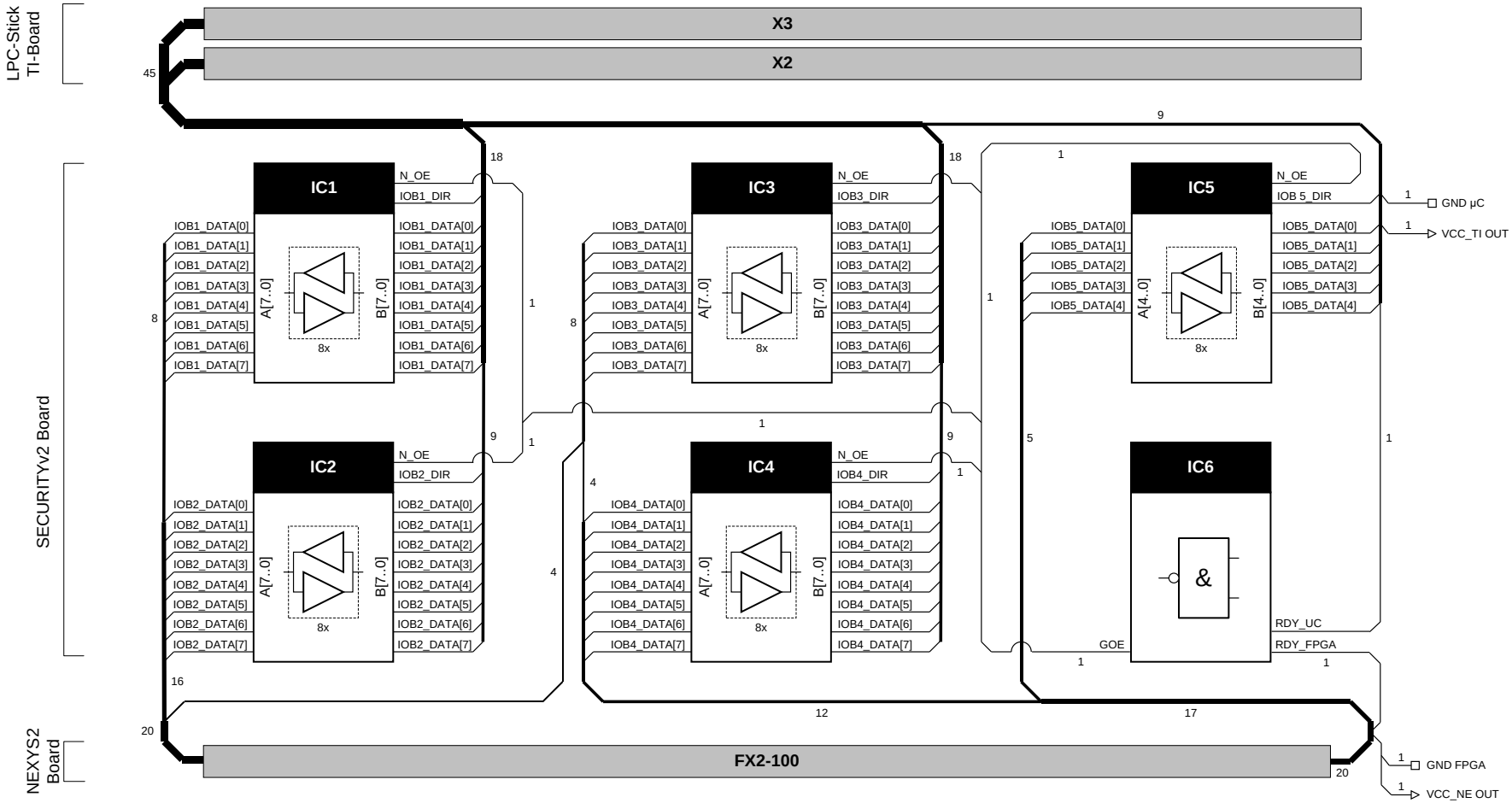


Abb. 4.2: Blockschaltbild des SECURITYv2.1-Boards

Alle Bustreiber können mit Jumpern des SECURITYv2.1-Boards statisch oder durch Signalzuweisungen der Steuersignale während des Betriebs konfiguriert werden (vgl. Kapitel 4.4). Für die Kommunikation und sequentielle oder automatisierte Datenübertragung zwischen µController und FPGA sind im Anhang E die Software-Routinen (C-Funktionen) beschrieben.

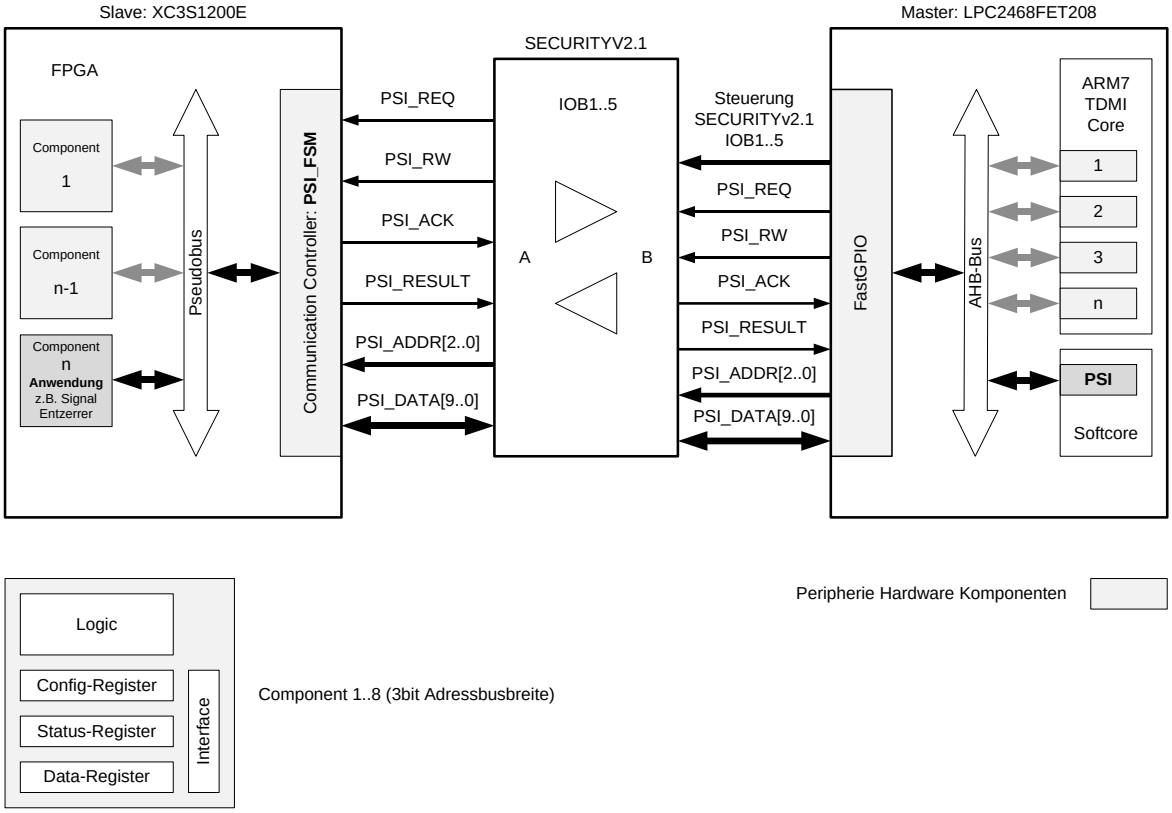


Abb. 4.3: Kommunikation über PSI-Schnittstellenkopplung

Die Schutzschaltung trennt die Development-Board-Komponenten FPGA und µController, wenn nachfolgende Zustände eintreten.

- Eines des Ready-Signale (RDY_FPGA und RDY_UC) ist nicht auf 1 geschaltet
- Bootvorgang vom FPGA oder µController
- Reset des NEXYS2- oder TI-LPC-Stick-Boards
- Beim drücken des Tasters (vgl. Abb. 4.4)
- Schalterstellung auf 3-State

Wenn der Schalter auf 3-State gelegt wurde, können die Development-Boards NEXYS2 und TI-LPC-Stick stand-alone betrieben werden.

4.2 Funktionsübersicht und Partitionierung der BUS-Treiber

Die Bustreiberstufen des SECURITYv2.1-Boards können durch die entsprechende Jumper-Konfiguration in zwei Partitionen, wie nachfolgend beschrieben, betrieben werden.

Tab. 4.1: Steuersignale für das SECURITYv2.1-Board

Daten-/Steuersignale				Development-Boards						
				TI-LPC	SECURITYv2.1			NEXYS2		
Name	Typ	Anmerkung	Transfer	µC Port[Pin]	Bustreiber	Connector Typ	Connector Pin	FPGA Signal	FPGA Pin	Connector FX2-100 Port-Pin
PSI_REQ	S	Auftrag	unidirektional	P1[12]	IOB4	X2	24	IOB4<3>	F11	J1A-34
PSI_ACK	S	Auftragsbestätigung	unidirektional	P1[8]	IOB5	X2	25	IOB5<2>	C14	J1A-41
PSI_RW	S	Schreiben/Lesen	unidirektional	P1[11]	IOB4	X2	23	IOB4<3>	F11	J1A-34
PSI_RESULT	S	Auftragsfertigstellung	unidirektional	P2[12]	IOB5	X3	9	IOB5<1>	A14	J1A-40
PSI_ADDR[0]	S	Adresse des Prozessorelements	unidirektional	P1[15]	IOB4	X2	33	IOB4<0>	B11	JA1-31
PSI_ADDR[1]				P1[14]			34	IOB4<1>	C11	JA1-32
PSI_ADDR[2]				P1[13]			29	IOB4<2>	E11	JA1-33
PSI_DATA[0]	D	Datum	bidirektional	P0[5]	IOB1	X3	36	IOB1<0>	A4	J1A-07
PSI_DATA[1]				P1[10]		X2	26	IOB1<1>	C3	J1A-08
PSI_DATA[2]				P0[13]			5	IOB1<2>	C4	J1A-09
PSI_DATA[3]				P0[14]			7	IOB1<3>	B6	J1A-10
PSI_DATA[4]				P0[19]		X3	15	IOB1<4>	D5	J1A-11
PSI_DATA[5]				P0[20]			14	IOB1<5>	C5	J1A-12
PSI_DATA[6]				P0[21]		X2	19	IOB1<6>	F7	J1A-13
PSI_DATA[7]				P0[22]			18	IOB1<7>	E7	J1A-14
PSI_DATA[8]				P0[29]	IOB2		6	IOB2<0>	A6	J1A-15
PSI_DATA[9]				P0[30]			15	IOB2<1>	C7	J1A-16
IOB1_DIR	S	Transferrichtung der 1. IOB-Komponente für PSI_DATA[7..0]	unidirektional	P2[3]	IOB1	X3	32			
IOB2_DIR	S	Transferrichtung der 2. IOB-Komponente für PSI_DATA[9..8	unidirektional	P2[4]	IOB2	X3	33			

D : Datensignal , S : Steuersignal

Partion 1:

In dieser Konfiguration werden beide Bustreiber IOB1 und IOB2 über den Jumper J11 (vgl. Abb. 4.4) gekoppelt. Die Datentransferrichtung wird mit dem IOB1-Signal gesteuert. Somit können 16bit bidirektional gesteuert werden.

Bustreiber	DIR-Signal	Datenbreite	Jumper J11
IOB1+IOB2	IOB1_DIR	16bit	Pins 1-2
IOB3	IOB3_DIR	8bit	
IOB4	IOB4_DIR	8bit	
IOB5	IOB5_DIR	5bit	

Partion 2:

Die Datentransferrichtung der Bustreiber IOB1..5 werden in der Standardkonfiguration (vgl. Abb. 4.4) über das jeweilige IOB1..5_DIR-Signal gesteuert.

Bustreiber	**DIR-Signal**	**Datenbreite**	**Jumper J11**
IOB1	IOB1_DIR	8bit	Pins 2-3
IOB2	IOB2_DIR	8bit	
IOB3	IOB3_DIR	8bit	
IOB4	IOB4_DIR	8bit	
IOB5	IOB5_DIR	5bit	

4.3 Steuerung der Schutzschaltung

Die Schutzschaltung des SECURTIYv2.1-Boards zweiteilig aufgebaut. Einem NAND-Gatter und fünf Multiplexer. Auf das NAND-Gatter-Eingänge sind die beiden Ready-Signale RDY_FPGA und RDY_UC geschaltet. Der Ausgang des NAND-Gatters wird über den Hauptschalter S1 auf die Multiplexer-Eingänge 1 (vgl Abb. 4.4) geschaltet. Die Ausgänge der Multiplexer sind mit den low-aktiven Enable-Eingängen der Bustreiberstufen verbunden. Wird ein Ready-Signal auf 0 durch das FPGA oder dem µController gesetzt, der Taster T1 betätigt oder der Schalter auf 3-State geschaltet, liegen an den Multiplexer-Ausgängen ein high Signal an, welche die Bustreiberstufen in den 3-State-Zustand schaltet. Auf den Select-Eingängen der Multiplexer liegt die Netzversorgungsspannung VCC mit 3V3 des Development-Boards NEXYS2. Die Multiplexer-Eingänge 0 sind mit der Netzversorgungsspannung des TI-LPC-Stick-Board verbunden. Fällt die Netzversorgung des NEXS2-Boards ab, schaltet der Multiplexer auf den Eingang 0 und legt VCC des TI-LPC-Stick-Boards auf die Enable-Eingängen der Bustreiberstufen. Diese schalten somit in den 3-State-Zustand.

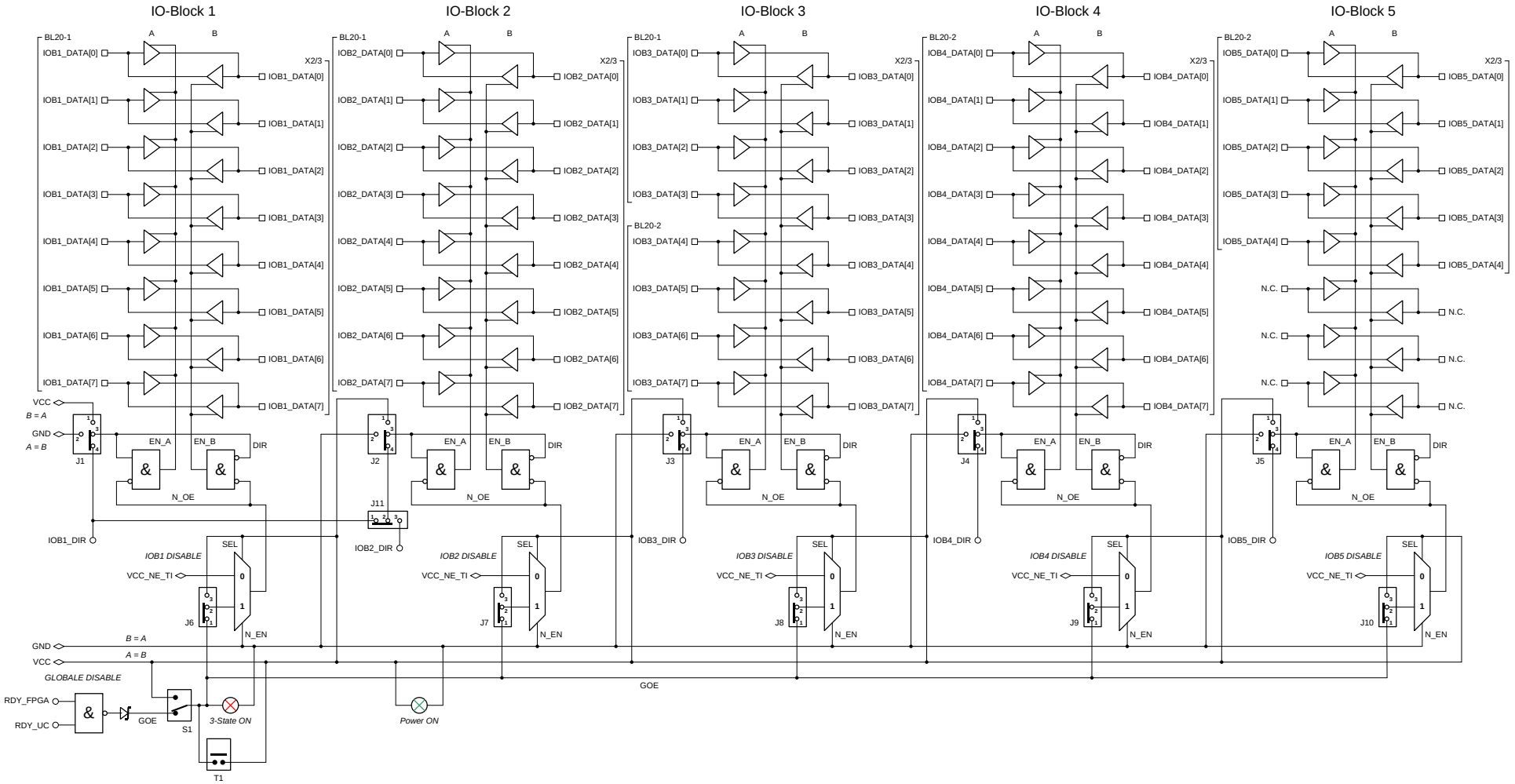


Abb. 4.4: Schaltlogik des SECURITYv2.1-Boards

4.4 Parametrisierung

Mit der Parametrisierung über die Jumper (vgl. Abb. 4.4 und B.14), können die Bustreiberstufen separat in die Zustände 3-State oder Betrieb geschaltet werden. Desweitern kann die Datentransferrichtung statisch über Jumper oder dynamisch zur Laufzeit konfiguriert werden.

4.4.1 Statisch mit Jumper

Bustreiberstufen in den Betriebszustand schalten:

Bustreiber	Jumper	Jumper-Pins
IOB1	J6	1-2
IOB2	J7	1-2
IOB3	J8	1-2
IOB4	J9	1-2
IOB5	J10	1-2

Bustreiberstufen in 3-State schalten:

Bustreiber	Jumper	Jumper-Pins
IOB1	J6	2-3
IOB2	J7	2-3
IOB3	J8	2-3
IOB4	J9	2-3
IOB5	J10	2-3

Datentransferrichtung der Bustreiberstufen vom FPGA zum µController:

Bustreiber	Jumper	Jumper-Pins
IOB1	J1	1-3 (VCC)
IOB2	J2	1-3 (VCC)
IOB3	J3	1-3 (VCC)
IOB4	J4	1-3 (VCC)
IOB5	J5	1-3 (VCC)

Datentransferrichtung der Bustreiberstufen vom µController zum FPGA:

Bustreiber	Jumper	Jumper-Pins
IOB1	J1	2-3 (GND)
IOB2	J2	2-3 (GND)
IOB3	J3	2-3 (GND)
IOB4	J4	2-3 (GND)
IOB5	J5	2-3 (GND)

4.4.2 Dynamisch zur Laufzeit

Datentransferrichtung der Bustreiberstufen dynamisch schalten:

Bustreiber	**Jumper**	**Jumper-Pins**
IOB1	J1	3-4
IOB2	J2	3-4
IOB3	J3	3-4
IOB4	J4	3-4
IOB5	J5	3-4

5. Asynchrones Kommunikationsinterface der SoC-Plattform

In dieser Ausarbeitung wird ein Ansatz zur Kommunikation zwischen dem FPGA und dem µController beschrieben. Der Schwerpunkt dieses Kapitels ist die Modellierung der PSI (Parallel Software Interface) Komponenten, zum 10bit parallelen asynchronen Datentransfer, als Software-Sequenz des µControllers und als Prozessorelement. Es werden sowohl die Entwicklung behandelt, als auch die Synthese-Ergebnisse und Timing-Simulation (s. Kapitel 5.4) des FPGAs analysiert. Die Zielsetzung der Module ist, die maximale Umschaltfrequenz von 3,7 MHz der FastGPIO-Ports (vgl. Anhang H), bei 72MHz Oszillatorfrequenz des µControllers, für die höchste Performanz der NEXSELv2-Plattform zu verwenden. Das Prozessorelement arbeitet in der Konstellation dieser Master-Slave-Konfiguration mit einer Taktfrequenz von 50MHz.

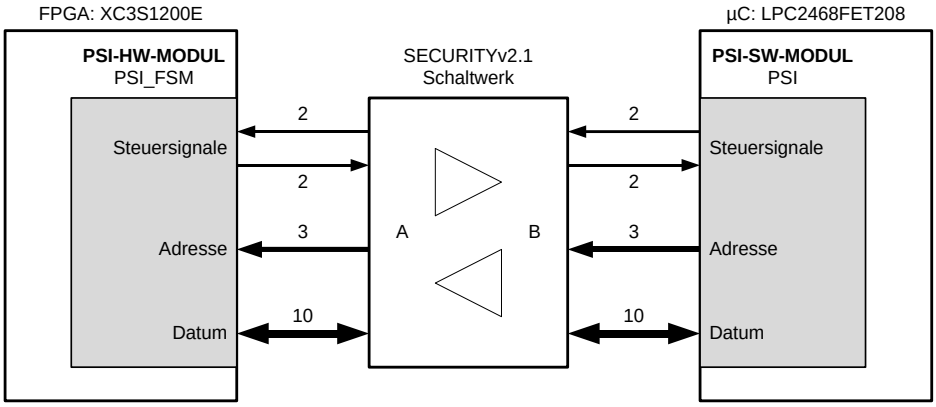


Abb. 5 1: PSI-Hardware/Software Module

Um die nebenläufige Flexibilität der Komponenten mit unterschiedlichen Taktraten sicherzustellen wird kein gemeinsames Taktsignal vom µController zum FPGA zur Synchronisation geschaltet. Demnach muss das PSI-Protokoll den zuverlässigen Datenaustausch gewährleisten. Hierzu wird unter Kapitel 5.1 das 4-Phasen-Handshake funktional beschrieben für asynchrone Master-Slave-Anwendung. Sowohl die Software-Sequenz als auch der Hardware-Automat implementieren dieses Übertragungsverfahren. Über das Doppel-Flip-Flop-Konzept [38] in der Synchronisationsstufe des Prozessorelements werden meta-stabile Signalzustände, bei den kongruenten Taktfrequenzen des µControllers und FPGAs, auf den Steuer- und Datenleitungen verhindert. Zur Reduzierung der diskreten Signalleitungen und I/O-Ports, werden die Nutzdaten bidirektional übertragen. Der Kommunikationstransfer über die Kontrollsignale (vgl. Abb. 4.2) zwischen den Modulen wird vom µController als Mastereinheit gesteuert und kontrolliert.

5.1 Software-Modulkonzept PSI

Wie in der Einführung (vgl. Kapitel 5) beschrieben, sind die SoC-µC-Komponenten asynchron gekoppelt. Das als Busprotokoll 4-Phasen-Handshake (vgl. Abb. 5.3) übernimmt, in dieser Hardware-Konstellation, die Sicherstellung des korrekten Datentransfers zwischen der Mastereinheit und dem Hardware-Beschleuniger. Die Kommunikation über das PSI, ist als Master-Slave-Verbindung realisiert. Jegliche Datenübertragungen (vgl. Abb. 3) werden vom Master initiiert. Hierzu werden das PSI_REQ-Signal (REQuest = ENQuiry) vom µController zum FPGA und das PSI_ACK-Signal (ACKnowledge) vom Slave zum Master übertragen.

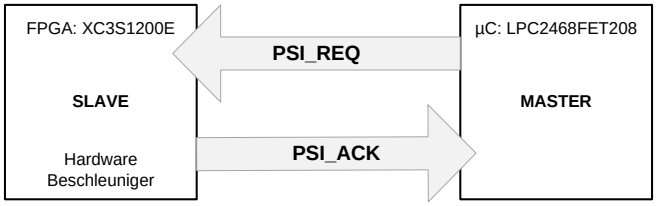


Abb. 5.2: Synchronisationssignale des SoC-µC-Interfaces

Die Daten- und Steuersignale sind über die Bibliotheksfunktionen (vgl. Anhang E) zu steuern.

5.1.1 Protokoll- und Transferstruktur: 4-Phasen-Handshake

Die Bus-Spezifikation sieht einen zentralen Master vor, der die Koordination der angeschlossenen Peripherie-Hardware übernimmt. Über die vier Transferphasen (vgl. Abb. 5.3) des Busprotokolls steuert der µController den Kommunikationsablauf. Für die Signalflussrichtungen der Steuer- und Datensignale, sowie die Busankopplung des Datums zur Laufzeit.

1. **Kommunikationsaufbau und Auftragserteilung**:
   Der Master fordert durch das Setzen des PSI_REQ-Signals eine Datenübertragung an.
2. **Auftragsannahme**:
   Mit dem Übergang des PSI_ACK-Signals auf High quittiert der Slave die Annahme des Datentransferauftrags.
3. **Auftragsbestätigung**:
   Als Reaktion auf die Kommunikationsbereitschaft des Hardware-Beschleunigers, setzt der µController das PSI_REQ-Signal zurück.

4. **Auftragsabschluss und Kommunikationsabbau**:
   Über den Pegelwechsel des Steuersignals PSI_ACK von High auf Low wird das Transferprotokoll durch das FPGA beendet.

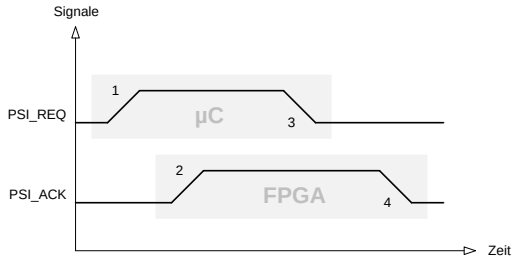


Abb. 5.3: 4-Phasen-Handshake (PSI)

5.1.2 PSI-Steuerungsmodi

Der Schreib- oder Lesemodus schaltet die Datentransferrichtung vom µController zum FPGA oder entgegengesetzt. Desweiteren werden die 3-State-Treiber des Prozessorelements und die Bustreiber des SECURITYv2.1-Boards gesteuert. Mit der Parametrisierung des PSI_RW-Signals (vgl. Tab. 5.1) kontrolliert die Mastereinheit, in Abhängigkeit der geforderten Transferrichtung des Datums PSI_DATA[9..0], die Datenkommunikation. Unverändert, durch die Voreinstellung des Steuersignals PSI_RW, ist der Kommunikationsablauf des 4-Phasen-Handshakes. Über die zwei implementierten Übertragungsmodi (s. Kapitel 5.1.3) des PSI, steuert der µController die 3-State-Treiber der bidirektionalen I/O-Eingänge des Hardware-Beschleunigers und die ersten beiden Bustreiber (IOB1 und IOB2) des SECURITYv2.1-Boards-Interfaces mit den Steuerleitungen IOB1_DIR und IOB2_DIR (vgl. Abb. 5.4 und 5.5). In Abhängigkeit der geschalteten Transferrichtung wird das Datum über das 4-Phasen-Handshake übermittelt.

Tab. 5.1: PSI_RW Hard- und Software-Funktionalität

Software-Einstellungen Signalparameter					**Interface-Steuerung** SECURITYv2.1			
Name	Funktion	Anmerkung	µC Port[Pin]	Pegel	Funktion	Komponente	µC Port[Pin]	Pegel
PSI_RW	Datentransferrichtung für 4-Phasen-Handshake. Steuerung über die C-Routine.: setPSIReadWrite() (vgl. Anhang C)	Schreiben 1	P1[11]	low	Richtung: von B nach A	IOB1	P2[3]	low
						IOB2	P2[4]	
		Lesen 2		high	Richtung: von A nach B	IOB1	P2[3]	high
						IOB2	P2[4]	

Richtungen: A = NEXYS2-Board und B = TI-Board LPC-Stick.

1. **Schreibemodus vom µController zum FPGA**
   Für den schreibenden Zugriff des µControllers, auf das 10bit Synchronisationsregister des FPGAs, schaltet dieser, durch Aufruf von setPSIReadWrite(IOB_DIR_A_2_B), die Datentransferrichtung der 3-State-Treiber über die PSI_FSM (s. Kapitel 5.2) mit dem Setzen PSI_RW-Signals und legt parallel die High-Pegel auf die Steuerleitungen IOB1..2_DIR (vgl. Abb. 5.4).

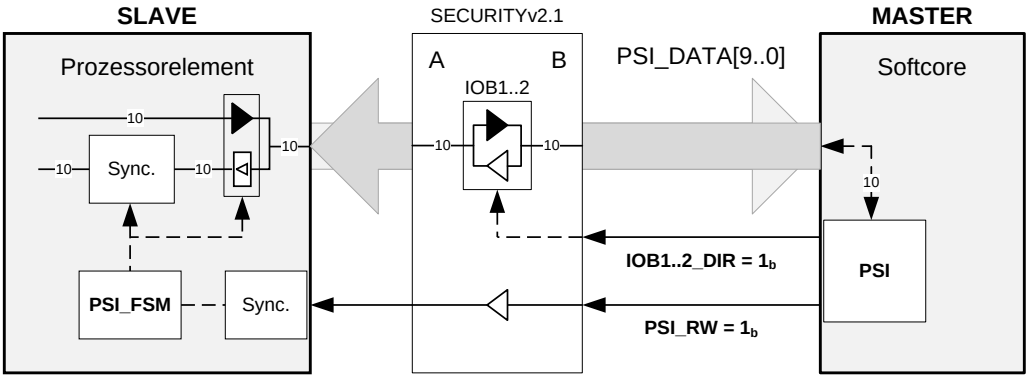


Abb. 5.4: Steuerung der Datenübertragung von B nach A

2. **Lesemodus vom FPGA zum µController**
   Dieser Kommunikationsmodus (vgl. Abb. 5.5) arbeitet konträr zum Schreibmodus und initiiert durch den Master den Kommunikationsfluss des Datums, vom FPGA zum µControllers, über das Rücksetzen der drei Kontrollsignale PSI_RW, IOB1_DIR und IOB2_DIR. Durch den parametrisierten C-Funktion-Aufruf mit der Konstante IOB_DIR_B_2_A werden die Pegel auf die Steuerleitungen gelegt.

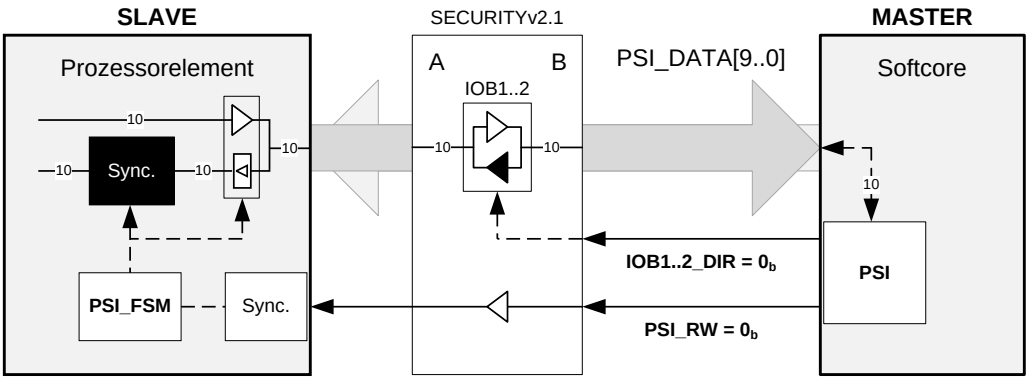


Abb. 5.5: : Steuerung der Datenübertragung von A nach B

5.1.3 PSI-Kommunikationsmodi

Ein Datentransferzyklus umfasst den Kommunikationstransfer eines 10bit Datums, über das 4-Phasen-Handshake, vom µController zum FPGA oder umgekehrt. Innerhalb eines Datentransfers (vgl. Abb. 5.6) kann nur ein Übertragungsmodi geschaltet werden. Demzufolge sind, für eine bidirektionale Datenübermittlung (Schreib- und Lesezugriff) des µControllers, zwei separate 4-Phasen-Handshake-Kommunikationstransfers sequentiell durchzuführen. Nach der Initialisierung der Kommunikation, in Abhängigkeit der Transferrichtung (vgl. Abb. 5.4 und 5.5), durch die Mastereinheit, wird die Übertragung gestartet.

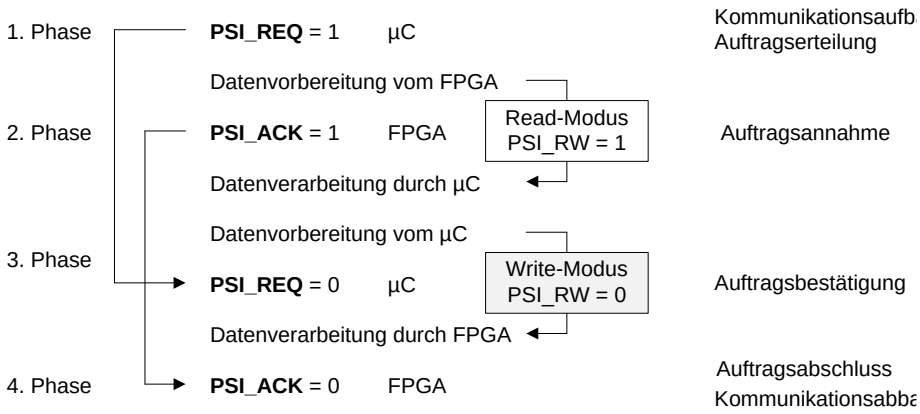


Abb. 5.6: Übertragungsmodi des 4-Phasen-Handshakes

Schreibemodus:

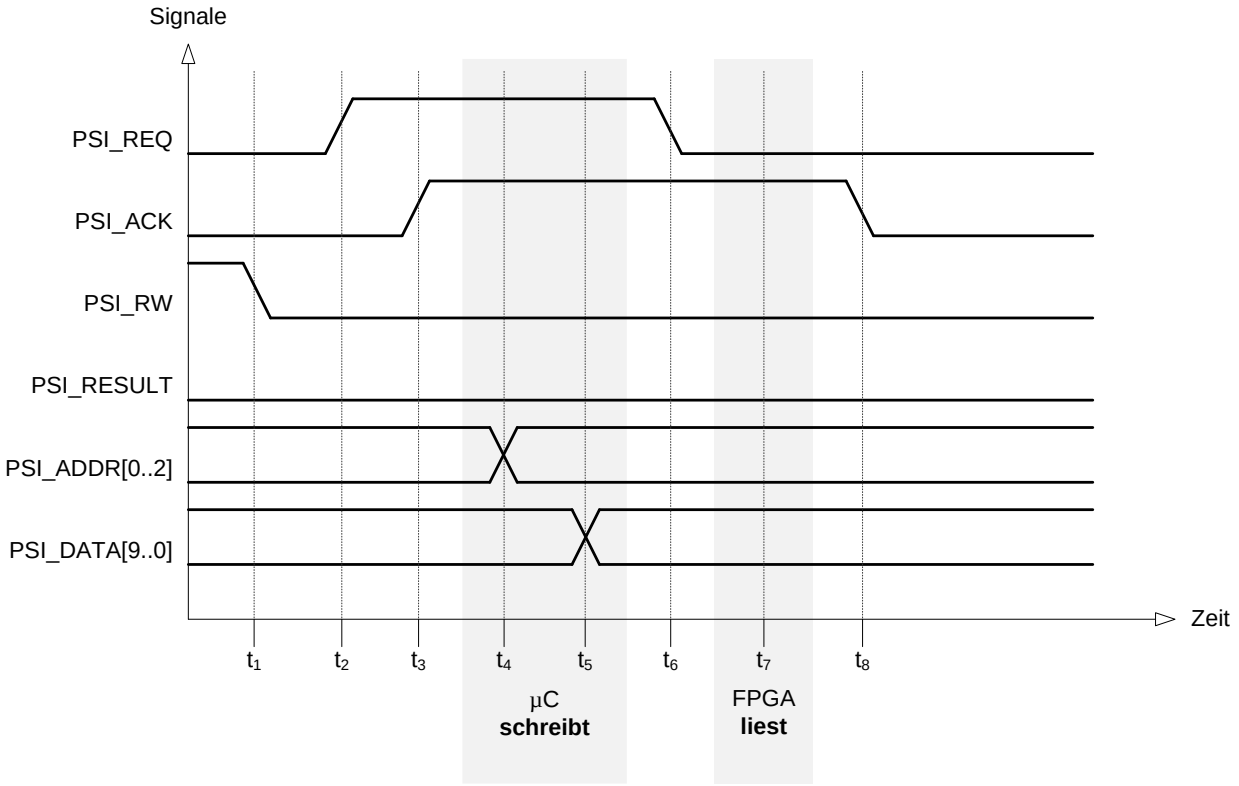


Abb. 5.7: 4-Phasen-Handshake – schreibender Zugriff des µControllers

Timing-Ablauf des 4-Phasen-Handshake im Schreibmodus:

t_1 : Der µController signalisiert dem FPGA durch das Rücksetzen des PSI_RW-Signals den Lesemodus und setzt die Transferrichtungen der Bustreiber IOB1..2 auf B→A.
t_2 : Das PSI_REQ-Signal wird vom Master auf High-Pegel gesetzt und startet den Kommunikationstransfer.
t_3 : Durch den High-Pegel des PSI-ACK-Signals quittiert der Hardware-Beschleuniger den Auftrag und signalisiert dem Master die Empfangsbereitschaft.
t_4 : Die 3bit Komponentenadresse – wenn mehrere Prozessorelemente vorhanden – wird vom Master zur Laufzeit konfiguriert.
t_5 : Der µController schreibt das 10bit Datum auf die GPIO-Leitungen.
t_6 : Durch die Pegeländerung des PSI_REQ-Signals auf Low, signalisiert der Master dem Slave die Verfügbarkeit des Datums an.
t_7 : Die PSI_FSM schaltet die Enable-Signale für die 3-State-Treiber der Prozessorkomponente und das Dateneingangsregister und taktet nebenläufig das Datum PSI_DATA[9..0] ein.
t_8 : Das Rücksetzen des PSI_ACK-Signals zeigt die abgeschlossene Datenaufnahme des FPGAs an, Der Kommunikationstransfer ist abgeschlossen.

Lesemodus:

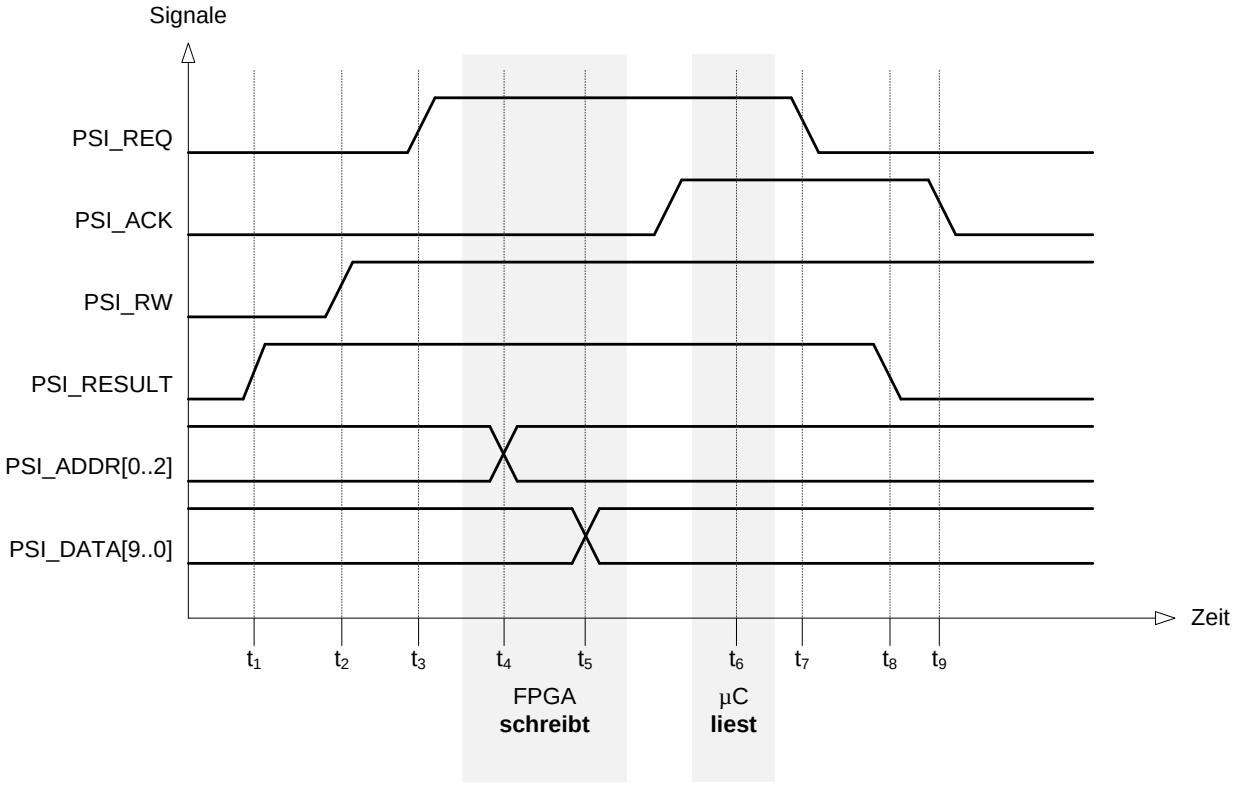


Abb. 5.8: 4-Phasen-Handshake – lesender Zugriff des µControllers

Timing-Ablauf des 4-Phasen-Handshake im Lesemodus:

t_1 : Der Hardware-Beschleuniger signalisiert der Mastereinheit über den High-Pegel des PSI_RESULT-Signals die Fertigstellung der Datenkalkulation.
t_2 : Vor dem Kommunikationsaufbau des 4-Phasen-Handshake, aktiviert der µC durch das Setzen des PSI_RW-Signals auf High den Schreibemodus und schaltet die Transferrichtung der Bustreiber IOB1+2 des SECURITYv2.1-Board auf A→B. Das Signal PSI_RW wird während eines Kommunikationszyklus, zur Reduzierung der Latenzzeit, nur einmalig gesetzt.
t_3 : Die 3bit Adresse – wenn mehrere Prozessorelemente vorhanden – wird vom Master konfiguriert. Für die Korrektheit des Datums, ist ggf. vorerst eine Rückantwort des FPGA an den µC notwendig, welche Beschleuniger-Komponente die Berechnungen abgeschlossen hat.
t_4 : Der µC signalisiert mit dem High-Pegel des PSI_REQ-Signals den Kommunikationstransfer über das 4-Phasen-Handshake.
t_5 : Durch das PSI_RW-Signal schalten die 3-State-Treiber der Prozessorkomponente auf Output und legen das 10bit Datenausgangsregister auf die Datenleitungen PSI_DATA[9..0].
t_6 : Über das Setzen des PSI_ACK-Signals auf High, bestätigt das FPGA die 2. Stufe des 4-Phasen-Handshakes (Auftragsbestätigung).
t_7 : Die Mastereinheit liest den Datenvektor PSI_DATA[9..0] ein und speichert diesen.
t_8 : Nach der Sicherung des aktuellen Datums, bestätigt der µC dem FPGA den Eingang mit dem Setzen des Low-Pegels des PSI_REQ-Signals.
t_9 : Mit dem Low-Pegel des PSI_RESULT –Signals signalisiert das FPGA das keine weiteren berechneten Daten verfügbar sind.

5.2 PSI-Komponente des FPGAs

Die PSI-Komponente synchronisiert die 1bit breiten Eingangssignale PSI_REQ, und PSI_RW. Das 3bit Eingangssignale PSI_ADDR werden nicht verwendet (vgl. Abb. 5.9), da das Prozessorelement nur eine Komponente zur Signalweiterverarbeitung implementiert. Die Ausgangssignale PSI_DATA mit 10bit, PSI_ACK und PSI_RESULT 1bit breit, werden über die Ausgangsregisterstufen auf das Hardware-Interface geschaltet. Der Steuerautomat ist eine Mealy-FSM. Diese implementiert das 4-Phasen-Handshake, steuert die empfangen Daten an Komponente zur Signalweiterverarbeitung, speichert das berechnete Datum im PSI_DATA-Ausgangsregister und schaltet die 3-State-Treiber. Ebenso signalisiert die FSM über das 1bit breite PSI_RESULT-Signal dem µController, dass die Berechnung abgeschlossen ist und das Datum vom µController empfangen werden kann.

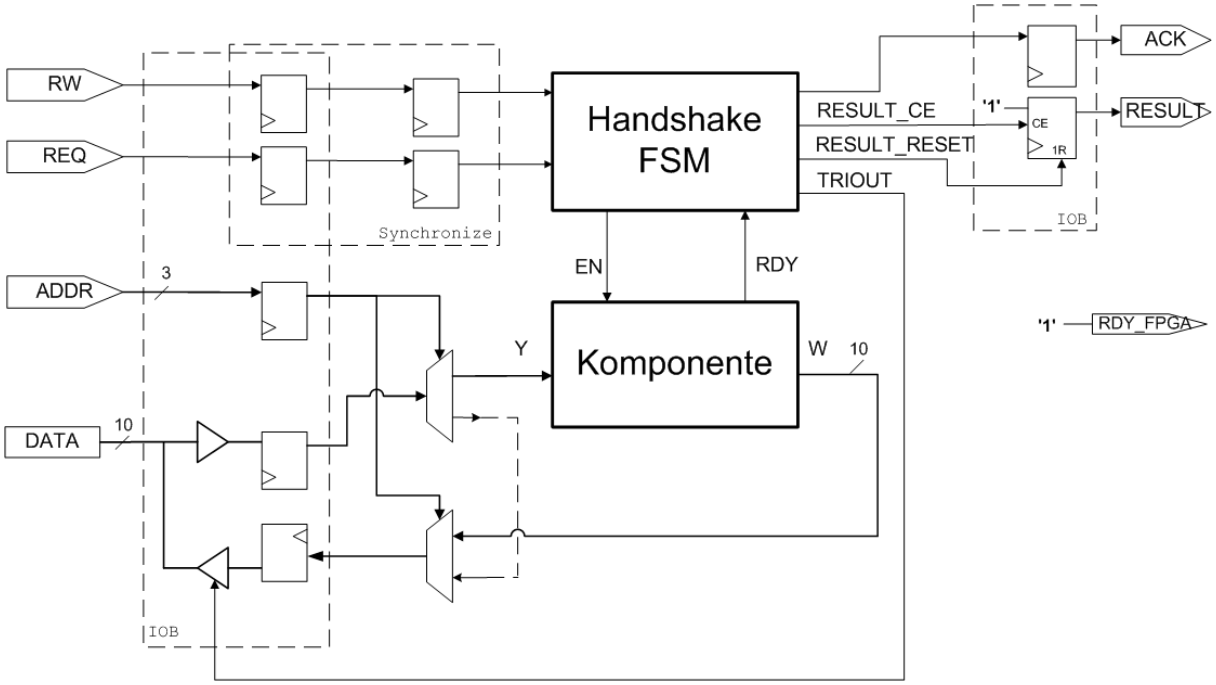


Abb. 5.9: PSI-Prozessorelement [Edres]

5.2.1 Synchronisation der Steuer- und Datensignale

Durch die asynchrone Signalkopplung, der 7 Steuer- und 10 Datenleitungen, zwischen dem µController und dem FPGA, mit den differierenden Oszillatorfrequenzen (Master f_{PCLK} = 48..72MHz und Slave f_{CLK} = 50MHz), könnten einstufige Signaleingangsregister des Prozessorelements während der Datenübertragung einen meta-stabilen Zustand erreichen, vgl. hierzu Abb. 5.10.

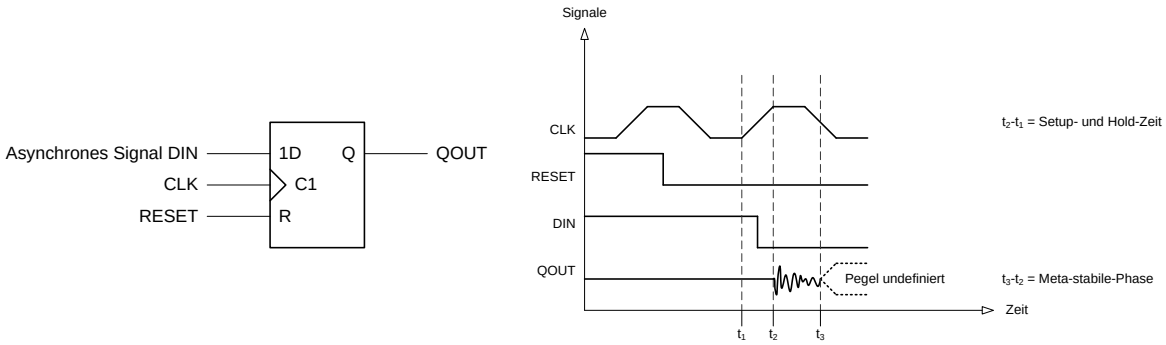


Abb. 5.10: Meta-stabiler Zustand eines Daten-Flip-Flops

Findet eine Pegeländerung am Eingang DIN eines Daten-Flip-Flops während der Setup- und Hold-Zeit statt, kann dies zu einem meta-stabilen [28] Zustand des Speicherausgangs QOUT führen. D.h. kann QOUT bis zum nächsten positiven Flankenwechsel einen Spannungswert aufweisen, der weder als logisch 0 = Low oder 1 = High - entsprechend der Low-Voltage-TTL Pegelkonvention - interpretiert wird oder einen falschen Signalpegel für QOUT liefert. Erst mit dem nächsten positiven Taktflankenwechsel, kann das Ausgangssignal einen korrekten Wert darstellen. Werden für jede im Eingangspfad der 17 Kommunikationsleitungen des Prozessorelements zwei hintereinander geschaltete Daten-Flip-Flops (Doppel-DFF-Verfahren, vgl. Abb. 5.11) implementiert [37], wird sichergestellt, dass sich die Ausgangsstufe QOUT2, des zweiten Daten-Flip-Flops, in einem stabilen Zustand befindet und ein konstanten Spannungspegel liefert.

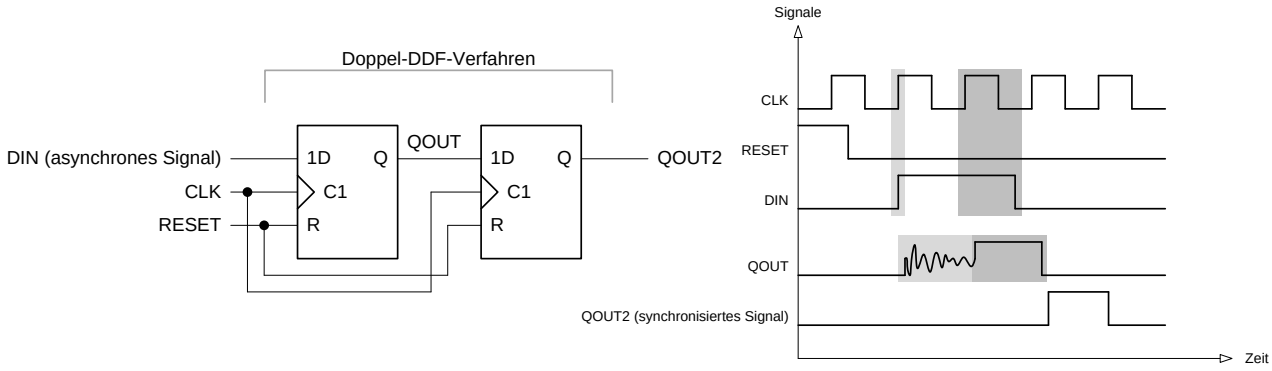


Abb. 5.11: Synchronisation von asynchronen Eingangssignalen

Mit der Synchronisierung der Eingangssignale verzögern sich diese, mit jeder Registerstufe um einen Systemtakt T_{CLK}, für die Signalverarbeitung durch die PSI_FSM.

$$T_{CLK} = \frac{1}{f_{CLK}} = \frac{1}{5 * 10^7\, Hz} = 20ns$$

Hieraus resultiert eine Latenz der Steuer- und Datensignale bei Doppel-DFF-Verfahren von $T_{Latenz} = T_{CLK} * A_{DDF}$ = 40ns. Eine höhere Stabilisierung der asynchronen Transfersignale, wird durch zwei gekoppelte Doppel-Flip-Flop-Verfahren erreicht. Entsprechend erhöht sich die Verzögerungszeit.

5.2.2 Zustandsautomat zur Steuerung des Datentransfers

Der deterministische Mealy-Automat [16] PSI_FSM arbeitet in der Koppelung mit der Software-Sequenz des µControllers im Slave-Modus. Die Kommunikationssteuerung erfolgt sequentiell vom Master (vgl. Kapitel. 5.1). Hierzu wird unter Kapitel 5.2.1 zur Sicherung korrekter Eingangspegelzustände das Synchronisationskonzept beschrieben. Die Datentransfereingänge sind von den Ein- und Ausgangsregistern über Tri-State-Treiber mit dem Pseudobus gekoppelt. Über das PSI_RW-Signal wird die Transferrichtung der Daten gesteuert. Dieses Signal wirkt direkt auf das Ausgangschaltnetz der PSI_FSM. Zur Minimierung der Verzögerungszeit, wird ein zu berechnendes Datum an das Prozessorelement übertragen und nach Beendigung der Kalkulation wird das Ergebnis an den Master übers 4-Phasen-Handshake übermittelt. Hierzu signalisiert die PSI_FSM der Software-Sequenz über den High-Pegel des PSI_RESULT-Signals die Verfügbarkeit des berechneten Datums.

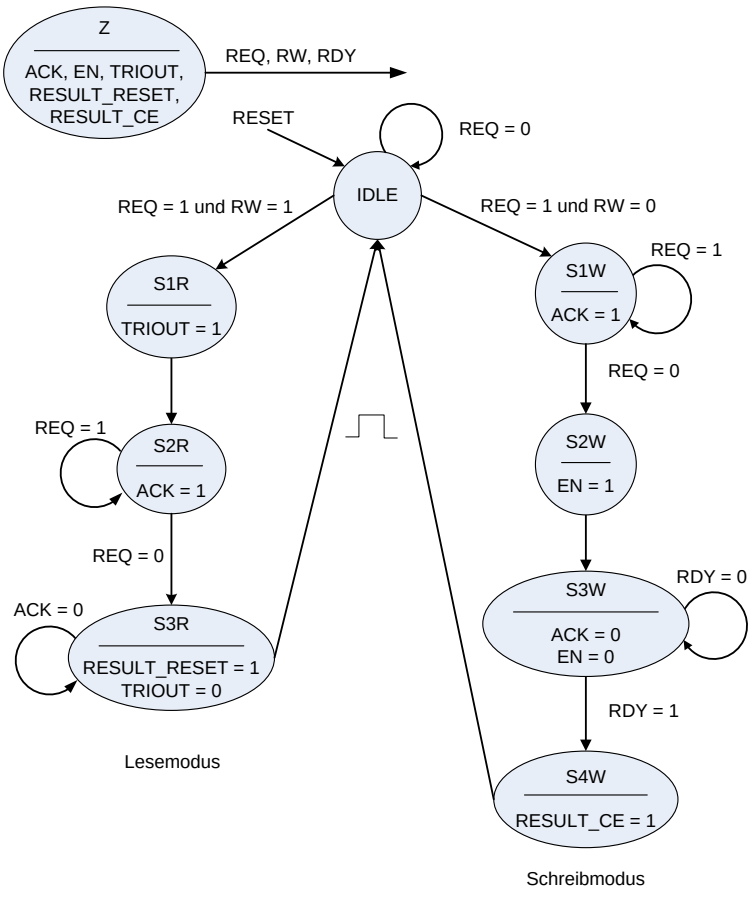


Abb. 5.12: PSI-FSM

5.3 ISR-Softwarekonzept des µControllers

Der Übertragungstransfer zwischen µController und FPGA kann in zwei Modi betrieben werden. Im Polling- oder IRQ-Verfahren. Über den Funktionsaufruf initPSI(FALSE) wird die Hardware fürs Polling-Verfahren voreingestellt. Ist Parameter TRUE aktiviert dieser den IRQ-Betrieb.

Im IRQ-Betrieb kann der µController sequentiell weiterarbeiten und wird nur über einen Interrupt zur Ausführung der zugehörigen ISR unterbrochen, wenn das FPGA den Pegel des PSI_RESULT auf High setzt. Die Latenz ist entsprechend größer als beim Polling-Verfahren, durch den Kontexwechsel des µControllers, der somit alle Daten des laufend Programs auf dem Stack sichert und die ISR-Programmdaten lädt.

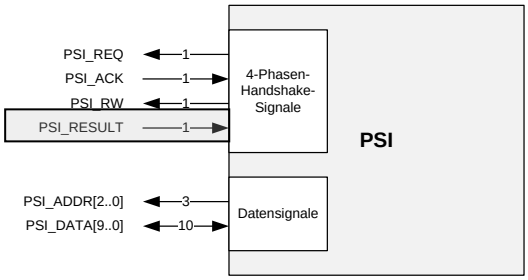


Abb. 5.13: PSI-Result

5.4 Synthese-Ergebnisse und Timings des Beschleunigers

In dem Synthese-Report sind die verwendeten Resourcen des FPGA aufgelistet, die das Prozessorelement nutzt (vgl. Tab. 5.2). Aus dem Synthese-Report ist ersichtlich, dass der Zustandsautomat in einer One-Hot Zustandskodierung synthetisiert wurde. Für jeden Zustand wird ein Daten-Flip-Flop genutzt und die Zustandscodierung ist mit insgesamt acht Flip-Flops realisiert worden.

Tab. 5.2: FPGA-Resourcenausnutzung für das Prozessorelement

	Anzahl	**Gesamtanzahl**	**% von Gesamt**
D-FFs	53	17344	1
4 Input LUTs	246	17344	1
Multipliers	1	28	3

Für die Timing-Simulation (vgl. Abb. 5.14) wurde eine Testbench erstellt, welche folgende Signale erzeugt:

- CLK
- RW
- REQ
- ACK
- DATA

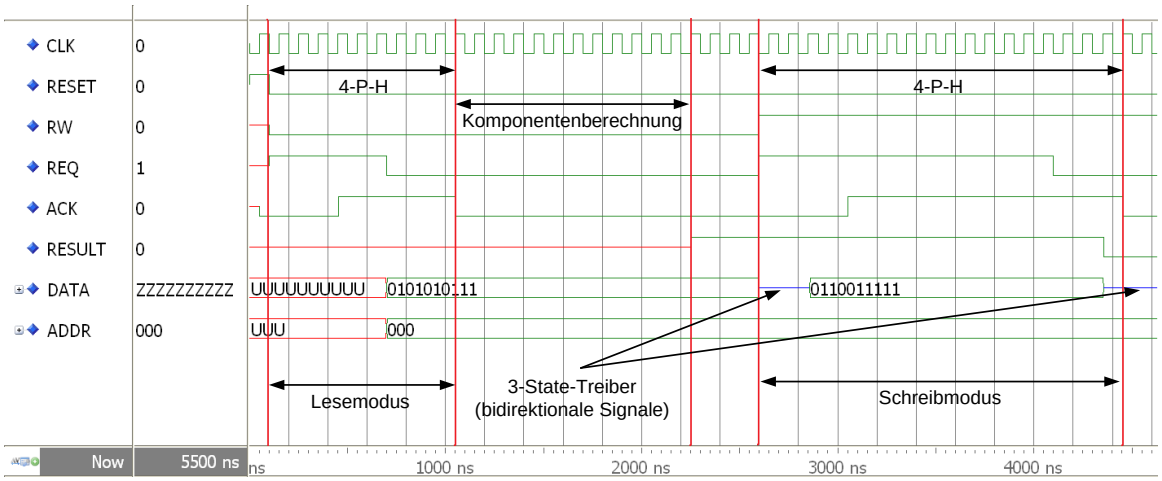


Abb. 5.14: Timing-Simulation des 4-Phasen-Handshake

5.5 Kombinierbare Hardware-Peripherie-Module mit PSI

Das Software-Modul PSI kann mit nachfolgenden Hardware-Peripherien des µControllers parallel betrieben werden.

ADC1,2 – Analog Ditital Converter
DAC – Digital Analog Converter
PWM1..3 – Pulse Width Modulation
SPI1 – Serial Peripheral Interface – 4 Drahtverbindung
UART2 – Universal Asynchronous Receiver Transmitter – RS232
I^2C – Inter Integrated Circuit – 2 Drahtverbindung
CAN – Controller Area Network – Feldbus

5.6 Grundkonfiguration des PSI-Moduls

Die Grundeinstellungen gewährleisten den korrekten Kopplungsbetrieb der diskreten Signalverbindungen, zwischen dem FPGA und µController, über die Schutzschaltung des SECURITYv2.1-Board. Über die Initialisierung mit der Software-Funktion initPSI() werden die Hardware-Pins des µController und des SECURITYv2.1-Boards für die Verwendung des PSI voreingestellt:

Bustreiber Steuersignal	Datenflussrichtung
IOB1..2	Input (FPGA zum µController)
IOB3	Statisch oder dynamisch als Input, nicht für die PSI-Kommunikation geschaltet
IOB4	Output (µController zum FPGA)
IOB5	Input (FPGA zum µController)

6. Varianten des Kommunikations- und Datentransfers

Werden unterschiedliche Komponenten, für die Datenweiterverarbeitung, auf der System-on-Chip-Plattform instanziiert oder unterschiedliche Software-Module im Softcore des µControllers verwendet, ist die Übertragung der bidirektionalen Adresse PSI_ADDR[2..0] (vgl. Abb. 6.3 und 6.4) während des Kommunikationstransfers für die Komponenten- oder Modulzuordnung erforderlich. Über die eindeutigen Bitkombinationen des 3bit Adressraums können sowohl Adressen, als auch Opcodes und ein 12bit Datenvektor übermittelt werden. Für einen höheren bidirektionalem Datendurchsatz bei gleichbleibenden Taktraten (FPGA mit CLK = 50MHz und der µControllermit einer Oszillatorfrequenz von 72MHz), kann die Datenbandbreite von 10bit auf 16bit, durch Veränderung der Vektorbreite von PSI_DATA[15|9..0] (vgl. Abb. 6.1 und 6.2), erweitert werden.

6.1 Kommunikation mit einer 16bit Datenbandbreite

Durch die Erweiterung der Bandbreite des PSI-DATA[15..0]-Signals, können pro Kommunikationstransfers wordbreite Datenpakete bidirektional übertragen werden. Die 3bit Vektorbreite des unidirektionalen Adressraums PSI_ADDR[2..0] für die Parametrisierung von 7 - wenn 0 = keine Funktion (nop) - ansonsten 8 FPGA Komponenten. Das 4-Phasen-Handshake und die Steuersignale PSI_RW und PSI_RESULT sind, wie zuvor in Kapitel 5.1 beschrieben, mit den vorherigen Konfigurationen identisch.

Tab. 6.1: PSI Hardware Pinbelegung und Software Signalvarianten 1

Daten-/Steuersignale				Development Boards						
Name	Typ	Anmerkung	Transfer Richtung	**TI-LPC** uC Port[Pin]	**SECURITYv2.1** Bustreiber	Connector Typ	Connector Pin	**NEXYS2** FPGA Signal	Pin	Connector FX2-100 Port-Pin
PSI_REQ	S	Auftrag	unidirektional	P1[12]	IOB4	X2	24	IOB4<3>	F11	J1A-34
PSI_ACK	S	Auftragsbestätigung	unidirektional	P1[8]	IOB5	X2	25	IOB5<2>	C14	J1A-41
PSI_RW	S	Schreiben/Lesen	unidirektional	P1[11]	IOB4	X2	23	IOB4<3>	F11	J1A-34
PSI_RESULT	S	Auftragsfertigstellung	unidirektional	P2[12]	IOB5	X3	9	IOB5<1>	A14	J1A-40
PSI_ADDR[0]	S	Adresse des Prozes-sorelements	unidirektional	P1[15]	IOB4	X2	33	IOB4<0>	B11	JA1-31
PSI_ADDR[1]				P1[14]			34	IOB4<1>	C11	JA1-32
PSI_ADDR[2]				P1[13]			29	IOB4<2>	E11	JA1-33
PSI_DATA[0]	D	Datum	bidirektional	P0[5]	IOB1	X3	36	IOB1<0>	A4	J1A-07
PSI_DATA[1]				P1[10]		X2	26	IOB1<1>	C3	J1A-08
PSI_DATA[2]				P0[13]			5	IOB1<2>	C4	J1A-09
PSI_DATA[3]				P0[14]			7	IOB1<3>	B6	J1A-10
PSI_DATA[4]				P0[19]		X3	15	IOB1<4>	D5	J1A-11
PSI_DATA[5]				P0[20]			14	IOB1<5>	C5	J1A-12
PSI_DATA[6]				P0[21]		X2	19	IOB1<6>	F7	J1A-13
PSI_DATA[7]				P0[22]			18	IOB1<7>	E7	J1A-14
PSI_DATA[8]				P0[29]	IOB2		6	IOB2<0>	A6	J1A-15
PSI_DATA[9]				P0[30]			15	IOB2<1>	C7	J1A-16
PSI_DATA[10]				**P0[4]**		**X3**	**37**	**IOB2<2>**	**F8**	**J1A-17**
PSI_DATA[11]				**P1[16]**		**X2**	**28**	**IOB2<3>**	**D7**	**J1A-18**
PSI_DATA[12]				**P1[17]**			**32**	**IOB2<4>**	**E8**	**J1A-19**
PSI_DATA[13]				**P1[18]**			**16**	**IOB2<5>**	**E9**	**J1A-20**
PSI_DATA[14]				**P1[19]**			**8**	**IOB2<6>**	**C9**	**J1A-21**
PSI_DATA[15]				**P1[20]**		**X3**	**6**	**IOB2<7>**	**A8**	**J1A-22**
IOB1_DIR	S	Transferrichtung der 1. IOB-Komponente	unidirektional	P2[3]	IOB1	X3	32			
IOB2_DIR	S	Transferrichtung der 2. IOB-Komponente	unidirektional	P2[4]	IOB2	X3	33			

Änderungen zur Standardkonfiguration sind fettgedruckt dargestellt.

Schreibemodus:

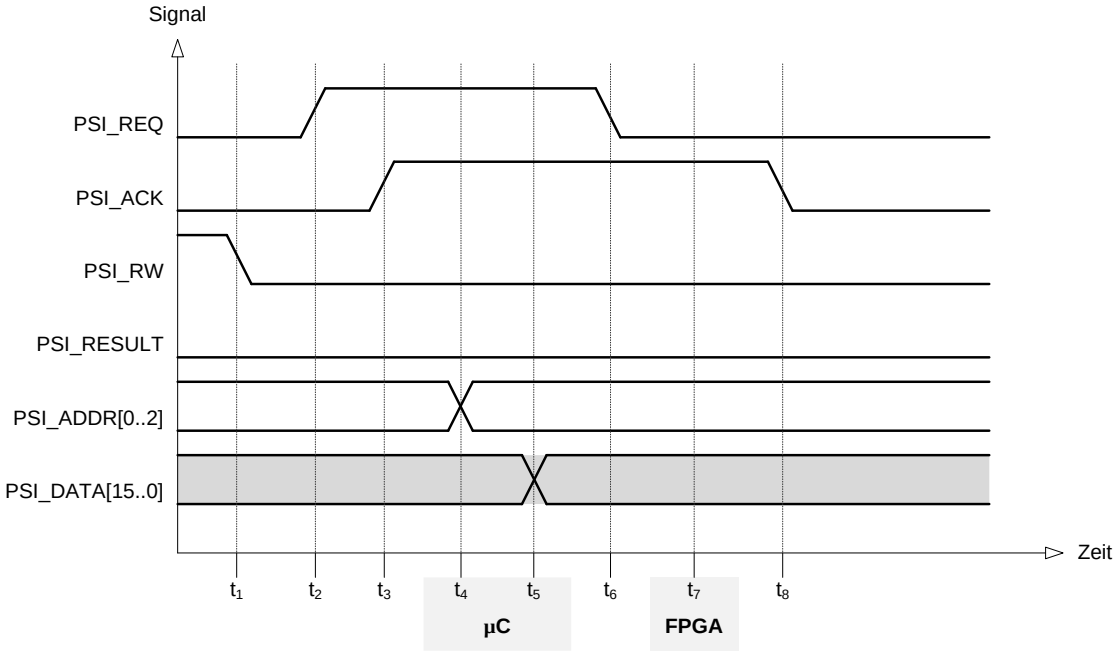


Abb. 6.1: 16bit Datentransfer vom µController zum FPGA

Lesemodus:

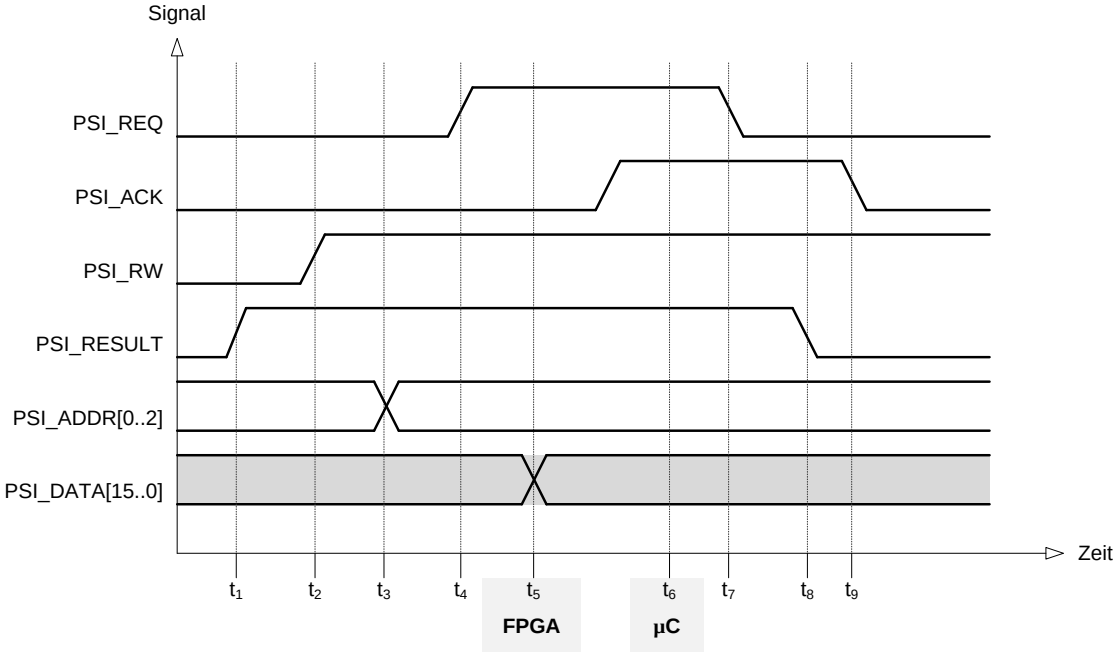


Abb. 6.2: 16bit Datentransfer vom FPGA zum µController

6.2 4-Phasen-Handshake mit bidirektionalen Adress- und Datensignalen

Die Adressierung mehrerer FPGA-Prozessorelemente oder Software-Module des µC erfordern bidirektionale Adresssignale. Voraussetzung ist die Realisierung einer FPGA Komponenten-Architektur, die ähnlich des µC LPC2468FET208 ist. D.h., dass jede Komponente Register implementiert, die die Konfiguration über lesenden Zugriff auf das Status- und Kontrollregister, als auch schreiben und lesenden Zugriff auf das ermöglichen. Rechenintensive Prozesse für Datenkalkulationen des µC können somit auf gleiche oder unterschiedliche Komponenten des Hardware-Beschleunigers, über 3bit-Kombinationen PSI_ADDR[2..0] zur Laufzeit, vom Master verteilt werden.

Tab. 6.2: PSI Hardware Pinbelegung und Software Signalvarianten 2

Daten-/Steuersignale				**Development Boards**						
Name	Typ	Anmerkung	Transfer Richtung	**TI-LPC** uC Port[Pin]	**SECURITYv2.1** Bustreiber	Connector Typ	Connector Pin	**NEXYS2** FPGA Signal	FPGA Pin	Connector FX2-100 Port-Pin
PSI_REQ	S	Auftrag	unidirektional	P1[12]	IOB4	X2	24	IOB4<3>	F11	J1A-34
PSI_ACK	S	Auftragsbestätigung	unidirektional	P1[8]	IOB5	X2	25	IOB5<2>	C14	J1A-41
PSI_RW	S	Schreiben/Lesen	unidirektional	P1[11]	IOB4	X2	23	IOB4<3>	F11	J1A-34
PSI_RESULT	S	Auftragsfertigstellung	unidirektional	P2[12]	IOB5	X3	9	IOB5<1>	A14	J1A-40
PSI_ADDR[0]	**S**	**Adresse des Prozessorelements**	**bidirektional**	**P1[18]**	**IOB2**	**X2**	**12**	**IOB2<5>**	**E9**	**J1A-20**
PSI_ADDR[1]				**P1[19]**			**8**	**IOB2<6>**	**C9**	**J1A-21**
PSI_ADDR[2]				**P1[20]**		**X3**	**6**	**IOB2<7>**	**A8**	**J1A-22**
PSI_DATA[0]	D	Datum	bidirektional	P0[5]	IOB1	X3	36	IOB1<0>	A4	J1A-07
PSI_DATA[1]				P1[10]		X2	26	IOB1<1>	C3	J1A-08
PSI_DATA[2]				P0[13]			5	IOB1<2>	C4	J1A-09
PSI_DATA[3]				P0[14]			7	IOB1<3>	B6	J1A-10
PSI_DATA[4]				P0[19]		X3	15	IOB1<4>	D5	J1A-11
PSI_DATA[5]				P0[20]			14	IOB1<5>	C5	J1A-12
PSI_DATA[6]				P0[21]		X2	19	IOB1<6>	F7	J1A-13
PSI_DATA[7]				P0[22]			18	IOB1<7>	E7	J1A-14
PSI_DATA[8]				P0[29]	IOB2		6	IOB2<0>	A6	J1A-15
PSI_DATA[9]				P0[30]			15	IOB2<1>	C7	J1A-16
PSI_DATA[10]				**P0[4]**		**X3**	**37**	**IOB2<2>**	**F8**	**J1A-17**
PSI_DATA[11]				**P1[16]**		**X2**	**28**	**IOB2<3>**	**D7**	**J1A-18**
IOB1_DIR	S	Transferrichtung der 1. IOB-Komponente	unidirektional	P2[3]	IOB1	X3	32			
IOB2_DIR	S	Transferrichtung der 2. IOB-Komponente	unidirektional	P2[4]	IOB2	X3	33			

Änderungen zur Standardkonfiguration sind fettgedruckt dargestellt.

Schreibemodus:

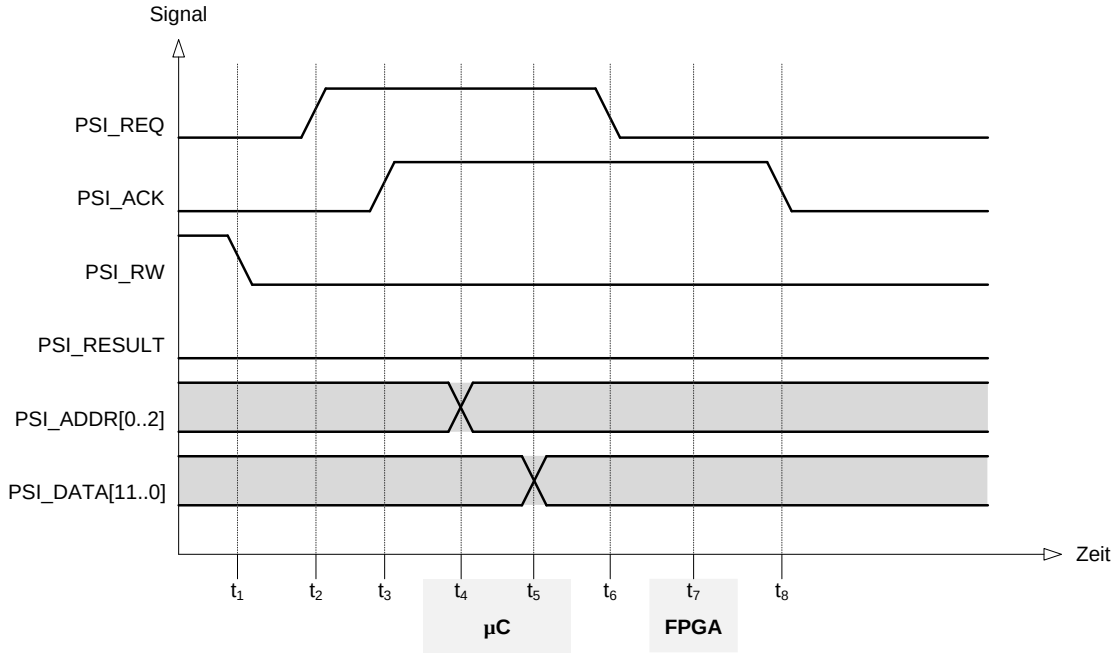


Abb. 6.3: Datentransfer vom µController zum FPGA mit 3bit bidirektionaler Adresse

Lesemodus:

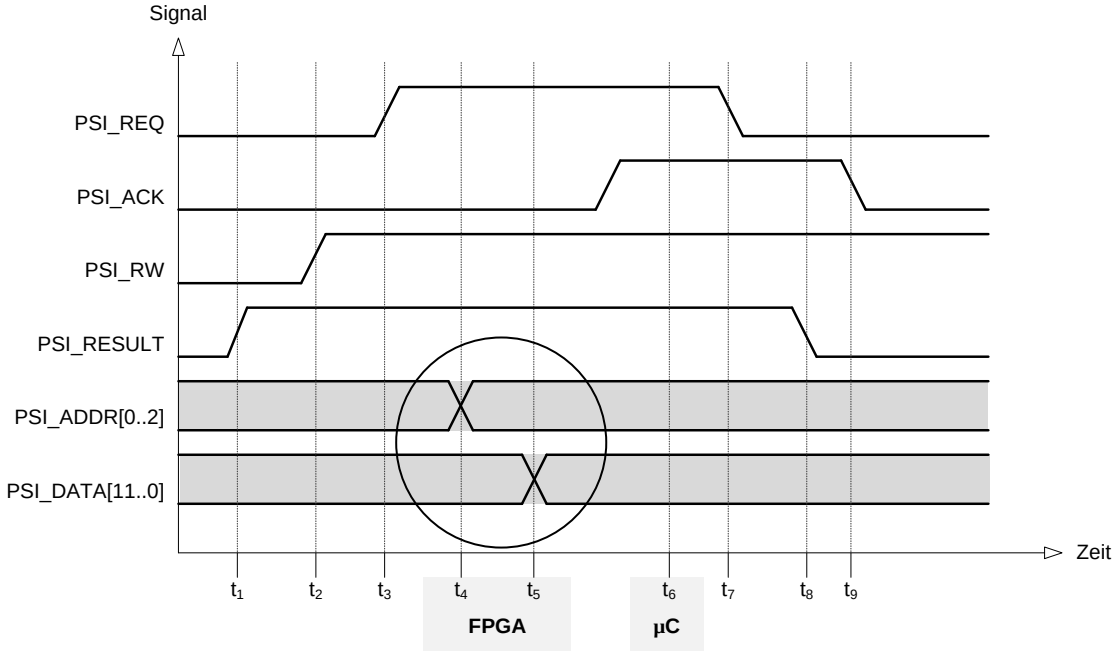


Abb. 6.4: Datentransfer vom FPGA zum µC mit 3bit bidirektionaler Adresse

7. Zusammenfassung

Im Rahmen des Forschungsprojekts FAUST (Fahrerassistenz- und Autonome Systeme), des Departments Informatik der Hochschule für Angewandte Wissenschaften (HAW), werden eingebettete Systeme entwickelt, die sich an den Vorgaben der derzeitigen Industriestandards orientieren. Diese Arbeit leistet hierzu einen Beitrag mit der Entwicklung einer Erprobungsplattform NEXSELv2 als Funktionsmuster, zum Aufbau individual konfigurierbarer Hardware-Software-Codesign-Anwendungen, für die Ausbildung der Studenten im Studienfach CE/CEP.

Die NEXSELv2-Plattform ist eine Hardware-Kopplung, zwischen der FPGA-basierten System-on-Programmable-Chip Plattform mit einem Spartan3E™ und einem µController ARM7™ TDMI. Diese ist aus zwei Basiseinheiten, den Development-Boards NEXYS2 und TI-LPC-2468-Stick und der Schutzschaltung des SECURITYv2.1-Boards aufgebaut. Die Hardware-Schnittstellen mit 38 I/O-Leitungen des FPGAs (FX-2 Connector) und des µControllers (X2 und X3) sind über das SECURITYv2.1-Board diskret verbunden. Alle Komponenten des Funktionsmusters können auch somit separat aktiv betrieben werden.

Für den technischen Schutz der diskret verbunden Hardware-I/Os und für die Betriebssicherheit, vor fehlerhafter Ansteuerung oder Beschaltung wurde das SECURITYv2.1-Board entwickelt. Das System ist aus vier Bustreiberstufen mit acht I/0-Leitungen und einer mit fünf I/O-Leitungen und einer zusätzlichen Sicherheitsschaltung bei abfallenden Netzspannungsversorgungen aufgebaut. Die fünf Bustreiberstufen können statisch über Jumper oder dynamisch zur Laufzeit, ebenso wie die Datentransferrichtung vom µController als Mastereinheit konfiguriert werden. Eine Abschaltung der einzelnen Bustreiberstufen ist möglich. Über den Hauptschalter des SECURITYv2.1-Boards werden alle Bustreiberstufen in den 3-State-Modus geschaltet, für den stand-alone Betrieb oder in den Transfermodus. Ein Zusätzlicher Taster ermöglicht zur Laufzeit die Development-Boards NEXYS2 und das TI-LPC-2468-Stick zu Entkopplung der I/Os.

Auf das im FPGA eingebettete Prozessorelement wurde ein VHDL-Modul zur Steuerung des 10bit Datentransfers als Slave-Request-Anwendung entwickelt und implementiert. Dieses Softwaremodul arbeitet mit synchronisierten Signalen über ein 4-Phasen-Handshake. Die SoPC-Plattform übernimmt in dieser Peripheriekonfiguration die Funktion eines Coprozessor zur beschleunigten digitalen Signalverarbeitung. Desgleichen wurde eine Master-Anwendung für den µController geschrieben, welche den bidirektionalen Datenfluss- und die Signalkopplung über ein lese-schreib-Signal steuert. Das 4-Phasen-Handshake-Protokoll besteht aus zwei Signalleitungen für den Kommunikationstransfer. Über die Adressierungsleitungen können Prozessorelemente im FPGA für die Signalverarbeitung ausgewählt werden.

8. Verzeichnisse

8.1 Literatur

[1] RECHENBERG, Peter: Technisches Schreiben (nicht nur) für Informatiker ▪ 3. Aufl. Carl Hanser Verlag München/Wien, 2006 ▪ ISBN-10: 3-446-40695-6, ISBN-13: 978-3-446-40695-7

[2] TTL Taschenbuch - 1. Teil ▪ 4. Aufl. IWT Verlag München, 2001 ▪ ISBN 3-88322-008-6

[3] TIETZE, Ulrich; SCHENK, Christoph: Halbleiter - und Schaltungstechnik ▪ 12. Aufl. Springer Verlag München, 2002 ▪ ISBN-10: 3540428496, ISBN-13: 978-3540428497

[4] FÜHRER, Arnold: Grundgebiete der Elektrotechnik Band 3 ▪ 4. Aufl. Fachbuchverlag Leipzig, 2000 ▪ ISBN-10: 3446192484 , ISBN-13: 978-3446192485

[5] NXP B.V.: ARM7™ TDMI (LPC2468FET208) UM10237 User Manual ▪ Rev. 01 - 18. July 2008 ▪ *http://www.nxp.com*

[6] hitex Development Tools GmbH: LPC-2468 Stick View ▪ Rev. 10/2007-001 ▪ *http://www.ehitex.de*

[7] hitex Development Tools GmbH: LPC-COM Board Data Sheet ▪ Rev. 12/2008-003 ▪ *http:// www.ehitex.de*

[8] hitex Development Tools GmbH: LPC-COM Board Schematics ▪ Rev. 12/2008-002 ▪ *http:// www.ehitex.de*

[9] hitex Development Tools GmbH: LPC-PROTO Board Data Sheet ▪ Rev. 12/2007-001 ▪ *http:// www.ehitex.de*

[10] XILINX Inc.: Spartan™-3E Gen. FPGA (XC3S1200E) Complete Data Sheet ▪ Rev. DS312 11/2006 ▪ *http:// www.xilinx.com*

[11] Digitlent Inc.: NEXYS2 Board Reference Manual And Specifications ▪ Rev. 01/2008-002 ▪ *http://www.digilentinc.com*

[12] Digitlent Inc.: FX2-BB Board User Manual ▪ Rev. 09/2006-003 ▪ *http://www.digilentinc.com*

[13] Digitlent Inc.: FX2-BB Board Schematics ▪ Rev. 02/2006-001 ▪ *http://www.digilentinc.com*

[14] Hochschule für Angewandte Wissenschaften Hamburg: TI Board BCv1 Manual ▪ Rev. 07/2008-002 ▪ *http://www.informatik.haw-hamburg.de* (TI-Labor)

[15] EFFORD Nick: Digital Image Processing ▪ 1. Aufl. Pearson/Addison Wesley Verlag, 2000 ▪ ISBN 0-201-59623-7

[16] REICHARDT, Jürgen; SCHWARZ, Bernd: VHDL-Synthese - Entwurf digitaler Schaltungen und Systeme ▪ 4. Oldenbourg Verlag München, 2007 ISBN: 978-3-486-58192-8

[17] HABERÄCKER, Peter; FISCHER, Max; NISCHWITZ, Alfred: Computergrafik und Bildverarbeitung ▪ 2. Aufl. Friedr. Vieweg & Sohn Verlag Wiesbaden, 2004 ▪ ISBN 978-3-8348-0186-9

[18] WAKERLY, John F.: Digital Design – Principles and Practices ▪ 4. Aufl. Pearson -Prentice Hall/-Education Inc. Verlag Stanford, 2006 ▪ ISBN: 0-13-186389-4

[19] CHU, Pong P.: FPGA Prototyping by VDHL Examples – XILINX Spartan™-3 ▪ 1. Aufl. John Wiley & Sons - Interscience Verlag New Jersey, Hoboken, 2008 ▪ ISBN: 978-0-470-18531-5

[20] SIEMERS, Christian; Sikora, Axel: Taschenbuch der Digitaltechnik ▪ 3. Aufl. Carl Hanser Verlag Leipzig, 2003 ▪ ISBN: 3-446-21862-9

[21] SCHNEIDER, Uwe; WERNER, Dieter: Taschenbuch der Informatik ▪ 4. Aufl. Carl Hanser Verlag Leipzig, 2001 ▪ ISBN: 3-446-21753-3

[22] WOLF, Wayne: High Performance Embedded Computing ▪ 1. Aufl. Morgan Kaufmann Verlag Chicago Amsterdam, 2006 ▪ ISBN-10: 012369485X ISBN-13: 978-0123694850

[23] BRINGSCHULTE, Uwe; Ungerer, Theo: Mikrocontroller und Mikroprozessoren ▪ 3. Aufl. Springer -Verlag Berlin Heidelberg New York, 2002 ▪ ISBN: 3-540-43095-4

[24] WOLF, Wayne: Computer As Components ▪ 2. Aufl. Morgan Kaufmann Verlag Boston New York, 2008 ▪ ISBN-10: 0123743974, ISBN-13: 978-0123743978

[25] KERNIGHAN, Brian W.; RICHIE, Dennis M.: The C Programming Language ▪ 2. Aufl. Prentice Hall/Engelwood Cliff. Verlag New York, 1988 ▪ ISBN: 0-13-110362-8

[26] Tektronix: TDS 544A Data Sheet ▪ 30. Mai 2009 ▪ *http://www.tek.com*

[27] Technology Limited: picoScope ADC-200 VI Referenz Manual ▪ 30. Mai 2009 ▪ *http:// www.picotech.com/oscilloscope.html*

[28] FLUKE Corp.: Digitalmultimeter 177 True RMS Manual▪ 30. Mai 2009 ▪ *http://www.fluke.de*

[29] TENMA TW: Multimeter 72-2055 User Manual Series ..2050 ▪ 30. Mai 2009 ▪ *http://* www.mcmconnect.com/tenma/

[30] Philips Electronics N.V.: PM-2525 ▪ 30. Mai 2009 ▪ http://www.arm.com

[31] ARM™ Community: Data Sheets ▪ 30. Mai 2009 ▪ http://www.arm.com

[32] GINOSAR, Ran: Fouteen Ways To Fool Your Synchronizer ▪ VLSI Systems Research Center, Technion - Israel Institute of Technology Haifa 32000, 2003 ▪ Proceedings of the Ninth International Symposium on Asynchronous Circuits and Systems (ASYNC'03)

8.2 Abkürzungen

[ANSI] ***American National Standards Institute***
Das amerikanische Standardisierungs- und Normungsgremium, welches Konventionen, wie: ASCII und ANSI-Standard-C, definierte (deutsches Gegenstück : DIN).

[ARM] ***Ehem. Acorn RISC Maschine/Advanced RISC Maschine***
Die ursprüngliche Bedeutung der Abkürzungen, wurde mittlerweile formal verworfen. Aktuell ein 32bit SoC-Mikroprozessor basierend auf den RISC Entwicklungsprinzipien.

[C] ***C/ANSI-Standard-C: C89, C95, C99, ISO-C***
Die Programmiersprache C, an den Bell Laboratories von Informatiker Dennis Ritchie entwickelt und Anfang 1973 veröffentlicht, gehorcht den imperativen und strukturierten Programmierparadigmen.

[DIGI] ***Digilent Inc.***
Digilent Inc. ist ein Marktführer in der Entwicklung, Herstellung und dem weltweiten Vertrieb von FPGA und Mikrocontroller-Technologien [www.digilentinc.com].

[FPGA] ***Field Programmable Gate Array***
Gate Array ist ein Integrierter Schaltkreis (IC) der Digitaltechnik, in den eine logische Schaltung programmiert werden kann. Die englische Bezeichnung kann übersetzt werden als: *im (Anwendungs-)Feld-programmierbare (Logik-)Gatter-Anordnung*. Anders als bei der Programmierung von Computern oder Steuerungen bezieht sich hier der Begriff Programm nur in zweiter Linie auf die Vorgabe zeitlicher Abläufe im Baustein, sondern vor allem auf die Definition von dessen Funktionsstruktur. Durch die Programmierung von Strukturvorschriften wird zunächst die grundlegende Funktionsweise einzelner universeller Blöcke im FPGA und deren Verschaltung untereinander festgelegt. Man spricht daher auch von der Konfiguration eines FPGAs [www.wikipedia.org].

[HITEX] ***Hitex Development Tools GmbH***
Hitex Development Tools ist seit vielen Jahren als Anbieter von innovativen und zuverlässigen Tools für Embedded-Entwickler bekannt [www.hitex.de].

[VHDL] ***Very High Speed Integrated Circuit Hardware Description Language (VHSIC)***
Die Programmier- und Hardwarebeschreibungssprache, entwickelt 1980 im Auftrag des US-DoD, zurs Konzeptionierung, Modellierung und Synthetisierung komplizierter abstrakter digitaler Schaltungssysteme. Existierende Erweiterungen wie VHDL-AMS (analog/mixed signal) dienen der Beschreibung analoger Systeme. Im Jahr 1985 wurde die erste kommerzielle Version der Programmier-sprache vom Konsortium: IBM, Texas Instruments und Intermetrics veröffentlicht. Aktuell ist VHDL-2008 durch den Standard IEEE-1076 genormt und vereinheitlicht. Gegenstück der europaweit verwendeten VHDL-Syntax ist Verilog.

[XILINX] ***XILINX Inc.(www.xilinx.com)***
Seit der Gründung vor mehr als 25 Jahren ist Xilinx ein Halbleiter-Marktführer. Xilinx programmierbare Chips sind die Innovation-Plattform, für heutige führende Unternehmen, für das Produktdesign [www.xilinx.com].

8.3 Grafiken und Fotos

7.4 Tabellen

A. Übersichten der Modulkomponenten, Ports und Pinbelegungen

A.1 NEXYS2-Board

Hirose FX2 Connector Pin Assignments [DIGI]

J1A	Name	FPGA	J1B	Name	FPGA
1	VCC3V3				
1	SHIELD				
2	VCC3V3		2	GND	
3	TMS	D15	3	TDO-ROM	
4	JTSEL		4	TCK	A17
5	TDO-FX2		5	GND	
6	FX2-IO1	B4	6	GND	
7	FX2-IO2	A4	7	GND	
8	FX2-IO3	C3	8	GND	
9	FX2-IO4	C4	9	GND	
10	FX2-IO5	B6	10	GND	
11	FX2-IO6	D5	11	GND	
12	FX2-IO7	C5	12	GND	
13	FX2-IO8	F7	13	GND	
14	FX2-IO9	E7	14	GND	
15	FX2-IO10	A6	15	GND	
16	FX2-IO11	C7	16	GND	
17	FX2-IO12	F8	17	GND	
18	FX2-IO13	D7	18	GND	
19	FX2-IO14	E8	19	GND	
20	FX2-IO15	E9	20	GND	
21	FX2-IO16	C9	21	GND	
22	FX2-IO17	A8	22	GND	
23	FX2-IO18	G9	23	GND	
24	FX2-IO19	F9	24	GND	
25	FX2-IO20	D10	25	GND	
26	FX2-IO21	A10	26	GND	
27	FX2-IO22	B10	27	GND	
28	FX2-IO23	A11	28	GND	
29	FX2-IO24	D11	29	GND	
30	FX2-IO25	E10	30	GND	
31	FX2-IO26	B11	31	GND	
32	FX2-IO27	C11	32	GND	
33	FX2-IO28	E11	33	GND	
34	FX2-IO29	F11	34	GND	
35	FX2-IO30	E12	35	GND	
36	FX2-IO31	F12	36	GND	
37	FX2-IO32	A13	37	GND	
38	FX2-IO33	B13	38	GND	
39	FX2-IO34	E13	39	GND	
40	FX2-IO35	A14	40	GND	
41	FX2-IO36	C14	41	GND	
42	FX2-IO37	D14	42	GND	
43	FX2-IO38	B14	43	GND	
44	FX2-IO39	A16	44	GND	
45	FX2-IO40	B16	45	GND	
46	GND		46	FX2-CLKIN	B9
47	FX2-CLKOUT	D9	47	GND	
48	GND		48	FX2-CLKIO	M9
49	VCCFX2		49	VCCFX2	
50	VCCFX2		50	SHIELD	

A.2 SECURITYv2.1-Board

Pin-Connect Block (LPC2468-Stick/PCM80-Connector)

Steuerregister [Pins]	I/O-Port [Pin]	I/O-FPGA [Pin]	Peripherie/Funktion [00/1]		[01/2]	[10/3]	[11/4]	Reset
				(Software)				
PINSEL0[1:0]	P0[0]		GPIO		RD1	TXD3	SDA1	0x00
PINSEL0[3:2]	P0[1]		GPIO		TD1	RXD3	SCL1	0x00
PINSEL0[9:8]	P0[4]	IOB2[2]	GPIO		I2SRX_CLK	RD2	CAP2[0]	0x00
		> LCDVD[0]						
PINSEL0[11:10]	P0[5]	IOB1[0]	GPIO	(PSI_DATA[0])	I2SRX_WS	TD2	CAP2[1]	0x00
		> LCDVD[1]						
PINSEL0[21:20]	P0[10]		GPIO		TXD2	SDA2	MAT3[0]	0x00
PINSEL0[23:22]	P0[11]		GPIO		RXD2	SCL2	MAT3[1]	0x00
PINSEL0[25:24]	P0[12]		GPIO		USB_PPWR2	MISO1	AD0[6]	0x00
PINSEL0[27:26]	P0[13]	IOB1[2]	GPIO	(PSI_DATA[2])	USB_UP_LED2	MOSI1	AD0[7]	0x00
PINSEL0[29:28]	P0[14]	IOB1[3]	GPIO	(PSI_DATA[3])	USB_HSTEN2	USB_CONN ECT2	SSEL1	0x00
PINSEL0[31:30]	P0[15]	IOB4[7]	GPIO		TXD1	SCK0	SCK	0x00
PINSEL1[1:0]	P0[16]	IOB4[6]	GPIO		RXD1	SSEL0	SSEL	0x00
PINSEL1[3:2]	P0[17]	IOB5[4]	GPIO		CTS1	MISO0	MISO	0x00
PINSEL1[5:4]	P0[18]	IOB4[5]	GPIO		DCD1	MOSI0	MOSI	0x00
PINSEL1[7:6]	P0[19]	IOB1[4]	GPIO	(PSI_DATA[4])	DSR1	MCICLK	SDA1	0x00
PINSEL1[9:8]	P0[20]	IOB1[5]	GPIO	(PSI_DATA[5])	DTR1	MCICMD	SCL1	0x00
PINSEL1[11:10]	P0[21]	IOB1[6]	GPIO	(PSI_DATA[6])	RI1	MCIPWR	RD1	0x00
PINSEL1[13:12]	P0[22]	IOB1[7]	GPIO	(PSI_DATA[7])	AD0[0]	I2SRX_CLK	CAP3[0]	0x00
PINSEL1[17:16]	P0[24]		GPIO		AD0[1]	I2SRX_WS	CAP3[1]	0x00
PINSEL1[19:18]	P0[25]		GPIO		AD0[2]	I2SRX_SDA	TXD3	0x00
PINSEL1[21:20]	P0[26]		GPIO		AD0[3]	AOUT	RXD3	0x00
PINSEL1[27:26]	P0[29]	IOB2[0]	GPIO	(PSI_DATA[8])	USB_D+1	Reserved	Reserved	0x00
PINSEL1[29:28]	P0[30]	IOB2[1]	GPIO	(PSI_DATA[9])	USB_D-1	Reserved	Reserved	0x00
PINSEL1[31:30]	P0[31]		GPIO		USB_D+2	Reserved	Reserved	0x00
PINSEL2[1:0]	P1[0]	IOB3[0]	GPIO		ENET_TXD0	Reserved	Reserved	0x00
PINSEL2[3:2]	P1[1]	IOB3[1]	GPIO		ENET_TXD1	Reserved	Reserved	0x00
PINSEL2[5:4]	P1[2]	IOB3[2]	GPIO		ENET_TXD2	MCICLK	PWM0[1]	0x00
PINSEL2[7:6]	P1[3]	IOB3[3]	GPIO		ENET_TXD3	MCICMD	PWM0[2]	0x00
PINSEL2[9:8]	P1[4]	IOB3[4]	GPIO		ENET_TX_EN	Reserved	Reserved	0x00
PINSEL2[11:10]	P1[5]	IOB3[5]	GPIO		ENET_TX_ER	MCIPWR	PWM0[3]	0x00
PINSEL2[13:12]	P1[6]	IOB3[6]	GPIO		ENET_TX_CLK	MCIDAT0	PWM0[4]	0x00
PINSEL2[15:14]	P1[7]	IOB3[7]	GPIO		ENET_COL	MCIDAT1	PWM0[5]	0x00

Pin-Connect Block (LPC2468-Stick/PCM80-Connector)

Steuerregister [Pins]	I/O-Port [Pin]	I/O-FPGA [Pin]	Peripherie/Funktion [00/1]	(Software)	[01/2]	[10/3]	[11/4]	Reset
PINSEL2[17:16]	P1[8]	IOB5[2]	GPIO	(PSI_ACK)	ENET_CRS_DV ENET_CRS	Reserved	Reserved	0x00
PINSEL2[19:18]	P1[9]	IOB5[3]	GPIO		ENET_RXD0	Reserved	Reserved	0x00
PINSEL2[21:20]	P1[10]	IOB1[1]	GPIO	(PSI_DATA[1])	ENET_RXD1	Reserved	Reserved	0x00
PINSEL2[23:22]	P1[11]	IOB4[4]	GPIO	(PSI_RW)	ENET_RXD2	MCIDAT2	PWM0[6]	0x00
PINSEL2[25:24]	P1[12]	IOB4[3]	GPIO	(PSI_REQ)	ENET_RXD3	MACIDAT3	PCAP0[0]	
PINSEL2[27:26]	P1[13]	IOB4[2]	GPIO	(PSI_ADDR[2])	ENET_RX_DV	Reserved	Reserved	0x00
PINSEL2[29:28]	P1[14]	IOB4[1]	GPIO	(PSI_ADDR[1])	ENET_RX_ER	Reserved	Reserved	0x00
PINSEL2[31:30]	P1[15]	IOB4[0]	GPIO	(PSI_ADDR[0])	ENET_REF_CLK ENET_RX_CLK	Reserved	Reserved	0x00
PINSEL3[1:0]	P1[16]	IOB2[3]	GPIO		ENET_MDC	Reserved	Reserved	0x00
PINSEL3[3:2]	P1[17]	IOB2[4]	GPIO		ENET_MDIO	Reserved	Reserved	0x00
PINSEL3[5:4]	P1[18]	IOB2[5]	GPIO		USB_UP_LED1	PWM1[1]	CAP1[0]	0x00
PINSEL3[7:6]	P1[19]	IOB2[6]	GPIO		USB_TX_E1	USB_PPWR1	CAP1[1]	0x00
PINSEL3[9:8]	P1[20]	IOB2[7]	GPIO		USB_TX_DP1 > LCDVD[6] > LCDVD[10]	PWM1[2]	SCK0	0x00
PINSEL3[11:10]	P1[21]		GPIO		USB_TX_DM1 > LCDVD[7] > LCDVD[11]	PWM1[3]	SSEL0	0x00
PINSEL3[13:12]	P1[22]		GPIO		USB_RCV1 > LCDVD[8] > LCDVD[12]	USB_PW1	MAT1[0]	0x00
PINSEL3[23:22]	P1[27]		GPIO		USB_INT1 > LCDVD[13] > LCDVD[21]	USB_OVRCR1	CAP0[1]	0x00
PINSEL3[25:24]	P1[28]		GPIO		USB_SCL1 > LCDVD[14] > LCDVD[22]	PCAP1[0]	MAT0[0]	0x00
PINSEL3[27:26]	P1[29]		GPIO		USB_SDA1 > LCDVD[15] > LCDVD[23]	PCAP1[1]	MAT0[1]	0x00
PINSEL3[29:28]	P1[30]		GPIO		USB_PWRD2	VBUS	AD0[4]	0x00
PINSEL4[1:0]	P2[0]		GPIO		PWM1[1]	TXD1	TRACECLK > LCDPWR	0x00
PINSEL4[3:2]	P2[1]		GPIO		PWM1[2]	RXD1	PIPESTAT0 > LCDLE	0x00

Pin-Connect Block (LPC2468-Stick/PCM80-Connector)

Steuerregister [Pins]	I/O-Port [Pin]	I/O-FPGA [Pin]	Peripherie/Funktion [00/1] (Software)	[01/2]	[10/3]	[11/4]	Reset
PINSEL4[5:4]	P2[2]		GPIO	PWM1[3]	CTS1	PIPESTAT1 > LCDDCLK	0x00
PINSEL4[7:6]	P2[3]		GPIO (IOB1_DIR)	PWM1[4]	DCD1	PIPESTAT2 > LCDFP	0x00
PINSEL4[9:8]	P2[4]		GPIO (IOB2_DIR)	PWM1[5]	DSR1	TRACESYNC > LCDENAB > LCDM	0x00
PINSEL4[11:10]	P2[5]		GPIO (IOB3_DIR)	PWM1[6]	DTR1	TRACEPKT0 > LCDLP	0x00
PINSEL4[17:16]	P2[8]		GPIO (IOB4_DIR)	TD2	TXD2	TRACEPKT3 > LCDVD[2] > LCDVD[6]	0x00
PINSEL4[19:18]	P2[9]		GPIO (IOB5_DIR)	USB_CONN ECT1	RXD2	EXTIN0 > LCDVD[3] > LCDVD[7]	0x00
PINSEL4[23:22]	P2[11]	IOB5[0]	GPIO	EINT1 LCDCLKIN	MCIDAT1	I2STX_CLK	0x00
PINSEL4[25:24]	P2[12]	IOB5[1]	GPIO (PSI_RESULT)	EINT2 > LCDVD[4] > LCDVD[3] > LCDVD[8] > LCDVD[18]	MCIDAT2	I2STX_WS	0x00
PINSEL4[27:26]	P2[13]	RDY_UC	GPIO (RDY_UC)	EINT3 > LCDVD[5] > LCDVD[9] > LCDVD[19]	MCIDAT3	I2STX_SDA	0x00

> : Eine externe Ansteuerung der LCD-Peripherie ist nicht im Chipsatz des LPC2468FET208 implementiert

Tab. A.1: Jumper-Einstellungen des SECURITYv2.1-Boards

Jumper Settings																			
Type		Connection															Function		
				IC					Jumper		Connector		µC						
Name	Pin	Signal	Direction	Nr.	Pin	Description	Typ	Level	Name	Pin	Pin	Name	Signal	Function	Level		Description	Type	Code
J1	1	VCC	**IN**														A[7..0] as input (B = A)	static	2-1
	2	DIR	**OUT**	1	1	BUS Treiber	SN74LVC245ADW												
	3	GND	**IN**														B[7..0] as input (A = B)	static	2-3
	4	IO_DIR0	**IN**								32	X3	P2.3	IO_DIR0	low		A[7..0] as input (B = A)	dynamic	2-4
															high		B[7..0] as input (A = B)		
J2	1	CE	**IN**						J13	2							Input settings of J13	static	2-1
	2	N_OE	**OUT**	1	19	BUS Treiber	SN74LVC245ADW												
	3	VCC	**IN**														IC disabled (A[7..0] = B[7..0] = 'Z')	static	2-3
J3	1	VCC	**IN**														A[7..0] as input (B = A)	static	2-1
	2	DIR	**OUT**	2	1	BUS Treiber	SN74LVC245ADW												
	3	GND	**IN**														B[7..0] as input (A = B)	static	2-3
	4	IO_DIRx	**IN**						J11	2							Input settings of J11	static	2-4
J4	1	CE	**IN**						J13	2							Input settings of J13	static	2-1
	2	N_OE	**OUT**	2	19	BUS Treiber	SN74LVC245ADW												
	3	VCC	**IN**														IC disabled (A[7..0] = B[7..0] = 'Z')	static	2-3
J5	1	VCC	**IN**														A[7..0] as input (B = A)	static	2-1
	2	DIR	**OUT**	3	1	BUS Treiber	SN74LVC245ADW												
	3	GND	**IN**														B[7..0] as input (A = B)	static	2-3
	4	IO_DIR2	**IN**								30	X3	P2.5	IO_DIR2	low		A[7..0] as input (B = A)	dynamic	2-4
															high		B[7..0] as input (A = B)		
J6	1	CE	**IN**						J13	2							Input settings of J13	static	2-1
	2	N_OE	**OUT**	3	19	BUS Treiber	SN74LVC245ADW												
	3	VCC	**IN**														IC disabled (A[7..0] = B[7..0] = 'Z')	static	2-3
J7	1	VCC	**IN**														A[7..0] as input (B = A)	static	2-1
	2	DIR	**OUT**	4	1	BUS Treiber	SN74LVC245ADW												
	3	GND	**IN**														B[7..0] as input (A = B)	static	2-3
	4	IO_DIR3	**IN**								29	X3	P2.8	IO_DIR3	low		A[7..0] as input (B = A)	dynamic	2-4

															high	B[7..0] as input (A = B)		
J8	1	CE	**IN**						J13	2						Input settings of J13	static	2-1
	2	N_OE	**OUT**	4	19	BUS Treiber	SN74LVC245ADW											
	3	VCC	**IN**													IC disabled (A[7..0] = B[7..0] = 'Z')	static	2-3
J9	1	VCC	**IN**													A[7..0] as input (B = A)	static	2-1
	2	DIR	**OUT**	5	1	BUS Treiber	SN74LVC245ADW											
	3	GND	**IN**													B[7..0] as input (A = B)	static	2-3
	4	IO_DIR4	**IN**								28	X3	P2.9	IO_DIR4	low	A[7..0] as input (B = A)	dynamic	2-4
															high	B[7..0] as input (A = B)		
J10	1	CE	**IN**						J13	2						Input settings of J13	static	2-1
	2	N_OE	**OUT**	5	19	BUS Treiber	SN74LVC245ADW											
	3	VCC	**IN**													IC disabled (A[7..0] = B[7..0] = 'Z')	static	2-3
J11	1	IO_DIR0	**IN**								32	X3	P2.3	IO_DIR0	low	A[7..0] as input (B = A)	dynamic	2-1
															high	B[7..0] as input (A = B)		
	2	IO_DIRx	**OUT**						J3	4								
	3	IO_DIR1	**IN**								33	X3	P2.4	IO_DIR1	low	A[7..0] as input (B = A)	dynamic	2-3
															high	B[7..0] as input (A = B)		
J12	1	VCC	**IN**	7	3	Spannungsregler	LF33ABDT	+3V3								Ext. main power	static	2-1
	2	VCC	**OUT**															
	3	VCC	**IN**								1	X3	+3V3	VCC		Int. main power	static	2-3
J13	1	READY	**IN**	6	3	NAND	74HC00D	low								ICs enabled	dynamic	2-1
				6	3	NAND	74HC00D	high								ICs disabled (A[7..0] = B[7..0] = 'Z')		
	2	CE	**OUT**						J2, 4, 6, 8, 10	1								
	3	VCC	**IN**													All ICs disabled (A[7..0] = B[7..0] = 'Z')	static	2-3

X-X Standardkonfiguration

VCC DC +3V3

Tab. A.2: Bustreiber-Pinning des SECURITYv2.1-Boards

Protection Circuit																	
NEXYS2 Board				**SECURITYv2 Board**									**LPC-Stick/TI-Board**				
FPGA XC3S1200E		Connector FX2-100[JA1]		Connector BL		Jumper 4pin	IC SN74LVC245ADW					Jumper 3pin	Connector SL		µC LPC2468FET208		
								A		B							
Pin	Function	Pin	Name	Nr.	Pin	Nr.	Nr.	Pin	Function	Pin	Function	Nr.	Pin	Name	Signal	Function	Board Collision
										20	VCC	J12[2]					
										19	N_OE	J2[2]					
						J1[2]		1	DIR								
A4	PSI_DATA[0]	7	FX2-IO02	1	1		1	2	1A1	18	1B1		36	X3	P0.5	PSI_DATA[0]	
C3	PSI_DATA[1]	8	FX2-IO03		2			3	2A1	17	2B1		39	X3	P0.12	PSI_DATA[1]	
C4	PSI_DATA[2]	9	FX2-IO04		3			4	3A1	16	3B1		5	X2	P0.13	PSI_DATA[2]	
B6	PSI_DATA[3]	10	FX2-IO05		4			5	4A1	15	4B1		7	X2	P0.14	PSI_DATA[3]	
D5	PSI_DATA[4]	11	FX2-IO06		5			6	5A1	14	5B1		15	X3	P0.19	PSI_DATA[4]	
C5	PSI_DATA[5]	12	FX2-IO07		6			7	6A1	13	6B1		14	X3	P0.20	PSI_DATA[5]	
F7	PSI_DATA[6]	13	FX2-IO08		7			8	7A1	12	7B1		19	X2	P0.21	PSI_DATA[6]	
E7	PSI_DATA[7]	14	FX2-IO09		8			9	8A1	11	8B1		18	X2	P0.22	PSI_DATA[7]	
								10	GND								
										20	VCC	J12[2]					
										19	N_OE	J4[2]					
						J3[2]		1	DIR								
A6	PSI_DATA[0]	15	FX2-IO10	1	9		2	2	1A1	18	1B1		6	X2	P0.29	PSI_DATA[0]	
C7	PSI_DATA[1]	16	FX2-IO11		10			3	2A1	17	2B1		15	X2	P0.30	PSI_DATA[1]	
F8	IO0	17	FX2-IO12		11			4	3A1	16	3B1		37	X3	P0.4	IO0	
D7	IO1	18	FX2-IO13		12			5	4A1	15	4B1		28	X2	P1.16	IO1	LCD-DB4 (D0)
E8	IO2	19	FX2-IO14		13			6	5A1	14	5B1		32	X2	P1.17	IO2	LCD-DB5 (D1)
E9	IO3	20	FX2-IO15		14			7	6A1	13	6B1		16	X2	P1.18	IO3	LCD-DB6 (D2)
C9	IO4	21	FX2-IO16		15			8	7A1	12	7B1		8	X2	P1.19	IO4	LCD-DB7 (D3)

A8	IO5	22	FX2-IO17		16			9	8A1	11	8B1		6	X3	P1.20	IO5	LCD-RS
								10	GND								
										20	VCC	J12[2]					
										19	N_OE	J6[2]					
						J5[2]		1	DIR								
G9	IO6	23	FX2-IO18		17		3	2	1A1	18	1B1		38	X2	P1.0	IO6	2mm Buchse: P1.0
F9	IO7	24	FX2-IO19	1	18			3	2A1	17	2B1		36	X2	P1.1	IO7	2mm Buchse: P1.1
D10	IO8	25	FX2-IO20		19			4	3A1	16	3B1		20	X2	P1.2	IO8	2mm Buchse: P1.2
A10	IO9	26	FX2-IO21		20			5	4A1	15	4B1		21	X2	P1.3	IO9	2mm Buchse: P1.3
B10	IO10	27	FX2-IO22		1			6	5A1	14	5B1		37	X2	P1.4	IO10	2mm Buchse: P1.4
A11	IO11	28	FX2-IO23	2	2			7	6A1	13	6B1		31	X2	P1.5	IO11	2mm Buchse: P1.5
D11	IO12	29	FX2-IO24		3		3	8	7A1	12	7B1		30	X2	P1.6	IO12	2mm Buchse: P1.6
E10	IO13	30	FX2-IO25		4			9	8A1	11	8B1		22	X2	P1.7	IO13	2mm Buchse: P1.7
								10	GND								
										20	VCC	J12[2]					
										19	N_OE	J8[2]					
						J7[2]		1	DIR								
B11	PSI_ADDR[2]	31	FX2-IO26		5			2	1A1	18	1B1		33	X2	P1.15	PSI_ADDR[2]	2mm Buchse/Taster: P1.15
C11	PSI_ADDR[1]	32	FX2-IO27		6			3	2A1	17	2B1		34	X2	P1.14	PSI_ADDR[1]	2mm Buchse/Taster: P1.14
E11	PSI_ADDR[0]	33	FX2-IO28		7			4	3A1	16	3B1		29	X2	P1.13	PSI_ADDR[0]	2mm Buchse/Taster: P1.13
F11	PSI_REQ	34	FX2-IO29		8		4	5	4A1	15	4B1		24	X2	P1.12	PSI_REQ	2mm Buchse/Taster: P1.12
E12	PSI_RW	35	FX2-IO30	2	9			6	5A1	14	5B1		23	X2	P1.11	PSI_RW	2mm Buchse/Taster: P1.11
F12	SPI_MOSI	36	FX2-IO31		10			7	6A1	13	6B1		19	X3	P0.18	SPI_MOSI	
A13	SPI_SSEL	37	FX2-IO32		11			8	7A1	12	7B1		20	X3	P0.16	SPI_SSEL	
B13	SPI_SCLK	38	FX2-IO33		12			9	8A1	11	8B1		21	X3	P0.15	SPI_SCLK	
								10	GND								
										20	VCC	J12[2]					
							5			19	N_OE	J10[2]					
						J9[2]		1	DIR								

E13	PSI_ACK	39	FX2-IO34	2	13		2	1A1	18	1B1		10	X3	P2.11	PSI_ACK	2mm Buchse: EINT1
A14	PSI_RESULT	40	FX2-IO35		14		3	2A1	17	2B1		9	X3	P2.12	PSI_RESULT	2mm Buchse: EINT2
C14	IO14	41	FX2-IO36		15		4	3A1	16	3B1		25	X2	P1.8	IO14	2mm Buchse/Taster: P1.8
D14	IO15	42	FX2-IO37		16		5	4A1	15	4B1		35	X2	P1.9	IO15	2mm Buchse/Taster: P1.9
B14	IO16	43	FX2-IO38		17		6	5A1	14	5B1		26	X2	P1.10	IO16	2mm Buchse/Taster: P1.10
A16	SPI_MISO	44	FX2-IO39		18		7	6A1	13	6B1		18	X3	P0.17	SPI_MISO	
							8	7A1	12	7B1						
							9	8A1	11	8B1						
							10	GND								

Tab. A.3: Ready-Signale des SECURITYv2.1-Boards

Protection Enable														
NEXYS2 Board				**SECURITYv2 Board**							**LPC-Stick/TI-Board**			
FPGA XC3S1200E		Connector FX2-100[JA1]		Connector BL		Jumper 4pin	IC 74HC00D			Jumper 3pin	Connector SL		µC LPC2468FET208	
Pin	Function	Pin	Name	Nr.	Pin	Nr.	Nr.	Pin	Function	Nr.	Pin	Name	Signal	Function
								1	1A		8	X3	P2.13	RDY_UC
B16	PDY_FPGA	45	FX2-IO40	2	19			2	1B					
						J13[1]		3	1Y					
								4	2A					
								5	2B					
								6	2Y					
								7	GND					
							6	8	3A					
								9	3B					
								10	3Y					
								11	4A					
								12	4B					
								13	4Y					
								14	VCC	J12[2]				

A.3 LPC-Stick/TI-Board

Tab. A.4: LPC-2468-Stick

Connector			
PCM80			
Pin	Function	Pin	Function
1	+3V3	**2**	+3V3
3	RESET_N	**4**	PWREN_N
5	USB_D-2	**6**	P0.00
7	P0.31	**8**	P0.01
9	P0.13	**10**	GND
11	P0.29	**12**	P1.20
13	P0.14	**14**	P1.21
15	P1.19	**16**	P2.13
17	P1.22	**18**	P2.12
19	P1.27	**20**	P2.11
21	GND	**22**	GND
23	P1.28	**24**	P0.10
25	P1.29	**26**	P0.11
27	P1.30	**28**	P0.20
29	P0.30	**30**	P0.19
31	P1.18	**32**	GNDA
33	GND	**34**	
35	P0.22	**36**	P0.17
37	P0.21	**38**	P0.18
39	P1.02	**40**	P0.16
41	P1.03	**42**	P0.15
43	P1.07	**44**	GND
45	P1.11	**46**	P0.24
47	P1.12	**48**	P0.23
49	P1.08	**50**	P0.25
51	P1.10	**52**	P0.26
53	GND	**54**	GND
55	P1.16	**56**	P2.09
57	P1.13	**58**	P2.08
59	P1.06	**60**	P2.05
61	P1.05	**62**	P2.01
63	P1.17	**64**	P2.03
65	P1.15	**66**	P2.04
67	P1.14	**68**	P2.02
69	P1.09	**70**	P2.00
71	P1.01	**72**	P0.05
73	P1.04	**74**	P0.04
75	P1.00	**76**	GND
77		**78**	(P0.12)
79	+5V0	**80**	+5V0

Tab. A.5: LPC-2468-Stick Modulpins

Pin			Port 0				Port 1				Port 2			
Bit		Info	Function Code PINSEL(0,1)				Function Code PINSEL(2,3)				Function Code PINSEL(4,5)			
dec	hex		00	01	10	11	00	01	10	11	00	01	10	11
00	0x00	Signal	P0[00]	RD1	TXD3	SDA1	P1[00]	ENET_TXD0			P2[00]	PWM1[1]	TXD1	TRACECLK
		Module	**GPIO P0**	**CAN 1**	**UART 3**	**I²C 1**	**GPIO P1**	**RMII/MII**			**GPIO P2**	**PWM 1**	**UART 1**	**Trace Clock**
		Direction	IO	I	O	IO	IO	O			IO	O	O	O
01	0x01	Signal	P0[01]	TD1	RXD3	SCL1	P1[01]	ENET_TXD1			P2[01]	PWM1[2]	RXD1	PIPESTAT0
		Module	**GPIO P0**	**CAN 1**	**UART 3**	**I²C 1**	**GPIO P1**	**RMII/MII**			**GPIO P2**	**PWM 1**	**UART 1**	**Pipeline Status Bit**
		Direction	IO	O	I	IO	IO	O			IO	O	I	O
02	0x02	Signal					P1[02]	ENET_TXD2	MCICLK	PWM0[1]	P2[02]	PWM1[3]	CTS1	PIPESTAT1
		Module					**GPIO P1**	**RMII/MII**	**SD/MMC**	**PWM 0**	**GPIO P2**	**PWM 1**	**UART 1**	**Pipeline Status Bit**
		Direction					IO	O	O	O	IO	O	I	O
03	0x03	Signal					P1[03]	ENET_TXD3	MCICMD	PWM0[2]	P2[03]	PWM1[4]	DCD1	PIPESTAT2
		Module					**GPIO P1**	**RMII/MII**	**SD/MMC**	**PWM 0**	**GPIO P2**	**PWM 1**	**UART 1**	**Pipeline Status Bit**
		Direction					IO	O	IO	O	IO	O	I	O
04	0x04	Signal	P0[04]	I2SRX_CLK	RD2	CAP2[0]	P1[04]	ENET_TX_EN			P2[04]	PWM1[5]	DSR1	TRACESYNC
		Module	**GPIO P0**	**I²S-Bus**	**CAN 2**	**Timer 2**	**GPIO P1**	**RMII/MII**			**GPIO P2**	**PWM 1**	**UART 1**	**Trace Synchronisation**
		Direction	IO	IO	I	I	IO	O			IO	O	I	O
05	0x05	Signal	P0[05]	I2SRX_WS	TD2	CAP2[1]	P1[05]	ENET_TX_ER	MCIPWR	PWM0[3]	P2[05]	PWM1[6]	DTR1	TRACEPKT0
		Module	**GPIO P0**	**I²S-Bus**	**CAN 2**	**Timer 2**	**GPIO P1**	**RMII/MII**	**SD/MMC**	**PWM 0**	**GPIO P2**	**PWM 1**	**UART 1**	**Trace Packet Bit**
		Direction	IO	IO	O	I	IO	O	O	O	IO	O	I	O
06	0x06	Signal					P1[06]	ENET_TX_CLK	MCIDAT0	PWM0[4]				
		Module					**GPIO P1**	**RMII/MII**	**SD/MMC**	**PWM 0**				
		Direction					IO	I	IO	O				
07	0x07	Signal					P1[07]	ENET_COL	MCIDAT1	PWM0[5]				
		Module					**GPIO P1**	**RMII/MII**	**SD/MMC**	**PWM 0**				
		Direction					IO	I	IO	O				
08	0x08	Signal					P1[08]	ENET_CRS_DV/ENET_CRS			P2[08]	TD2	TXD2	TRACEPKT3
		Module					**GPIO P1**	**RMII/MII**			**GPIO P2**	**CAN 2**	**UART 2**	**Trace Packet Bit**
		Direction					IO	I			IO	O	O	O

09	0x09	Signal					P1[09]	ENET_RXD0			P2[09]	USB_CONNECT1	RXD2	EXTIN0
		Module					**GPIO P1**	**RMII/MII**			**GPIO P2**	**USB P1**	**UART 2**	**External Trigger Inputt 0**
		Direction					IO	I			IO	O	I	I
10	0x0A	Signal	P0[10]	TXD2	SDA2	MAT3[0]	P1[10]	ENET_RXD1						
		Module	**GPIO P0**	**UART 2**	**I²C 2**	**Timer 3**	**GPIO P1**	**RMII/MII**						
		Direction	IO	O	IO	O	IO	I						
11	0x0B	Signal	P0[11]	RXD2	SCL2	MAT3[1]	P1[11]	ENET_RXD2	MCIDAT2	PWM0[6]	P2[11]	EINT1	MCIDAT1	I2STX_CLK
		Module	**GPIO P0**	**UART 2**	**I²C 2**	**Timer 3**	**GPIO P1**	**RMII/MII**	**SD/MMC**	**PWM 0**	**GPIO P2**	**External Interrupt 1**	**SD/MMC**	**I²S Bus**
		Direction	IO	I	IO	O	IO	I	IO	O	IO	I	IO	IO
12	0x0C	Signal	P0[12]	USB_PPWR2	MISO1	AD0[6]	P1[12]	ENET_RXD3	MCIDAT3	PCAP0[0]	P2[12]	EINT2	MCIDAT2	I2STX_WS
		Module	**GPIO P0**	**USB P2**	**SSP 1**	**ADC 0**	**GPIO P1**	**RMII/MII**	**SD/MMC**	**PWM 0**	**GPIO P2**	**External Interrupt 2**	**SD/MMC**	**I²S Bus**
		Direction	IO	O	IO	I	IO	I	IO	I	IO	I	IO	IO
13	0x0D	Signal	P0[13]	USB_UP_LED2	MOSI1	AD0[7]	P1[13]	ENET_RX_DV			P2[13]	EINT3	MCIDAT3	I2STX_SDA
		Module	**GPIO P0**	**USB P2**	**SPP 1**	**ADC 0**	**GPIO P1**	**RMII/MII**			**GPIO P2**	**External Interrupt 3**	**SD/MMC**	**I²S Bus**
		Direction	IO	O	IO	I	IO	I			IO	I	IO	IO
14	0x0E	Signal	P0[14]	USB_HSTEN2	USB_CONNECT2	SSEL1	P1[14]	ENET_RX_ER						
		Module	**GPIO P0**	**USB P2**	**USB 2**	**SSP 1**	**GPIO P1**	**RMII/MII**						
		Direction	IO	O	O	IO	IO	I						
15	0x0F	Signal	P0[15]	TXD1	SCK0	SCK	P1[15]	ENET_REF_CLK/ENET_RX_CLK						
		Module	**GPIO P0**	**UART 1**	**SSP 0**	**SPI**	**GPIO P1**	**RMII/MII**						
		Direction	IO	O	IO	IO	IO	I						
16	0x10	Signal	P0[16]	RXD1	SSEL0	SSEL	P1[16]	ENET_MDC						
		Module	**GPIO P0**	**UART 1**	**SSP 0**	**SPI**	**GPIO P1**	**RMII/MII**						
		Direction	IO	I	IO	IO	IO	O						
17	0x11	Signal	P0[17]	CTS1	MISO0	MISO	P1[17]	ENET_MDIO						
		Module	**GPIO P0**	**UART 1**	**SSP 0**	**SPI**	**GPIO P1**	**RMII/MII**						
		Direction	IO	I	IO	IO	IO	IO						
18	0x12	Signal	P0[18]	DCD1	MOSI0	MOSI	P1[18]	USB_UP_LED1	PWM1[1]	CAP1[0]				
		Module	**GPIO P0**	**UART 1**	**SSP 0**	**SPI**	**GPIO P1**	**USB P1**	**PWM 1**	**Timer 1**				
		Direction	IO	I	IO	IO	IO	O	O	I				

19	0x13	Signal	P0[19]	DSR1	MCICLK	SDA1	P1[19]	USB_TX_E1	USB_PPWR1	CAP1[1]				
		Module	**GPIO P0**	**UART 1**	**SD/MMC**	**I²C 1**	**GPIO P1**	**USB P1**	**USB 1**	**Timer 1**				
		Direction	IO	I	O	IO	IO	O	O	I				
20	0x14	Signal	P0[20]	DTR1	MCICMD	SCL1	P1[20]	USB_TX_DP1	PWM1[2]	SCK0				
		Module	**GPIO P0**	**UART 1**	**SD/MMC**	**I²C 1**	**GPIO P1**	**USB P1**	**PWM 1**	**SSP 0**				
		Direction	IO	O	IO	IO	IO	O	O	IO				
21	0x15	Signal	P0[21]	RI1	MCIPWR	RD1	P1[21]	USB_TX_DM1	PWM1[3]	SSEL0				
		Module	**GPIO P0**	**UART 1**	**SD/MMC**	**CAN 1**	**GPIO P1**	**USB P1**	**PWM 1**	**SSP 0**				
		Direction	IO	I	O	I	IO	O	O	IO				
22	0x16	Signal	P0[22]	RTS1	MCIDATA0	TD1	P1[22]	USB_RCV1	USB_PWRD1	MAT1[0]				
		Module	**GPIO P0**	**UART 1**	**SD/MMC**	**CAN 1**	**GPIO P1**	**USB P1**	**USB P1**	**Timer 1**				
		Direction	IO	0	IO	O	IO	I	I	O				
23	0x17	Signal	P0[23]	AD0[0]	I2SRX_CLK	CAP3[0]								
		Module	**GPIO P0**	**ADC 0**	**I²S Bus**	**Timer 3**								
		Direction	IO	I	IO	I								
24	0x18	Signal	P0[24]	AD0[1]	I2SRX_WS	CAP3[1]								
		Module	**GPIO P0**	**ADC 0**	**I²S Bus**	**Timer 3**								
		Direction	IO	I	IO	I								
25	0x19	Signal	P0[25]	AD0[2]	I2SRX_SDA	TXD3								
		Module	**GPIO P0**	**ADC 0**	**I²S Bus**	**UART 3**								
		Direction	IO	I	IO	O								
26	0x1A	Signal	P0[26]	AD0[3]	AOUT	RXD3								
		Module	**GPIO P0**	**ADC 0**	**DAC**	**UART 3**								
		Direction	IO	I	O	I								
27	0x1B	Signal					P1[27]	USB_INT1	USB_OVRCR1	CAP0[1]				
		Module					**GPIO P1**	**USB P1**	**USB P1**	**Timer 0**				
		Direction					IO	I	I	I				
28	0x1C	Signal					P1[28]	USB_SCL1	PCAP1[0]	MAT0[0]				
		Module					**GPIO P1**	**USB P1**	**PWM 1**	**Timer 0**				
		Direction					IO	IO	I	O				

29	0x1D	Signal	P0[29]	AD0[3]	AOUT	RXD3	P1[29]	USB_SDA1	PCAP1[1]	MAT0[1]				
		Module	**GPIO P0**	**ADC 0**	**DAC**	**UART 3**	**GPIO P1**	**USB P1**	**PWM 1**	**Timer 0**				
		Direction	IO	I	O	I	IO	IO	I	O				
30	0x1E	Signal	P0[30]	USB_D+1			P1[30]	USB_PWRD2	V_{BUS}	AD0[4]				
		Module	**GPIO P0**	**USB P1**			**GPIO P1**	**USB P2**	**USB Power**	**ADC**				
		Direction	IO	IO			IO	I	I	I				
31	0x1F	Signal	P0[31]	USB_D-1										
		Module	**GPIO P0**	**USB P1**										
		Direction	IO	IO										

Tab. A.6: LPC-2468-Stick Portzuweisung für das SECURITYv2.1-Board

Peripheral/Communication Module														LPC-2468 Stick		LPC-COM Board		LPC-PROTO Board	
Standard						Port	Optional							Connector		Connector		Connector	
Module		PINSEL					Module			PINSEL									
Type	Description	No.	DIR	Code	Signal	Pin	Type	Description	No.	DIR	Code	Signal		Type	Pin	Type	Pin	Type	Pin
ADC	Analog Digital Converter	1	→	01	AD0[0]	P0.23	TIMER	Hardware Timer	3	→	11	CAP3[0]			48		27		15
		2			AD0[1]	P0.24						CAP3[1]			46		25		16
		3			AD0[2]	P0.25	UART	TIA/EIA	3	←	11	TXD3			50		23		14
DAC	Digital Analog Convter		←	10	AOUT	P0.26				→		RXD3			52	X501	21		13
I²C	Fast Fieldbus	1	⇆	11	SDA1	P0.19									30		31	X202	22
					SCL1	P0.20									28		29		23
UART	TIA/EIA RS-232	2	←	01	TXD2	P0.10									24	X400	2		25
			→		RXD2	P0.11									26		3		24
CAN	Fieldbus	1	→	11	RD1	P0.21									37		34	X201	20
			←		TD1	P0.22									35		32		21
SPI	Serial Peripheral Interface	0	I←		SCK	P0.15									42		39		17
					SSEL	P0.16									40		37		18
			→I		MISO	P0.17									36		35		19
			I←		MOSI	P0.18								PCM80	38		33	X202	20
COMM	System Ready	0	↓	00	RDY_UC	P2.13									16		10		28
	4 Phase Handshake	1	I←		REQ	P0.04									74		30		3
			→I	01	ACK	P2.11	EINT	External Interrupt	1	→	01	EINT1			20		6		26
			I←	00	RW	P1.11									45	X501	38	X201	16
	Affermation	1	→I	01	RESULT	P2.12	EINT	External Interrupt	2	→	01	EINT2			18		8		27
	Data	1	⇆I	00	DATA[0]	P2.00	PWM	Pulse Width Modulation	1	←	01	PWM1[1]			70		26		5
					DATA[1]	P2.01			2			PWM1[2]			62		24		9
					DATA[2]	P2.02			3			PWM1[3]			68		22		6
					DATA[3]	P2.03			4			PWM1[4]			64		20	X202	8
					DATA[4]	P2.04			5			PWM1[5]			66		18		7
					DATA[5]	P2.05			6			PWM1[6]			60		16		10
					DATA[6]	P2.08									58		14		11
					DATA[7]	P2.09									56		12		12

DIR Legend

- I← LPC to NEXYS2
- →I NEXYS2 to LPC
- ⇆I LPC/NEXYS2
- ↓ To control the SECURITY
- ← Transmit
- → Receive
- ⇆ Transmit/Receive

					DATA[8]	P0.05								72		28		4
					DATA[9]	P1.12								47		36	X201	15
	Destination	1	⇆	00	ADDR[0]	P1.07								43		40		17
					ADDR[1]	P1.20								12		41	X202	30
					ADDR[2]	P1.21								14		42		29
Free/available pins													**30/62**		**0/30+(2)**		**30/62**	

Tab. A.7: LPC-2468-Stick Kontakte

Features		LPC-2468 Stick		LPC-COM Board		LPC-PROTO Board	
		Σ		Σ		Σ	
Description	Connector/Module	Pin	Max. Block	Pin	Max. Block	Pin	Max. Block
Ports	Port 0	24	14	14	12	24	14
	Port 1	27	23	5	2	27	23
	Port 2	11	6	11	8	11	6
System	Pmod	-	-	-	-	30	16
DC Power Supply	+3V3	2	-	2	-	7	2
	+5V0	2	-	2	-	5	2
	+9V0	-	-	2	-	-	-
	GND	8	-	5	-	9	4
	GNDA	2	-	-	-	2	2
System	Control	4	-	4	-	4	-
Number of pins		62		30		62	

A.4 LPC-COM-Board

Tab. A.8: Pinning des LPC-COM-Boards

Connector																	
X501			X502 (SD-Slot)			X400 (RS232 female)			X301 (USB-B)			X301 (USB-A)			X200 (Ethernet)		
*PCM80 Pin(s)	Pin	Function	*PCM80 Pin(s)	Pin	Function	*PCM80 Pin(s)	Pin	Function	*PCM80 Pin(s)	Pin	Function	*PCM80 Pin(s)	Pin	Function	*PCM80 Pin(s)	Pin	Function
*A	**1**	GND															
	2	CAN_BM															
*A	**3**	GND															
	4	CAN_BP															
	5	+9V0															
20	**6**	P2.11															
	7	+9v0															
18	**8**	P2.12															
79, 80	**9**	+5V0															
16	**10**	P2.13															
79, 80	**11**	+5V0															
56	**12**	P2.09															
1, 2	**13**	+3V3															
58	**14**	P2.08															
1, 2	**15**	+3V3															
60	**16**	P2.05															
*A	**17**	GND															
66	**18**	P2.04															
*A	**19**	GND															
64	**20**	P2.03															
52	**21**	P0.26															
68	**22**	P2.02															
50	**23**	P0.25															
62	**24**	P2.01															
46	**25**	P0.24															
70	**26**	P2.00															
48	**27**	P0.23															
72	**28**	P0.05															
28	**29**	P0.20															
74	**30**	P0.04															
30	**31**	P0.19															
35	**32**	P0.22															
38	**33**	P0.18															
14	**34**	P0.21															
36	**35**	P0.17															
47	**36**	P1.12															
40	**37**	P0.16															
45	**38**	P1.11															
42	**39**	P0.15															
43	**40**	P1.07															
12	**41**	P1.20															
14	**42**	P1.21															
*A	**43**	GND															
4	**44**	PWREN_N															
*A	**45**	GND															

3	**46**	RESET_N															
	47																
	48																

*PCM80 = X500
*A Pins: 10, 21, 22, 33, 44, 53, 54, 79

A.5 LPC-PROTO-Board

Tab. A.9: Pinning des LPC-PROTO-Boards

Connector																		
X201			X202			X203			X204			X205			X206		X207	
PCM80 Pin(s)	Pin	Function	PCM80 Pin(s)	Pin	Function	PCM80 Pin(s)	Pin	Function	PCM80 Pin(s)	Pin	Function	PCM80 Pin(s)	Pin	Function	Pin	Function	Pin	Function
77	**1**		79, 80	**1**	+5V0	79, 80	**1**	+5V0	79, 80	**1**	+5V0	79, 80	**1**	+5V0	**1**	Pin 2	**1**	Pin 2
75	**2**	P1.00	79, 80	**2**	+5V0	1, 2	**2**	+3V3	1, 2	**2**	+3V3	1, 2	**2**	+3V3	**2**	Pin 1	**2**	Pin 1
73	**3**	P1.04	74	**3**	P0.04	4	**3**	PWREN_N	34, 54, 76	**3**	GND	78	**3**	(P0.12)	**3**	Pin 4	**3**	Pin 4
71	**4**	P1.01	72	**4**	P0.05	1, 2	**4**	+3V3				1, 2	**4**	+3V3	**4**	Pin 3	**4**	Pin 3
69	**5**	P1.09	70	**5**	P2.00	*A	**5**	GND				*A	**5**	GND	**5**	Pin 6	**5**	Pin 6
67	**6**	P1.14	68	**6**	P2.02	*A	**6**	GND				*A	**6**	GND	**6**	Pin 5	**6**	Pin 5
65	**7**	P1.15	66	**7**	P2.04	*A	**7**	GND							**7**	Pin 8	**7**	Pin 8
63	**8**	P1.17	64	**8**	P2.03	*A	**8**	GND							**8**	Pin 7	**8**	Pin 7
61	**9**	P1.05	62	**9**	P2.01	32	**9**	GNDA							**9**	Pin 10	**9**	Pin 10
59	**10**	P1.06	60	**10**	P2.05	32	**10**	GNDA							**10**	Pin 9	**10**	Pin 9
57	**11**	P1.13	58	**11**	P2.08										**11**	Pin 12	**11**	Pin 12
55	**12**	P1.16	56	**12**	P2.09										**12**	Pin 11	**12**	Pin 11
51	**13**	P1.10	52	**13**	P0.26										**13**	Pin 14	**13**	Pin 14
49	**14**	P1.08	50	**14**	P0.25										**14**	Pin 13	**14**	Pin 13
47	**15**	P1.12	48	**15**	P0.23										**15**	Pin 16		
45	**16**	P1.11	46	**16**	P0.24										**16**	Pin 15		
43	**17**	P1.07	42	**17**	P0.15													
41	**18**	P1.03	40	**18**	P0.16													
39	**19**	P1.02	38	**19**	P0.18													
37	**20**	P0.21	36	**20**	P0.17													
35	**21**	P0.22	34	**21**														
31	**22**	P1.18	30	**22**	P0.19													
29	**23**	P0.30	28	**23**	P0.20													
27	**24**	P1.30	26	**24**	P0.11													
25	**25**	P1.29	24	**25**	P0.10													

23	**26**	P1.28	20	**26**	P2.11													
19	**27**	P1.27	18	**27**	P2.12													
17	**28**	P1.22	16	**28**	P2.13													
15	**29**	P1.19	14	**29**	P1.21													
13	**30**	P0.14	12	**30**	P1.20													
11	**31**	P0.29	8	**31**	P0.01													
9	**32**	P0.13	6	**32**	P0.00													
7	**33**	P0.31	4	**33**	PWREN_N													
5	**34**	USB_D-2	*A	**34**	GND													
3	**35**	RESET_N	1, 2	**35**	+3V3													
*A	**36**	GND	1, 2	**36**	+3V3													

*A Pins: 10, 21, 22, 33, 44, 53, 54, 79

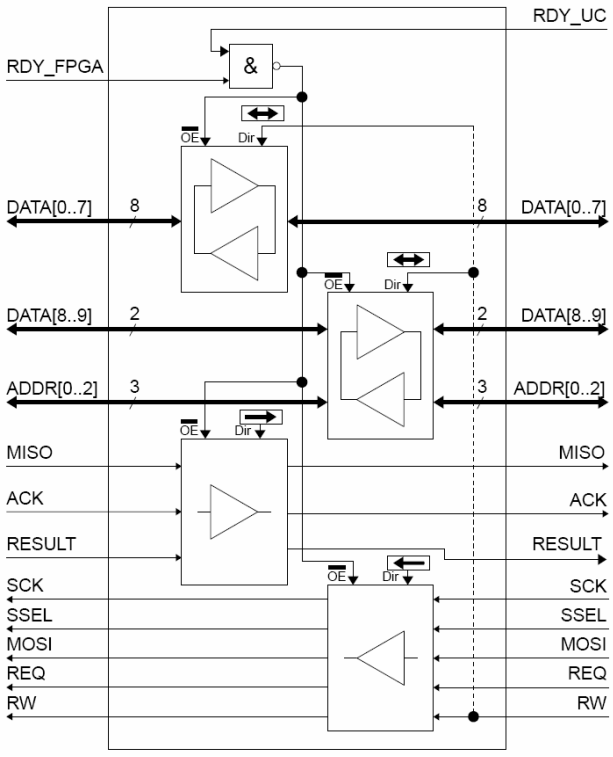
Die beschriebenen Funktionen der Konnektoren: X206 und X207 entsprechen den jeweiligen Pins des Konnektors.

B. Schalt-, Logik- und Bestückungspläne

B.1 SECURITYv1-Board

Das Schaltungskonzept des SECURITYv1-Boards sieht eine Kabelgebundene Verdrahtung Signalverbindungen, zwischen den Hardware-Komponenten FPGA und µController für die Peripherie- und Software-Module PSI und SPI vor. Die Aktivierung der Sicherheitsschaltung erfolgt über die Pegelsteuerung der Signale: RDY_FPGA und RDY_UC, welche nach der Bootsequenz der SoC-Chips und Initialisierung des user codes - zwingend als erste Programmanweisung - auf high gesetzt werden. Während des Startvorgangs der Hardware-Komponenten (Bootsequenz: FPGA ca. 270µs und µController ca. 320µs - Programm-Ladevorgang abhängig vom Datenvolumen [XILINX, HITEX]) oder eines Resets sind alle Bustreiber im Zustand 3-State. Die Signalflussrichtung der bidirektionalen Signale: DATA[0..7] und ADDR[0..2] wird von der Master-Anwendung über das Signal: RW zur Laufzeit geschaltet. In dieser Kooperation zwischen dem FPGA als Hardware-Beschleuniger und dem µController, arbeitet dieser als Mastereinheit. Das FPGA basiert als Slave.

FPGA



C

Das modular aufgebaute SECURITYv1-Board ist eine Komponente des zukünftigen Development-Boards NEXSELv1. Das SECURITYv1-Board verfügt über vier konfigurierbare highspeed Sende-/Empfangseinheiten mit 8fach bidirektionalen Schnittstellen und Tri-State Modus. Der M3-Sockel ist für die mechanische Aufnahme des LPC-COM-Boards oder LPC-PROTO-Boards. Je zwei LSTTL genormte DIP14 IC-Sockel zur Signalanpassung, Signalverteilung mit frei wählbarer Blockgröße und prellfreiem LVTTL/TTL Taster/Schalter. Statische Kodierung der ICs über Kurzschlussstecker. Alle Signale sind mess- und verteilbar über die vorhandenen Buchsenleisten. Die Netzversorgung (VU und VCC) des auf dem M3-Sockel befestigten Kommunikations- und Erweiterungs-Boards erfolgt über die Konnektoren des SECURITYv1-Boards (vgl. Anhang A).

Schalt- und Bestückungspläne:

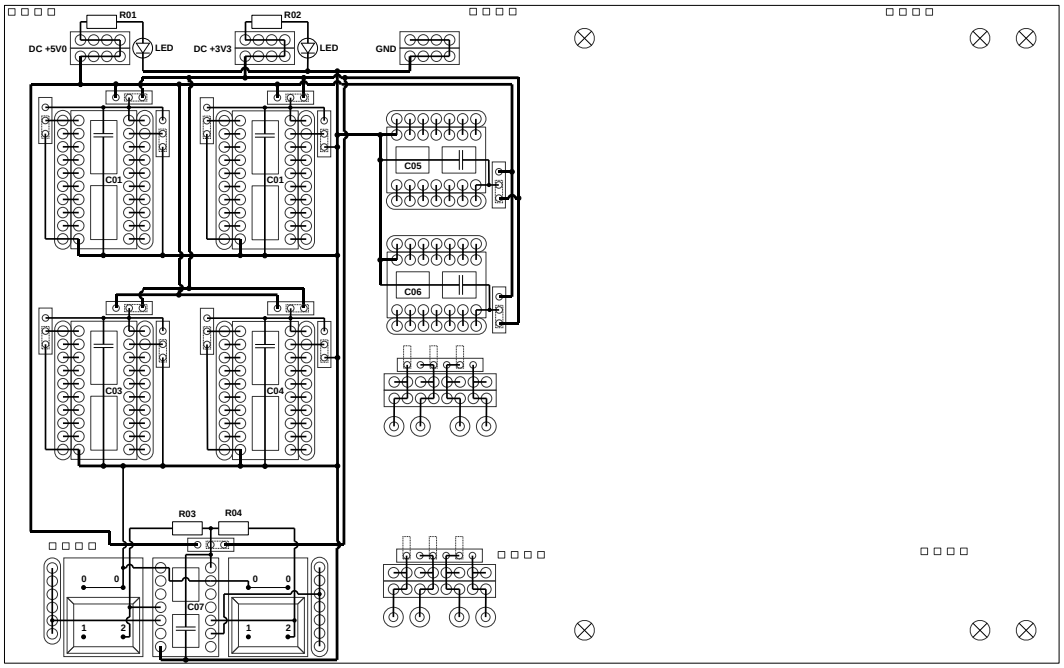


Abb. B.1: Schaltplan des SECURITYv1-Board

Bohrlöcher	⦸ innen 3,5 mm	
KNF 30 Muttern	M3, H: 1,5 mm	Richco
Schrauben	M3, 16 mm, PVC, Philips, Flachkopf	Richco
Unterlegscheiben	M3, 0,1 mm, PVC, Form A	
	M3, 0,5 mm, Edelstahl, Form A, DIN 125	DURATOOL
Distanzbolzen	M3, 20 mm, PVC, Innengewinde beidseitig – Platinen Isolatoren	DURATOOL
	M3, 20 mm, Edelstahl	DURATOOL
R01	820 Ω ± 10%, ¼ W, E12, axial, MS	Tyco
R02	390 Ω ± 10%, ¼ W, E12, axial, KS	Phoenix
R03/4	430 Ω ± 5%, ⅛ W, E24, axial, MS	Tyco
C01..07	100 nF ± 10%, DC 50V0, X7R	EPCOS
LED	Max. 4 mA (DC 5V0), SSL 1,8 mm, R: 2,54 mm	Kingbright
IC1..4 Sockel	20-pin, DIP 2,54 mm, ggf. mit C 100 nF	HARWiN
IC5..7 Sockel	14-pin, DIP 2,54 mm, ggf. mit C 100 nF	HARWiN
Taster 1,2	Serie K, DC 20V0, 20 mA, Kontaktprellen < 1 ms Pins 0-1: offene Verbindung Pins 0-2: geschlossene Verbindung	ITT

Name	Bezeichnung	Funktion	Pins	Typ	Hersteller
IC1..4	SN74HCT245N	8x BUS Transceivers	20-pin	DIP	Texas Instruments
IC5	74HC00N	4x 2-NAND	14-pin	DIP	NXP
IC6	74HCU04N	6x Inverter	14-pin	DIP	Philips Semiconductors
IC7	SN74HC74N	2x D-FF, pos. e. trig.	14-pin	DIP	Texas Instruments

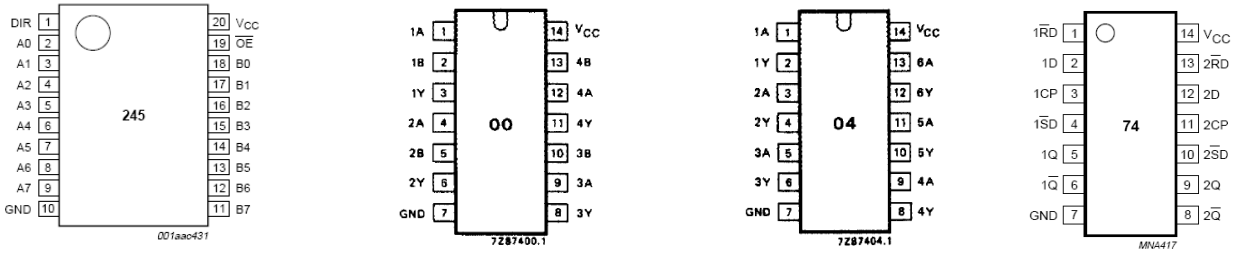


Abb. B.2: SECURITYv1-Board Hardware

Tab. B.1: SECURITYv1-Board ICs

Max. gemessene Umschaltfrequenz* $T = f(t_{PHL\_max}/t_{PLH\_max})$	SN74HCT245N	74HC00N	74HCU04N	SN74HC74N
$f_{max} = 1/T$	20,07 MHZ	20,03 MHz	19,74 MHz	20,05 MHz

Die BUS-Transceiver können mit einer maximalen Impulsfrequenz von 40MHz betrieben werden (vgl. Messergebnisse). Im FastGPIO-Modus des ARM7™ TDMI wird die Kommunikationsfrequenz der µController Peripherie von ~30MHz nicht von den Bustreibern beschränkt. Unabhängig der Signalflussrichtung: FPGA (B) ⇨ µC (A) oder FPGA (B) ⇦ µC (A) sind die Transfer- und Umschaltzeiten ~7,0ns mit VCC 3,3V und ~5,0ns bei VU 5,0V. Die Signallaufzeitverzögerung der Festkörperschaltkreise beträgt 9,5ns.

Messungen:

Alle Messungen wurden mit dem nachfolgend aufgeführten Equipment durchgeführt.

Funktionsgenerator: KONTRON 8020 (max. 20MHz)
V_{Out}: DC 6,60V
Signalform: Rechtecksignal (Tastgrad: 50% Voreinstellung)
Messleitung: Koaxialkabel RG-56 (flexibel) 1m, BNC-B/BNC-B
Oszilloskop: Tektronix TDS 544 A
Tastköpfe: Tektronix P6139 A (10fach, max. 500MHz, 10MΩ/8pF) 1 m
Durchgangswiderstand: Philips Semiconductors HZ 22 50Ω (Spannungsteiler: 2:1) BNC-A/BNC-B
BNC F: PONOMA BNC-A/BNC-A
Messadapter: BNC-B/ 2-pin 0,635 mm, 10cm
Stromversorgungsgerät: Rohde & Schwarz NGT 20 (DC 0..6..20V, 0..1..5A)

IC1..4: 8fach BUS-Transceivers SN74HCT245N

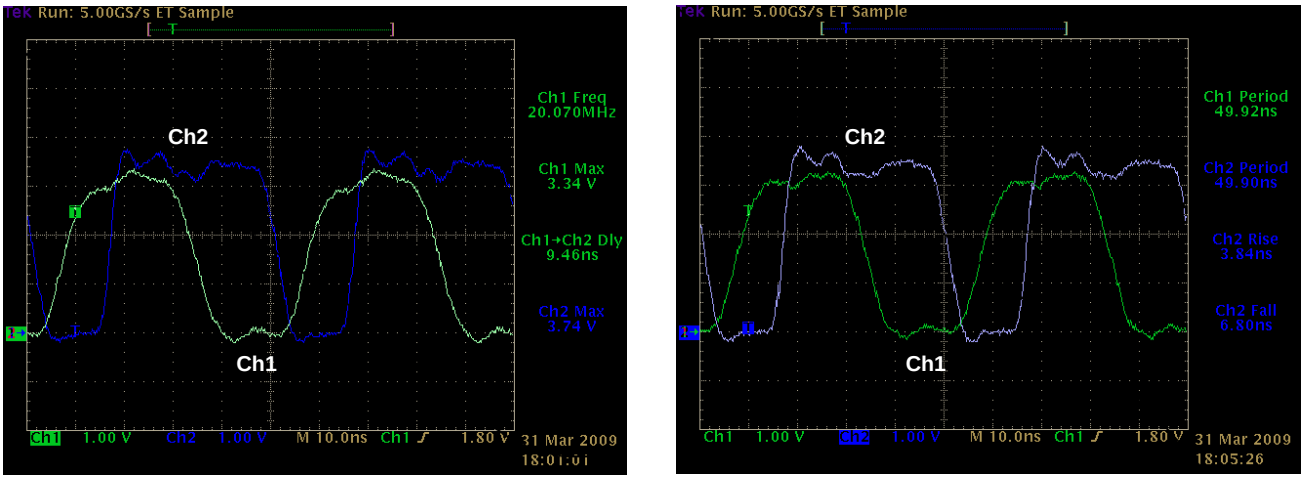


Abb. B.3: Latenzzeit und Spannungspegel der Bustreiber ICs

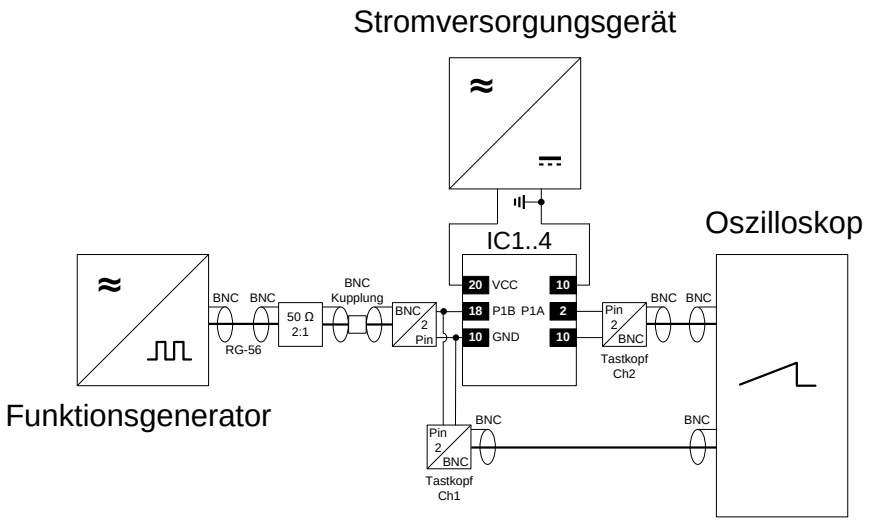


Abb. B.4: Messaufbau Bustreiber ICs

Verstärkung: $$\upsilon_U = \frac{U_{P1A}}{U_{P1B}} = \frac{3{,}74V}{3{,}34V} \approx 112\%$$

IC5: 4x 2-NAND 74HC00N

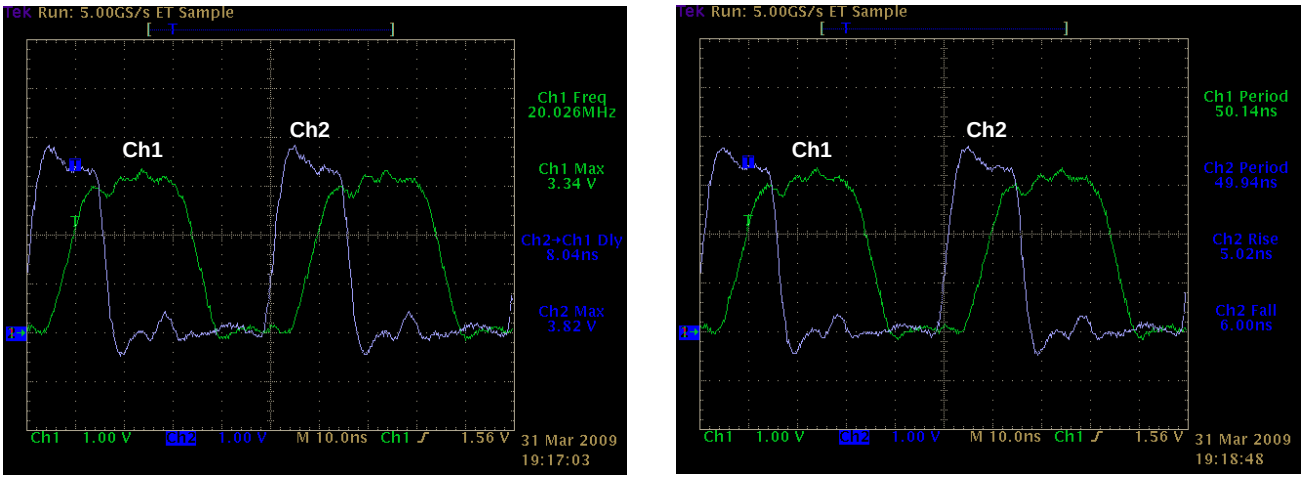


Abb. B.5: Latenzzeit und Spannungspegel der NAND ICs

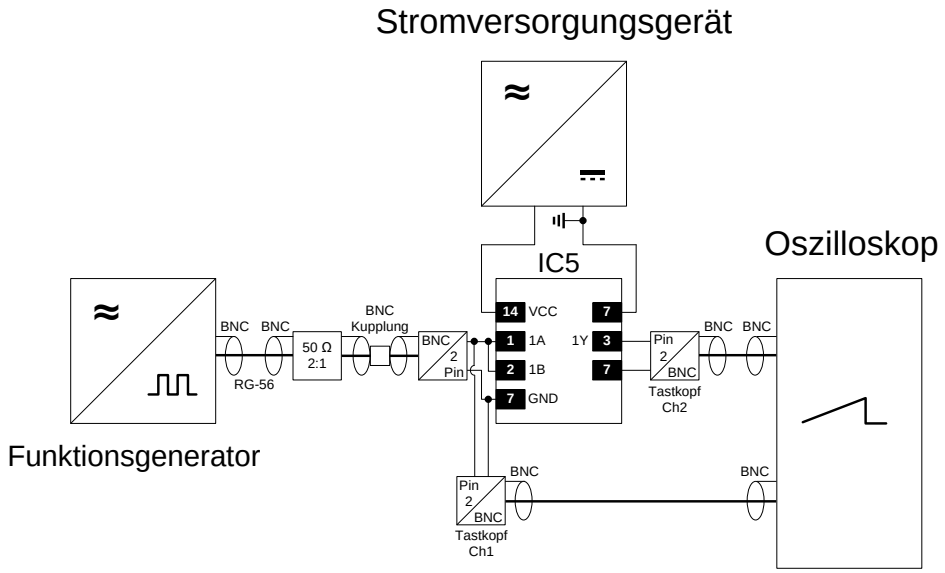


Abb. B.6: Messaufbau NAND ICs

Verstärkung: $\upsilon_U \approx 114\%$

IC6: 6x Inverter 74HCU04N

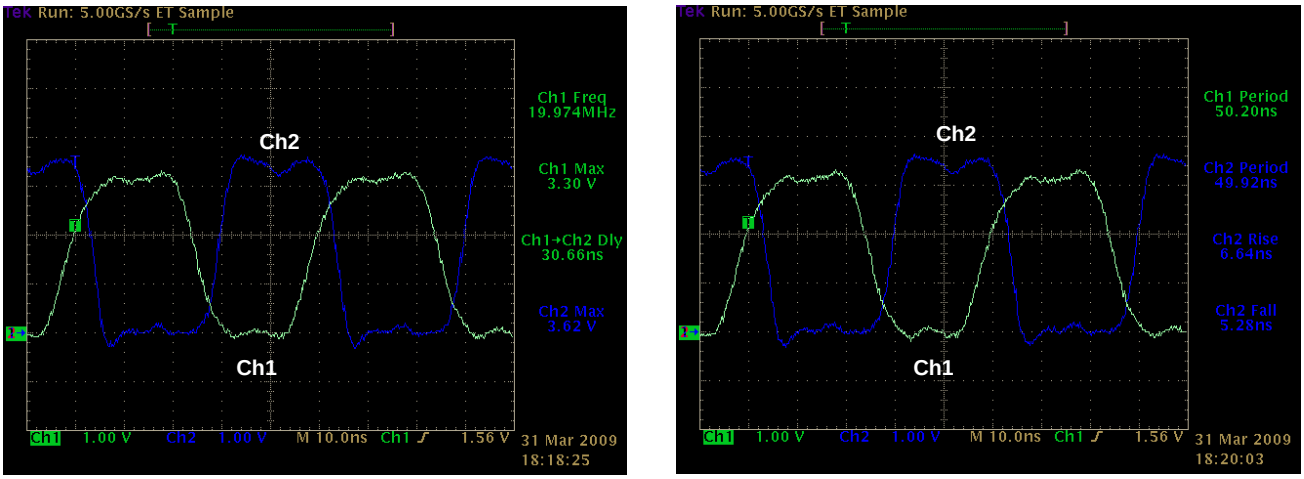


Abb. B.7: Latenzzeit und Spannungspegel der Inverter ICs

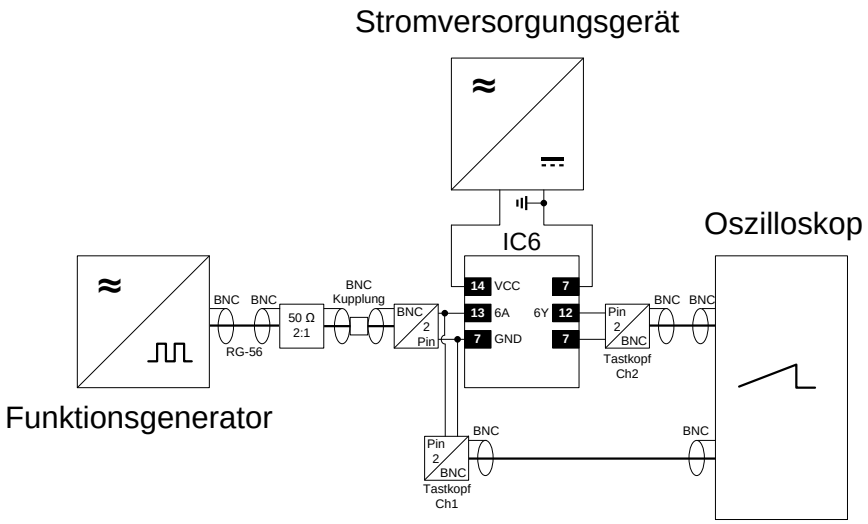


Abb. B.8: Messaufbau Inverter ICs

Verstärkung: $\upsilon_U \approx 110\%$

IC7: 2x D-FF SN74HC74N

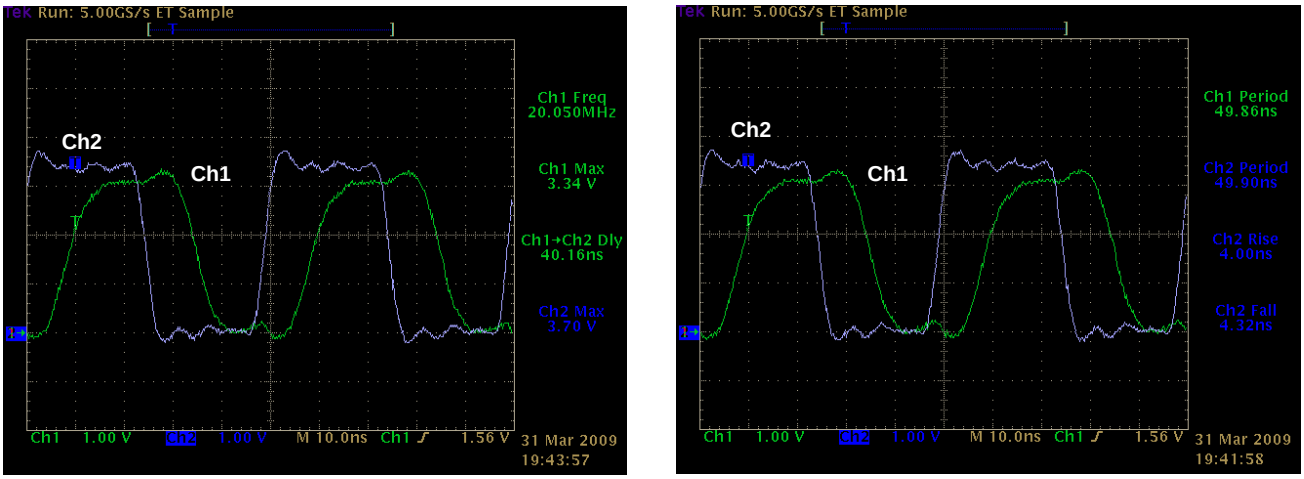


Abb. B.9: Latenzzeit und Spannungspegel der D-FF ICs

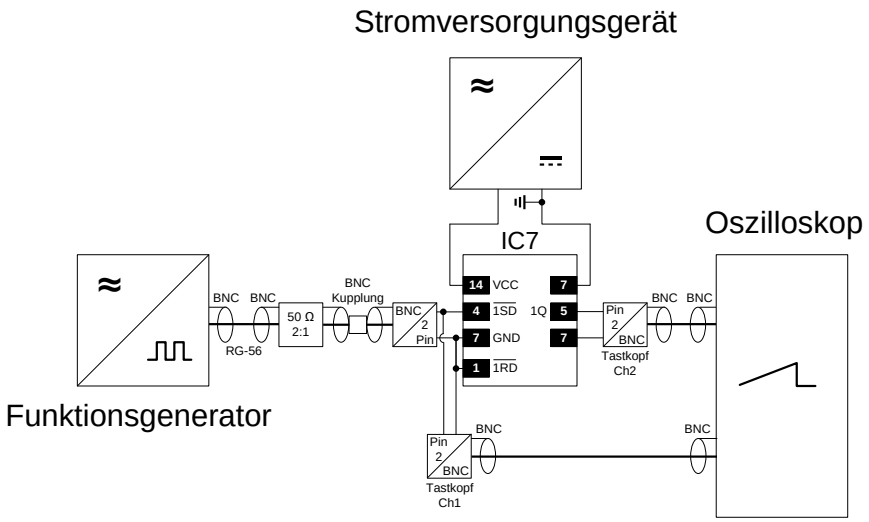


Abb. B.10: Messaufbau D-FF ICs

Verstärkung: $\upsilon_U \approx 108\%$

Spezifikationen:

Platine

Typ:	Laborkarte
Material:	Epoxy FR4, DIN 41612
Kontaktierung:	2x Layer (beidseitig, nicht durchkontaktiert)
Lochrasterabstand:	DOT R: 2,54 mm
Format:	Europakarte (160mm x 100mm x 1,6mm)
Höhe	34mm (ohne Distanzbolzen 13mm)

Bidirektionale Signalverbindungen

Gesamtanzahl:	32

JEDEC DIP14 Sockel

Anzahl:	2 (siehe 3. Funktion, Steuerung und Schnittstellen)

Schaltzustände (Taster: T1/2)

High/Low:	Konfigurierbar über IC6 (V_{out}: VU oder VCC über [J460])

Modifikationen und Erweiterung des SECURITYv1-Boards sind durch den Austausch der ICs 5/6 bedingt des DIN-Layouts möglich.

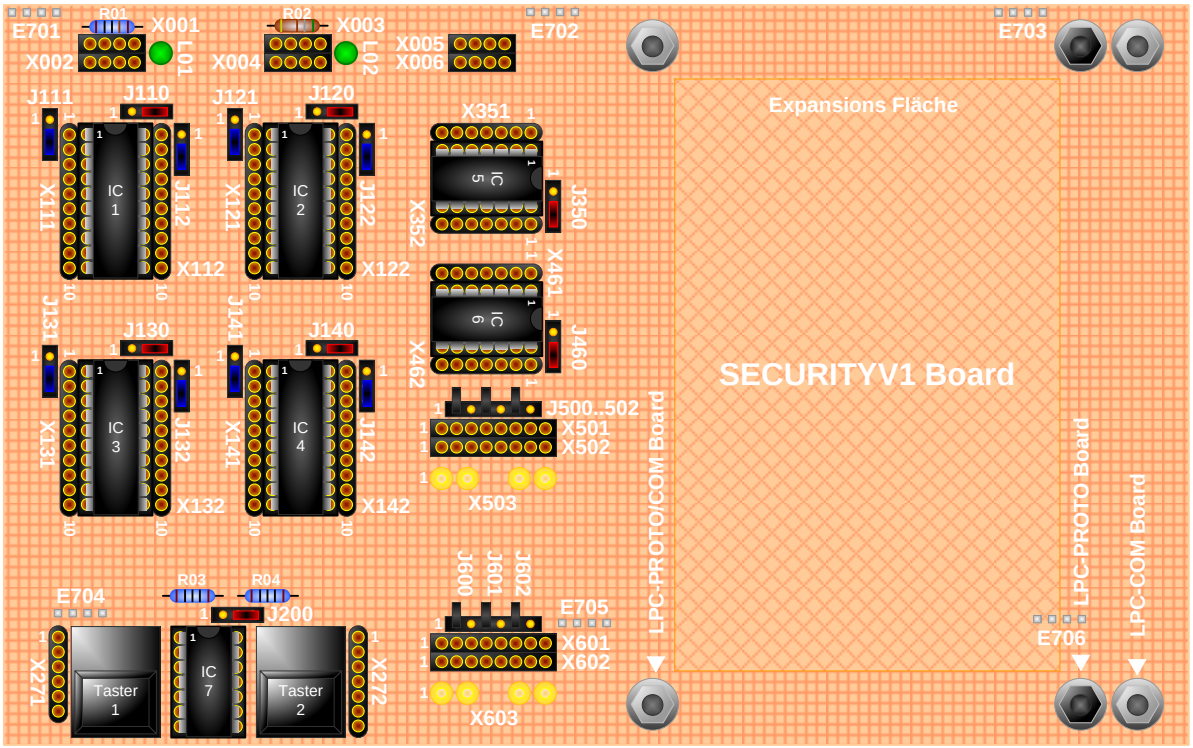


Abb. B.11: SECURITYv1-Board Platinengrafik

- Netzspannungsversorung über die BL [X001 oder X002] VU DC +5V0, [X003 oder X004] VCC DC +3V3 und [X005 oder X006] GND (nicht als GNDA verwendbar)
- Signal LEDs (grün) für VU [L01] und VCC [L02]
- Spannungsversorgung/-verteilerblöcke der ICs (VCC oder VU) einstellbar über die Jumper [J110, J120, J130, J140, J200, J350 und J460]
- 4x8 Uni-/Bidirektionale Signalanschlüsse (A/B) [X111/112, X121/122, X131/132 und X141/142] - HW kodierte Beschaltung/Funktion über die Kurzschlussstecker [J111/112, J121/122, J131/132 und J141/142]
- Signalinvertierung über IC5 [X351/352], IC6 [X461/462] und oder anderer Festkörperschaltkreise
- 2x TTL/LVTTL konforme prellfreie Taster [X271 und X272] – bei der Kombination mit dem LPC-COM Board ist ein Taster als RESET_N zu beschalten (siehe 9. Pinbelegung)
- 2x Signalverteilerblöcke 4x2..1x8 [X601/X602 und X501/X502] mit 2mm, rund 0,85mm und 0,56mm Buchsen (siehe 4. Schaltplan und Bestückung) - Blockgröße einstellbar über [J500..502 und J600..602]
- Sockelungen sind LSTTL Pin kompatibel und JEDEC übereinstimmend, zur Verwendung ggf. weiterer ICs
- Durchkontaktierte Expansionsfläche mit einem LR von 2,54mm
- 4-pin Expansionskonnektoren: [E701..706]
- M3-Sockel für die Entwickler Platinen: LPC-COM- und LPC-PROTO-Board

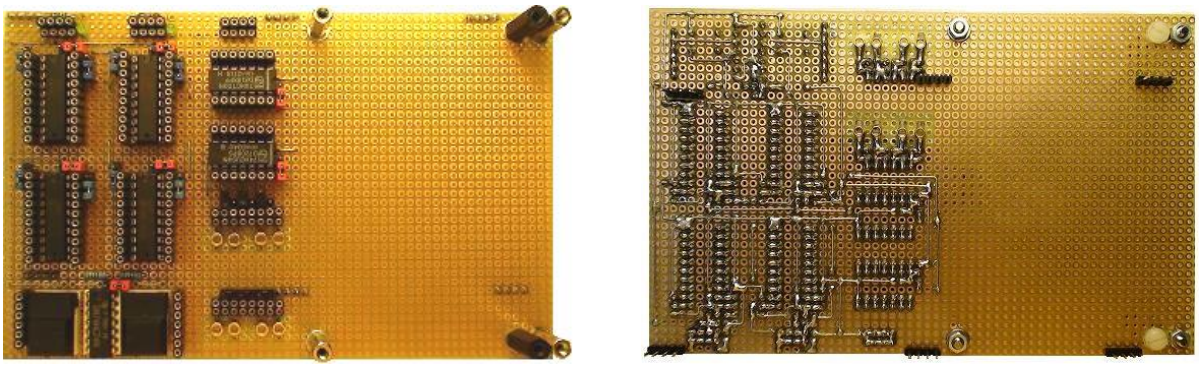


Abb. B.12: SECURITYv1-Board Platinen-Layout

Netzbetrieb:

Beim Anschluss der externen Netzversorgung an das SECURITYv1-Board über die Konnektoren (X002, X004 und X006), sind die vorgegeben Spannungswerte (max. VU) und die korrekte Beschaltung durch die Jumper-Einstellungen der Development-Boards zwingend einzuhalten (vgl. Funktionen, Steuerung und Schnittstellen des SECURITYv1-Boards). Für den fehlerfreien Betrieb der Festkörperschaltkreise und ggf. bei der Versorgung weiterer Komponenten (LPC-2468-Stick und LPC-COM-Board/LPC-PROTO-Board) ist die Strombegrenzung des Netzgerätes anzupassen.

Ø Stromaufnahme (Standby-Modus, DC 3V3 und 5V0)

Tab. B.2: Stromverbrauch der NEXSELv1-Board Komponenten

mA	**SECURITYV1 Board**	**NEXYS2 Board + FX2-BB Board**	**LPC-PROTO Board + LPC-2468 Stick**	**LPC-COM Board + LPC-2468 Stick**
SECURITYV1 Board	**73,30**	160,20	165,10	233,40
NEXYS2 Board + FX2-BB Board	160,20	**86,90**	178,70	247,00
LPC-PROTO Board + LPC-2468 Stick	165,10	178,70	**91,80**	251,90
LPC-COM Board + LPC-2468 Stick	233,40	247,00	251,90	**160,10**

Tab. B.3: Stromverbrauch der NEXSELv1-Board Komponentenkombinationen

mA	**NEXYS2 Board + FX2-BB Board**	**SECURITYV1 Board**	**LPC-PROTO Board + LPC-2468 Stick**	**LPC-COM Board + LPC-2468 Stick**
NEXSELV1 Board	**252,00**			
	320,30			

B.2 SECURITYv2.1-Board

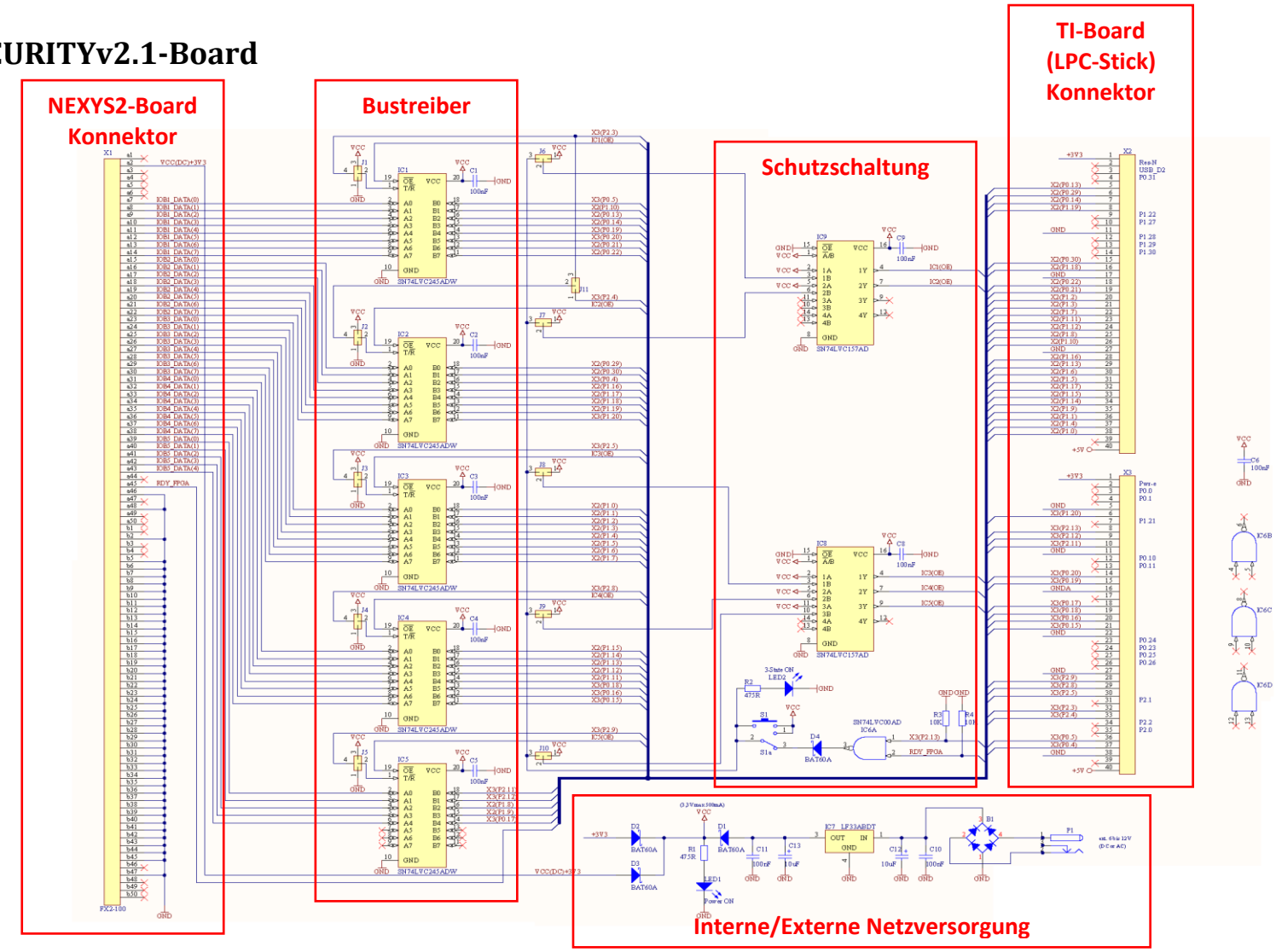


Abb. B.13: SECURITYv2.1-Board Schaltplan

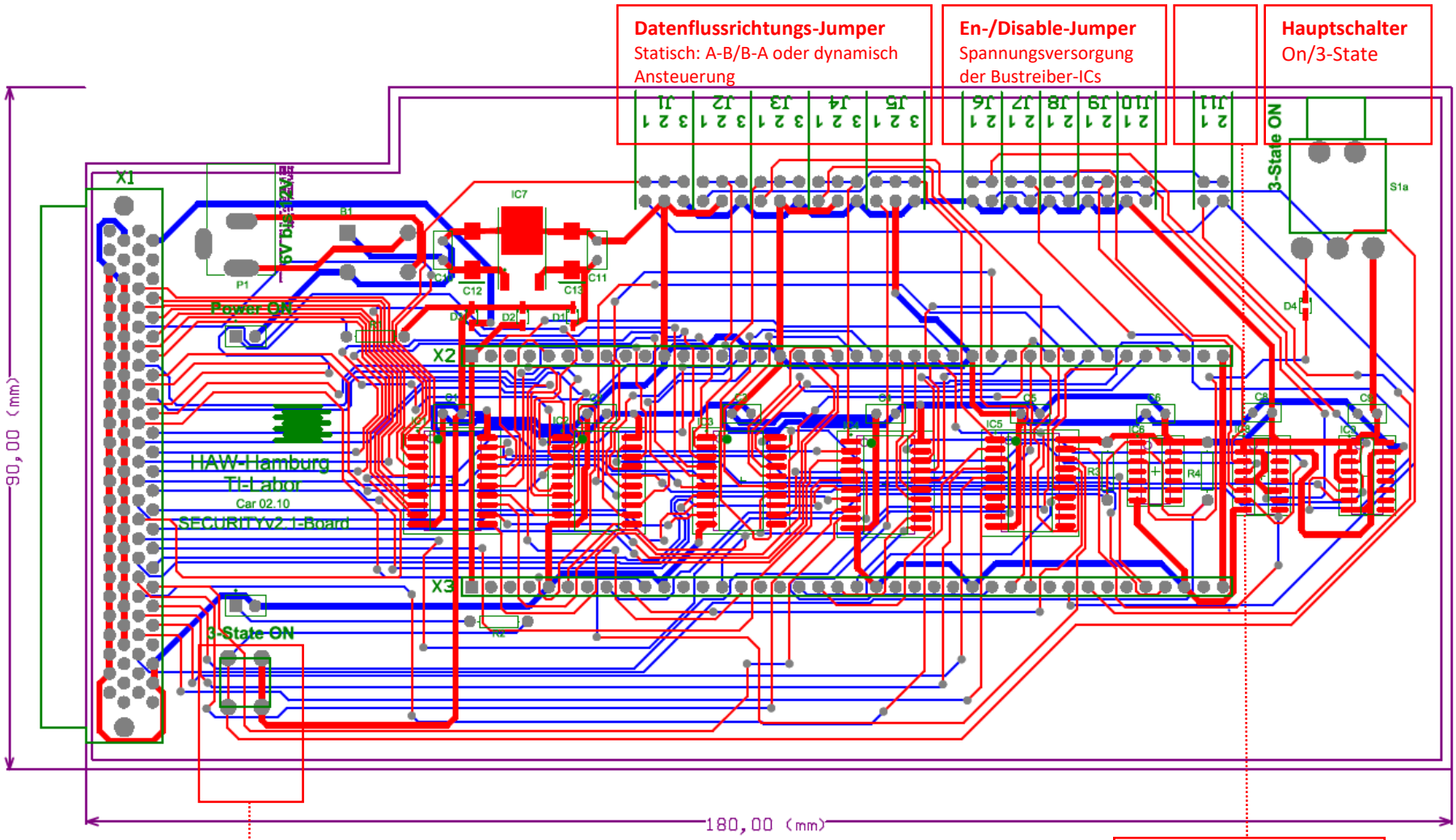


Abb. B.14: SECURITYv2.1-Board Platinen-Layout

SECURITYV2.1

Power Connector

X001		X002		X003		X004		X005		X006	
Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function
1	+5V0	1	+5V0	1	+3V3	1	+3V3	1	GND	1	GND
2	+5V0	2	+5V0	2	+3V3	2	+3V3	2	GND	2	GND
3	+5V0	3	+5V0	3	+3V3	3	+3V3	3	GND	3	GND
4	+5V0	4	+5V0	4	+3V3	4	+3V3	4	GND	4	GND

I/O Splitter

X501		X502		X503		X601		X602		X603	
Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function
1	X501 Pin2	1	X501 Pin2	1	X501 Pin1	1	X601 Pin2	1	X601 Pin2	1	X601 Pin1
2	X501 Pin1	2	X501 Pin1	2	X501 Pin3	2	X601 Pin1	2	X601 Pin1	2	X601 Pin3
3	X501 Pin4	3	X501 Pin4	3	X501 Pin6	3	X601 Pin3	3	X601 Pin3	3	X601 Pin6
4	X501 Pin3	4	X501 Pin3	4	X501 Pin8	4	X601 Pin4	4	X601 Pin4	4	X601 Pin7
5	X501 Pin6	5	X501 Pin6			5	X601 Pin6	5	X601 Pin6		
6	X501 Pin5	6	X501 Pin5			6	X601 Pin5	6	X601 Pin5		
7	X501 Pin8	7	X501 Pin8			7	X601 Pin8	7	X601 Pin8		
8	X501 Pin7	8	X501 Pin7			8	X601 Pin7	8	X601 Pin7		

Transceiver

IC1				IC2				IC3			
X111		X112		X121		X122		X131		X132	
A		B		A		B		A		B	
Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function
1	J111 Pin2	1	J110 Pin2	1	J121 Pin2	1	J120 Pin2	1	J131 Pin2	1	J130 Pin2
2	1A1	2	J112 Pin2	2	1A2	2	J122 Pin2	2	1A3	2	J132 Pin 2
3	2A1	3	1B1	3	2A2	3	1B2	3	2A3	3	1B3
4	3A1	4	2B1	4	3A2	4	2B2	4	3A3	4	2B3
5	4A1	5	3B1	5	4A2	5	3B2	5	4A3	5	3B3
6	5A1	6	4B1	6	5A2	6	4B2	6	5A3	6	4B3
7	6A1	7	5B1	7	6A2	7	5B2	7	6A3	7	5B3
8	7A1	8	6B1	8	7A2	8	6B2	8	7A3	8	6B3
9	8A1	9	7B1	9	8A2	9	7B2	9	8A3	9	7B3
10	GND	10	8B1	10	GND	10	8B2	10	GND	10	8B3

IC4			
X141		X142	
A		B	
Pin	Function	Pin	Function
1	J141 Pin2	1	J140 Pin2
2	1A4	2	J142 Pin2
3	2A4	3	1B4
4	3A4	4	2B4
5	4A4	5	3B4
6	5A4	6	4B4
7	6A4	7	5B4
8	7A4	8	6B4
9	8A4	9	7B4
10	GND	10	8B4

Logic

IC5				IC6			
X351		X352		X461		X462	
Pin	Function	Pin	Function	Pin	Function	Pin	Function
1	1A	1	J360 Pin2	1	1A	1	J460 Pin2
2	1B	2	4B	2	1Y	2	4A
3	1Y	3	4A	3	2A	3	4Y
4	2A	4	4Y	4	2Y	4	5A
5	2B	5	5B	5	3A	5	5Y
6	2Y	6	5A	6	3Y	6	6A
7	GND	7	5Y	7	GND	7	6Y

SECURITYV2.1																							
Power Connector												I/O Splitter											
X001		X002		X003		X004		X005		X006		X501		X502		X503		X601		X602		X603	
Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function	Pin	Function
1	+5V0	1	+5V0	1	+3V3	1	+3V3	1	GND	1	GND	1	X501 Pin2	1	X501 Pin2	1	X501 Pin1	1	X601 Pin2	1	X601 Pin2	1	X601 Pin1
2	+5V0	2	+5V0	2	+3V3	2	+3V3	2	GND	2	GND	2	X501 Pin1	2	X501 Pin1	2	X501 Pin3	2	X601 Pin1	2	X601 Pin1	2	X601 Pin3
3	+5V0	3	+5V0	3	+3V3	3	+3V3	3	GND	3	GND	3	X501 Pin4	3	X501 Pin4	3	X501 Pin6	3	X601 Pin3	3	X601 Pin3	3	X601 Pin6
4	+5V0	4	+5V0	4	+3V3	4	+3V3	4	GND	4	GND	4	X501 Pin3	4	X501 Pin3	4	X501 Pin8	4	X601 Pin4	4	X601 Pin4	4	X601 Pin7
												5	X501 Pin6	5	X501 Pin6			5	X601 Pin6	5	X601 Pin6		
												6	X501 Pin5	6	X501 Pin5			6	X601 Pin5	6	X601 Pin5		
												7	X501 Pin8	7	X501 Pin8			7	X601 Pin8	7	X601 Pin8		
												8	X501 Pin7	8	X501 Pin7			8	X601 Pin7	8	X601 Pin7		

Transceiver Power Jumper								Transceiver Direction Jumper				Transceiver Enable Jumper									
J110		J120		J130		J140		J111		J121		J131		J141		J112		J122		J132		J142	
Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]
1	+5V0	1	+5V0	1	+5V0	1	+5V0	1	A➡B/DC	1	A➡B/DC	1	A➡B/DC	1	A➡B/DC	1	Off/DC	1	Off/DC	1	Off/DC	1	Off/DC
2	IC1[20]	2	IC2[20]	2	IC3[20]	2	IC4[20]	2	IC1[01]	2	IC2[01]	2	IC3[01]	2	IC4[01]	2	IC1[19]	2	IC2[19]	2	IC3[19]	2	IC4[19]
3	+3V3	3	+3V3	3	+3V3	3	+3V3	3	A⬅B/GND	3	A⬅B/GND	3	A⬅B/GND	3	A⬅B/GND	3	On/GND	3	On/GND	3	On/GND	3	On/GND

I/O Splitter Jumper											
X501/X502						X601/X602					
J500		J501		J502		J600		J601		J602	
Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]
1	[1,2]	3	[3,4]	5	[5,6]	1	[1,2]	3	[3,4]	5	[5,6]
2	[3,4]	4	[5,6]	6	[7,8]	2	[3,4]	4	[5,6]	6	[7,8]

Logic Power Jumper				Pwr. Jmp.	
J350		J460		J200	
Pin	Function [Pin]	Pin	Function [Pin]	Pin	Function [Pin]
1	+5V0	1	+5V0	1	+5V0
2	IC5[14]	2	IC6[14]	2	IC7[14]
3	+3V3	3	+3V3	3	+3V3

Kurzschluss zwischen der SPI-Schnittstelle und dem SPI-Flash

Wenn IOB5 statisch vom µController zum FPGA geschaltet ist, arbeitet der Ausgang von IC5, am Anschluss B4 (Pin P0.17 (IOB5 Pin 4), gegen das Ausgangssignal MISO0 des SPI-Schnittstelle (vgl. Anhang G.3). Das SPI-Interface kann technisch nicht einwandfrei funktionieren (vgl. Abb. B.15).

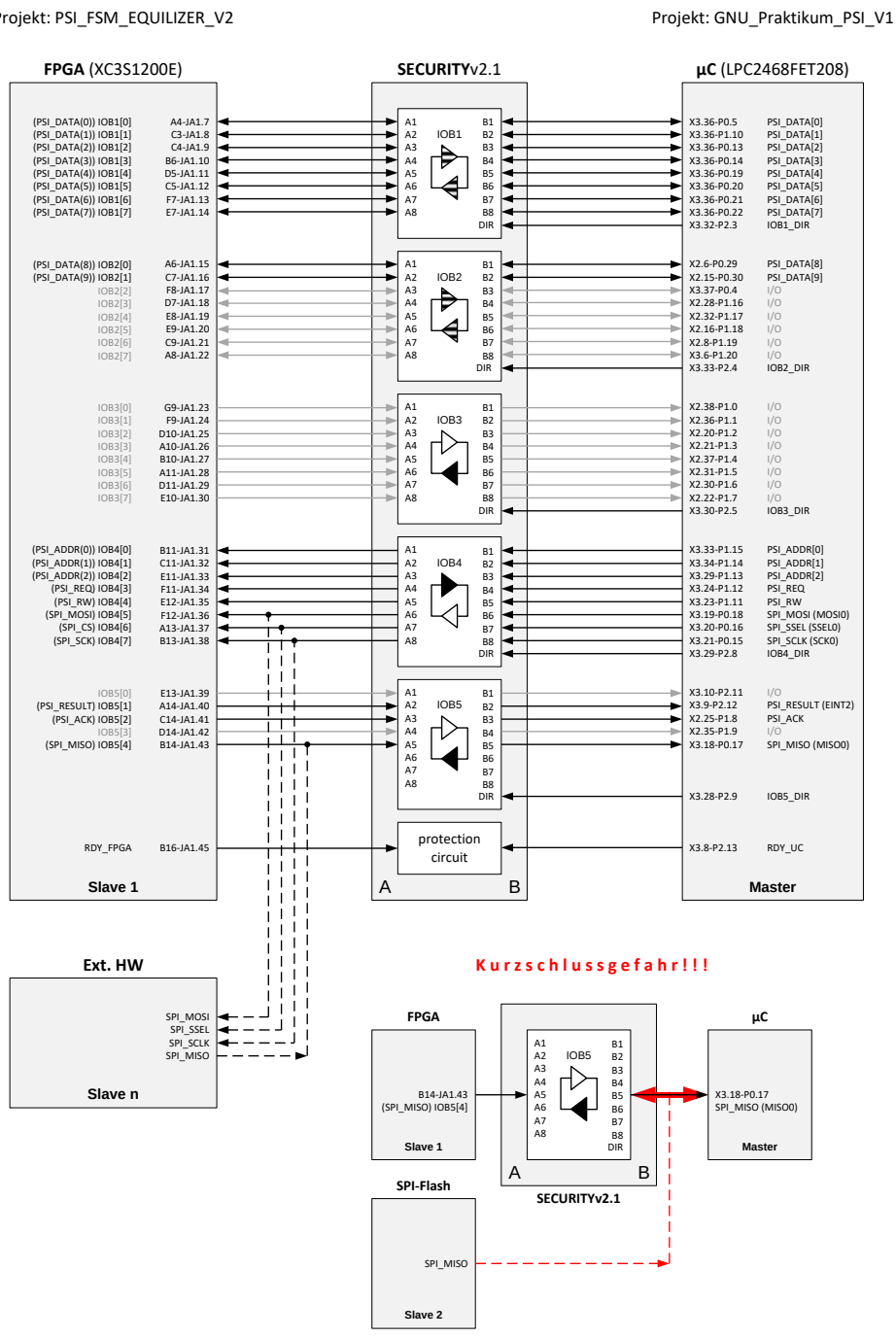


Abb. B.15: SECURITYv2.1-Board SPI-Hardware Kurzschluss

Ein dynamisches Umschalten ist sehr aufwändig realisierbar, da das SPI-Flash in der Hauptschleife angesprochen werden muss und die Kommunikation mit dem FPGA im Interrupt abläuft. Dadurch wird das Zeitverhalten für den Datentransfer beeinträchtigt und die Taktfrequenz der ISR-Ausführung dekrementiert. Somit ist eine korrekte Berechnung des Zeitverhaltens nicht möglich (vgl. Abb. B.16).

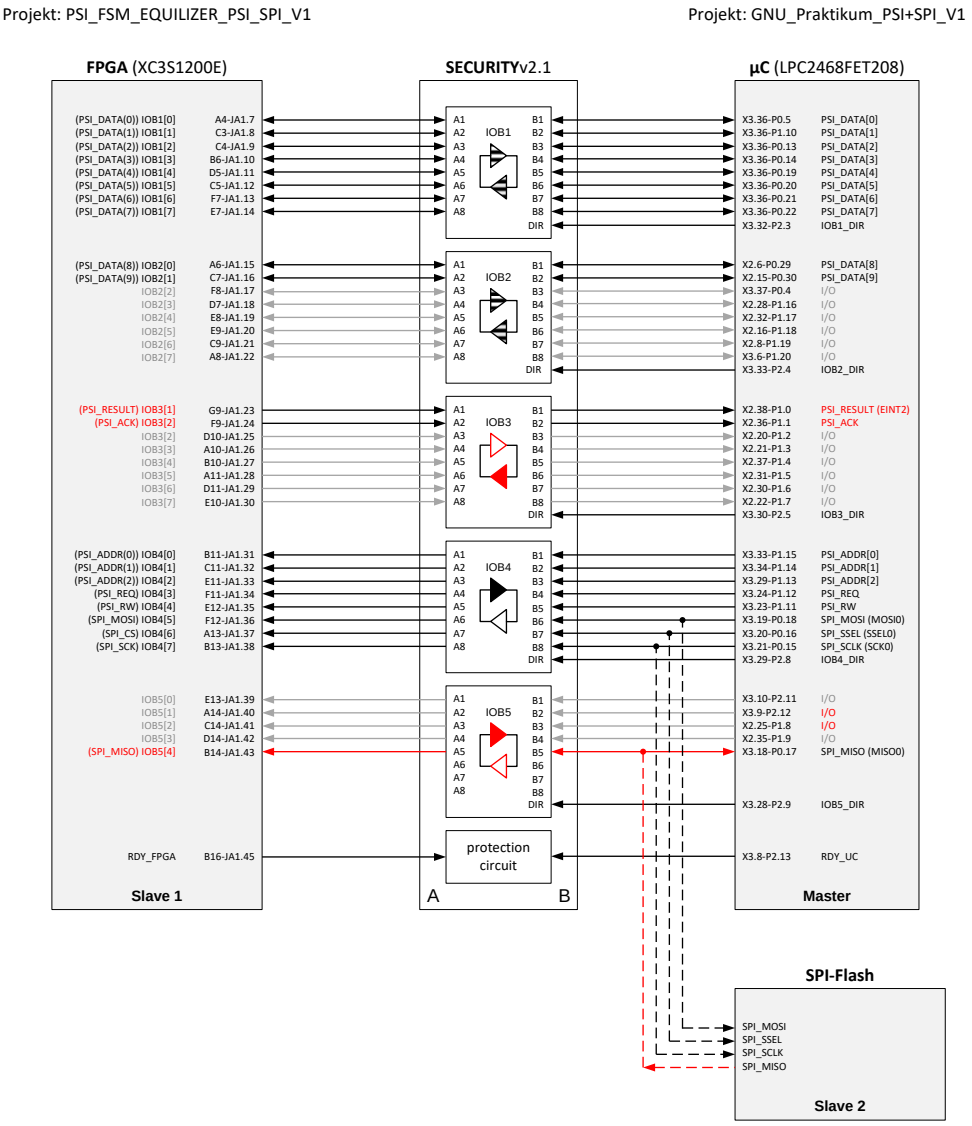


Abb. B.16: SECURITYv2.1-Board SPI-Hardware Lösung des Kommunikationsproblem

C. Funktionsmuster des TI Development-Boards NEXSELv1

C.1 Schaltungskonzept

Die NEXSELv1-Plattform besitzt zwei Basiseinheiten NEXYS2-Board [DIGI] und dem LPC-2468-Stick. Diese sind über die Schutzschaltung des SECURITYv1-Boards (vgl. Anhang B) gekoppelt. Die diskreten Signalkontaktierungen der I/O-Ports des LPC-2468-Sticks mit den IOB-Pins des FPGAs werden über Jumperkabel vorgenommen (vgl. SECURITYv1-Board, Bustreiber-Konfigurationen). Die zusätzlichen Logik-ICs (NAND und Inverter oder andere Komponentenbausteine) sowie der Taster und Schalter des SECURITYv1-Boards, ermöglichen eine Hardware-Beschaltung als User-Interface.

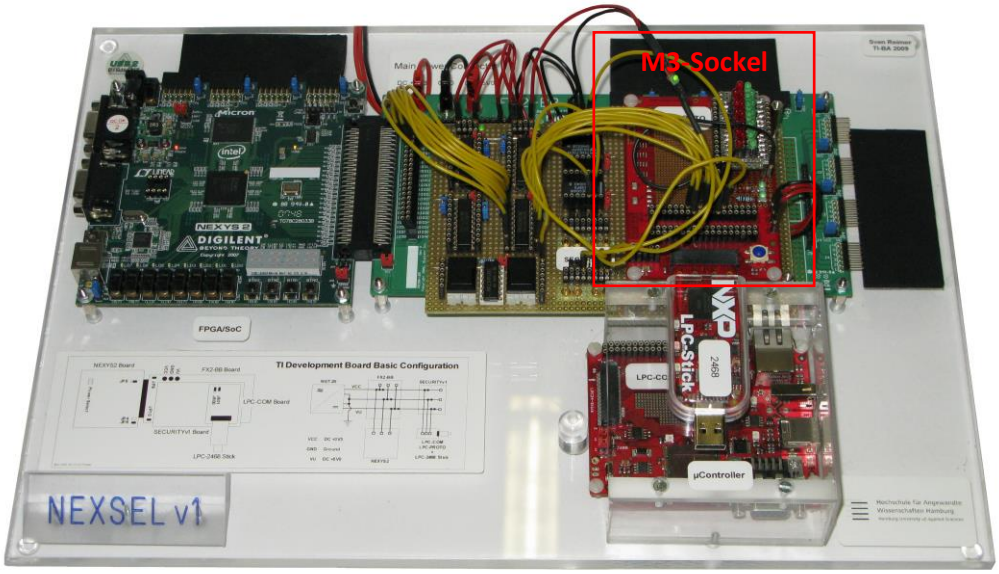


Abb. C.1: NEXSELv1-Plattform

C.2 Modulstecksystem (Peripherie der LPC-Komponenten)

Über den M3-Sockel (vgl. Abb. C.1) der NEXSELv1-Plattform können die externen Peripherien mit dem LPC-2468-Stick verbunden werden. Die Erweiterungsboards LPC-PROTO-Board und LPC-COM-Board [HITEX] sind mit dem LPC-2468-Stick über das PCM80-Stecksystem (vgl. LPC-COM-Board Beschreibung) kontaktiert.

LPC-COM-Board

Das Kommunikations-Erweiterungs-Board mit Erweiterungsstecker (Buchse PCM80, Samtec MEC6-140-02-L-D-RA1) für die LPC Applikation: Development Sticks LPC-2468 (Stecker PCM80), bietet eine Vielzahl externer Hardware-Funktionen.

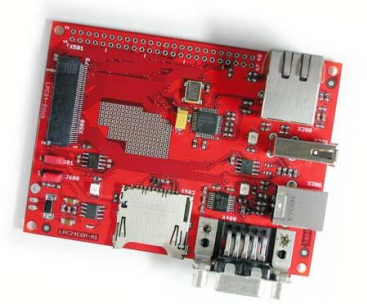


Abb. C.2: LPC-COM-Board

Das LPC-COM Board hat **34 freikonfigurierbare Pins** (vgl. Anhang A) und verfügt über komplexe externen Peripherie-Schnittstellen:

- USB Host und Device
- Ethernet und Phy
- UART
- CAN
- SD-Card Reader

Sowie einen Anschluss für die externe Netzspannungsversorgung (VU 5V0 oder VCC 3V3).

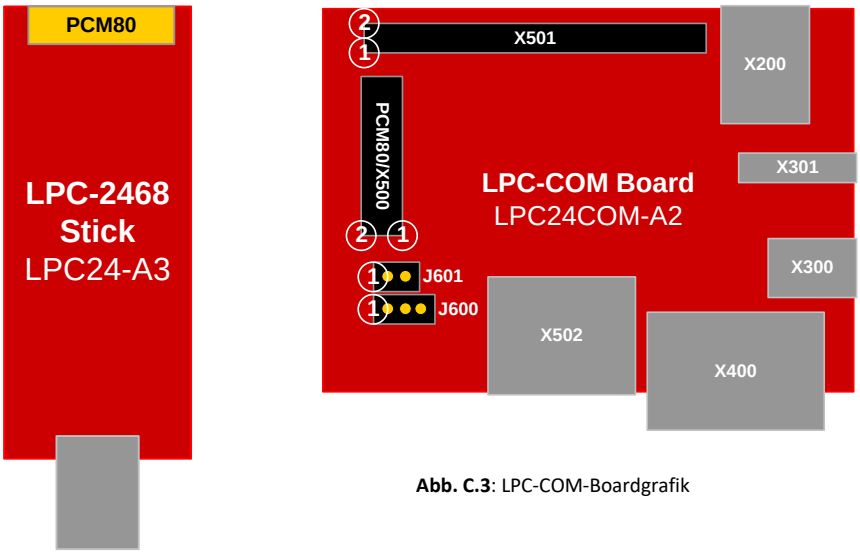


Abb. C.3: LPC-COM-Boardgrafik

- Funktionsblock/Modulanschluss Buchenleiste: [X501] mit 2x VU2 DC +9V0, 2x VU DC +5V0, 2x VCC DC +3V3, 6x GND (nicht als GNDA verwendbar), CAN-BM/P, PWREN_N, RESET_N weitere siehe 6. Pinbelegung
- Externe Peripherie: USB2-A/B [X300 und X301], Ethernet/Phy [X200], UART/CAN [X400] und SD-Card Reader [X502]
- Jumper Blocks [J600 und J601]
- 4x Verschraubungen zur Befestigung auf dem M3-Sockel des SECURITYv1-Boards (vgl. Anhang C.1)

Beim Anschluss der externen Spannungsversorgung an das LPC-COM-Board [HITEX], darf der eingesteckte LPC-2468 Stick nicht über den USB-Port mit dem PC verbunden sein und umgekehrt - Kurzschlussgefahr!

Spannungsversorgung über den USB- Anschluss des LPC-2468 Sticks:

(Jumper Blocks)

Abb. C.4: Jumperblock J600-1

Spannungsversorgung über ein externes Netzversorgungsgerät mit VU via BL: [X501]:

Abb. C.5: Jumperblock J600-1

Verfügbare Pin-Anzahl der Ports und Module des µControllers LPC2468FET208:

- Port P0[31..0]: 14
- Port P1[31..0]: 5
- Port P2[31..0]: 11
- Steuerungsfunktionen: 4

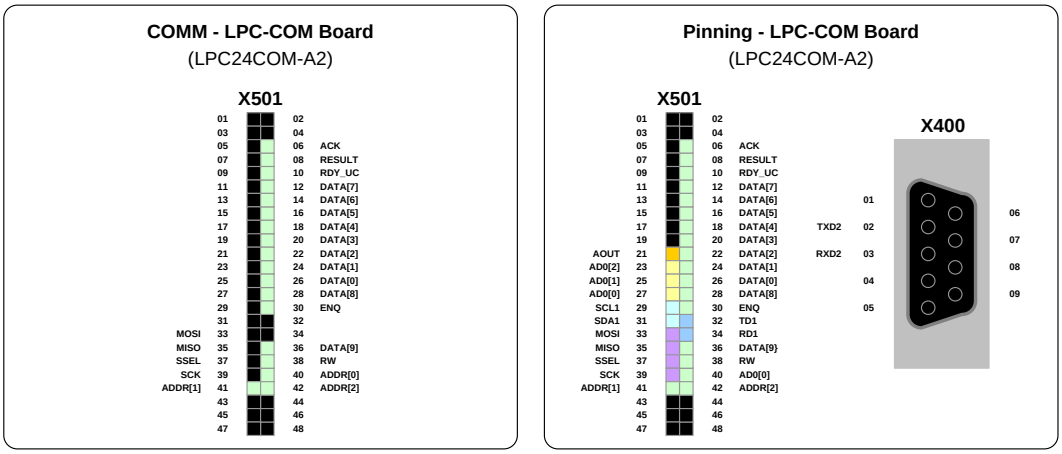


Abb. C.6: LPC-COM-Board Pin-Belegung

Externe Peripherie:

- USB2-A: 1 (abwärtskompatibel)
- USB2-B: 1 (abwärtskompatibel)
- UART/CAN (RS-232): 1
- Ehternet (RJ-45): 1 (10 Mb/s)
- SD-Card Reader: 1 (9 MB/s, Typ: I/II)

Die aktivierbaren Hardware-Module - über die C-Funktion: PINSELx auf die verfügbaren Pins s. o. - und weitere Optionen können der Tabellenkalkulation (vgl. Anhang A.6) entnommen werden:

LPC-PROTO-Board

Prototypen-Erweiterungs-Board mit Erweiterungsstecker (Buchse PCM80, Samtec MEC6-140-02-L-D-RA1) und low-active Reset-Taster (Signal: RESET_N) für die LPC- Development-Applikation LPC-2468-Stick mit Steckerleiste PCM80.

Abb. C.7: LPC-PROTO-Board

Das LPC-PROTO-Board bietet gegenüber dem LPC-COM-Board die Nutzung aller verfügbaren Module und Peripheriefunktionen der Ports P0, P1, und P2 des µControllers LPC2468FET208, die über den Schnittstellenstecker PCM80 diskret (vgl. Anhang A.7) verbunden sind.

	LPC-COM Board	**LPC-PROTO Board**
Pin gesamt Anzahl:	34 (1:1.95)	66

Bestückung der Lochraster: 2,54 mm, 1,27mm und SMD-Pads (SOT-223, TSSOP-20):

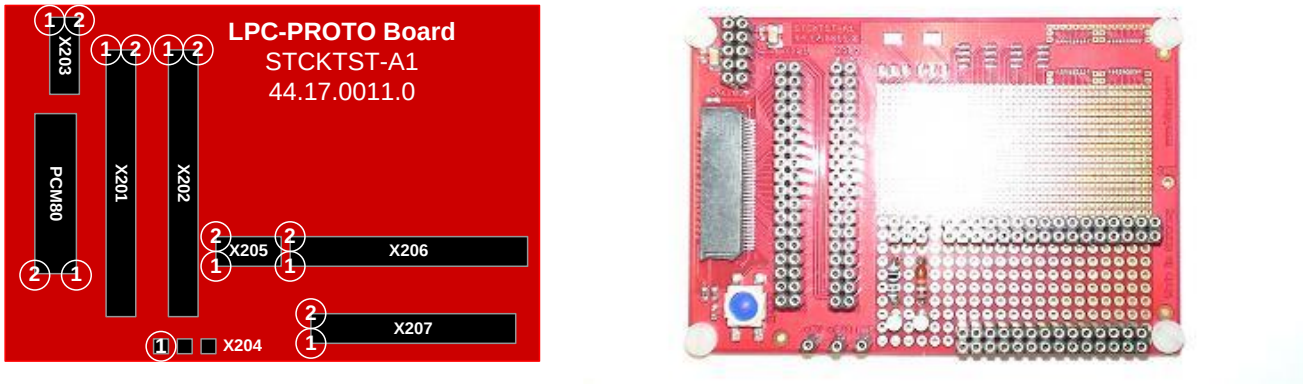


Abb. C.8: LPC-PROTO-Boardgrafik

- Stromversorung Pins [X204] mit 1x VU DC +5V0, 1x VCC DC +3V3 und 1x GND (nicht als GNDA verwendbar)
- Funktionsblock/Spannungsverteiler Buchsenleiste [X203] mit 1x VU, 2x VCC, 4x GND, 2x GNDA (nicht als GND verwendbar) und Signal PWREN_N
- Modulanschlüsse BL: [X201] und [X202] mit siehe 7. Pinbelegungsübersicht
- 4x Verschraubungen zur Befestigung auf dem M3-Sockel des SECURITYV1 Boards

Die Evaluierung der LPC-PROTO-Platine ergab folgende notwendigen Erweiterungen:

- Spannungsversorgung/-verteilerblock mit 2x VU, 2x VCC und 2x GND BL: [X205]
- Signal LEDs für VCC und VU (grün)
- Buchsenleisten 2x 8-pin und 2x 7-pin für die Aufnahme von Pmods oder zur Verwendung als Verteilerblock 2x 15-pin BL: [X206 und X207]

Beim Anschluss der externen Spannungsversorgung an das LPC-PROTO-Board, darf der eingesteckte LPC-2468-Stick nicht über den USB-Port mit dem PC verbunden sein und umgekehrt - Kurzschlussgefahr!

Buchsenleisten Typ: RM2,54

Nutzbare Pin Anzahl der ARM7™ TDMI Module und Ports:

- Port P0[31..0]: 24
- Port P1[31..0]: 27
- Port P2[31..0]: 11
- Steuerungsfunktionen: 4

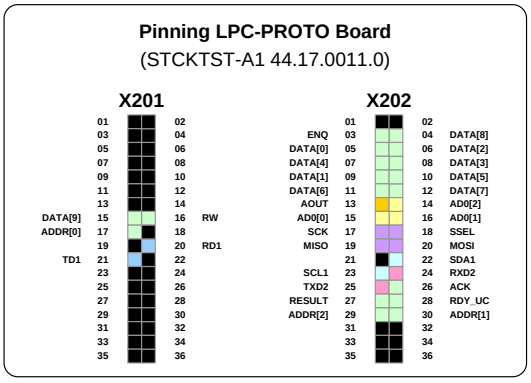


Abb. C.9: LPC-PROTO-Pin-Belegung

Die aktivierbaren Hardware-Module - über die Funktion: PINSELx auf die verfügbaren Pins s. o. - und weitere Optionen können der Tabellenkalkulation (vgl. Anhang A) entnommen werden:

C.3 Grundkonfiguration

Alle Einstellungen (vgl. Abb. C.1) des Funktionsmusters NEXSELv1 sind im Netzspannungsfreiem Zustand vorzunehmen. Die Versorgung des Gesamtsystems erfolgt über drei 2mm Buchsen (GND, VCC 3V3 und VU 5V0) des FX2-BB-Boards [DIGI]. Wenn die Development-Boards einzeln mit Netzspannung versorgt werden, sind die Jumper JP9, JP10, J600 und J601 (Trennung der Versorgungsspannung) abzuziehen.

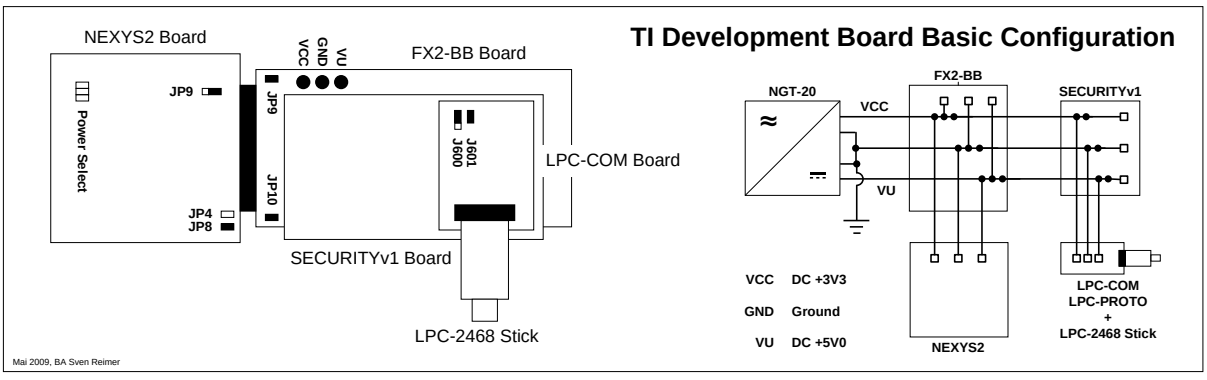


Abb. C.10: NEXSELv1 Grundkonfiguration

D. Konstruktionspläne

D.1 SECURITYv1-Board-Platine

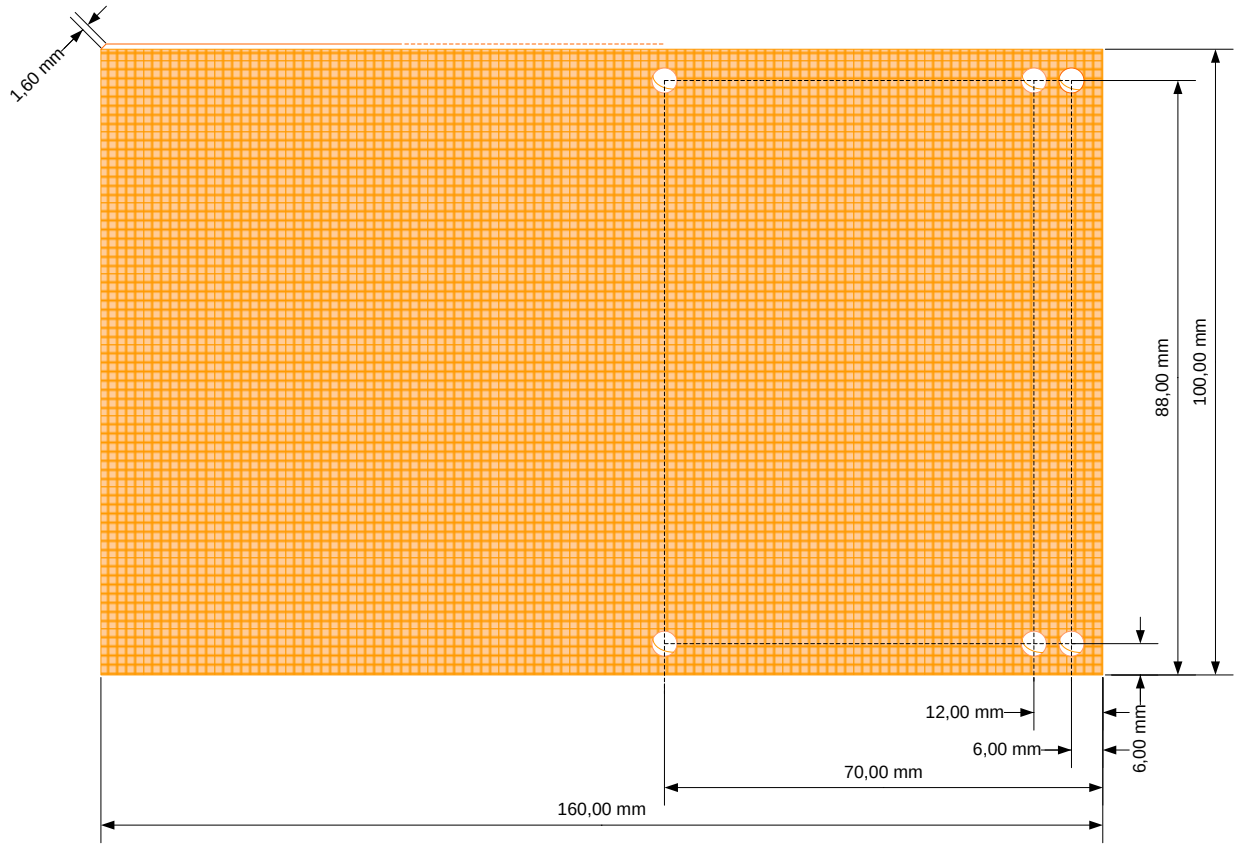


Abb. D.1: SECURITYv1-Board Platinenabmessungen

E. C-Funktionen für den Datentransfer

Alle nachfolgenden aufgeführten Konstanten (#defines) und Variablen sind in der Header-Datei config.h (s. CD-ROM) implementiert.

E.1 Steuerung des SECURITYv2.1-Boards (securegpio.c/.h)

Tab. E.1: C-Bibliothek securegpio.h/.c

Fkt.	Routine	Beschreibung	Parameter							Rückgabewert			
			Pos.	Typ	Größe	Vorz.	Konstanten Bezeichung	Wert	Funktion	Typ	Größe	Vorz.	Bemerkung
Init	**initIOB**	Initialisiert die Grundkonfiguration der IOBs und GPIO-Pins	1	BOOL	8bit	unsigned	INIT_ALL_IOBS INIT_USER_IOBS	0x01 0x00	Grunkonfig. Nur alle IOBs auf Eingang				
	Beispiel:	initIOB(INIT_ALL_IOBS) - IOB1..5 sind auf Eingang geschaltet und die Pull-Down-Resistors sind für alle Pins aktiviert.											
Set	**setIOBDir**	Schaltet die Transferrichtung der Bustreiber	1	BYTE	8bit	unsigned	IOB1, IOB2..IOB5	0x01..0x05	Bustreiber				
			2	BOOL	8bit	unsigned	IOB_DIR_A_2_B IOB_DIR_B_2_A	0x01 0x00	µC empfängt µC sendet				
	Beispiel:	setIOBDir(IOB1, IOB_DIR_A_2_B) - Die Transferrichtung, der acht Datenbits des IOB1, ist vom FPGA zum µC geschaltet											
	setIOBPin	Setzt den Pegel des Datenbits vom gewählten IOB	1	BYTE	8bit	unsigned	IOB1, IOB2..IOB5	0x01..0x05	Bustreiber				
			2	BYTE	8bit	unsigned	PIN0, PIN1..PIN7	0x00..0x07	Datenbit				
			3	BOOL	8bit	unsigned	TRUE FALSE	0x01 0x00	Wert				
	Beispiel:	setIOBPin(IOB1, PIN5, TRUE) - Setzt IOB1(5) = P0.20 auf High											
	setIOBPort	Setzt die 8 Pegel der Datenbits des gewählten IOBs	1	BYTE	8bit	unsigned	IOB1, IOB2..IOB5	0x01..0x05	Bustreiber				
			2	BYTE	8bit	unsigned							
	Beispiel:	setIOBPort(IOB3, 0x06) - Schaltet die Signalpegel IOB3(2 , 1) auf High und die restlichen 6 Bits auf Low											

Set	**setIOBPortDir**	Schaltet die Transfer-richtung des Daten-bytes des gewählten IOBs	1	BYTE	8bit	unsigned	IOB1, IOB2..IOB5	0x01..0x05	Bustreiber				
			2	BYTE	8bit	unsigned	GPIO_PORT_DIR_IN GPIO_PORT_DIR_OUT	0x00 0xFF	Alle als Input Alle als Output				
	Beispiel:	setIOBPortDir(IOB2, GPIO_PORT_DIR_IN) - Setzt das Datenbyte auf Empfangsmodus (A -> B)											
	setSystemReady	Setzt das Ready-Signal des µC für das SECURITYv2.1-Board	1	BOOL	8bit	unsigned	TRUE FALSE	0x01 0x00	System ready 3-State-Modus				
	Beispiel:	setSystemReady(TRUE) - Setzt den Pegel des RDY_UC-Signals (P2.13) auf High und aktiviert den Status: System bereit											
Get	**getIOBDir**	Gibt die aktuelle Transferrichtung des IOB zurück	1	BYTE	8bit	unsigned	IOB1, IOB2..IOB5	0x01..0x05	Bustreiber	BYTE	8bit	unsigned	0 = B -> A 1 = A -> B
	Beispiel:	getIOBDir(IOB1) - Liefert die Datentransferrichtung des IOB1											
	getIOBPin	Liefert den Signalpe-gel des Datenbits	1	BYTE	8bit	unsigned	IOB1, IOB2..IOB5	0x01..0x05	Bustreiber	BYTE	8bit	unsigned	Low High
			2	BYTE	8bit	unsigned	PIN0, PIN1..PIN7	0x00..0x07	Datenbit				
	Beispiel:	getIOBPin(IOB2, PIN3) - Gibt den Pegel von IOB2(3) = P1.16 zurück											
	getIOBPort	Gibt die aktuellen acht Signalpegel des gewählten IOBs zurück	1	BYTE	8bit	unsigned	IOB1, IOB2..IOB5	0x01..0x05	Bustreiber	BYTE	8bit	unsigned	8x Pegel
	Beispiel:	getIOBPort(IOB5) - Liefert die Signalpegel der 5 Datenbits IOB5(04..00) in einem Byte zurück											
	getIOBPortDir	Gibt die acht unab-hängigen Datentrans-fer- richtungen zurück	1	BYTE	8bit	unsigned	IOB1, IOB2..IOB5	0x01..0x05	Bustreiber	BYTE	8bit	unsigned	8x Dir.
	Beispiel:	getIOBPortDir(IOB2) - Gibt die Transferrichtungen der 8 Datenbits IOB2(07..00) zurück (0 = B -> A und 1 = A -> B)											
	getSystemReady	Gibt den Status des µC zurück								BYTE	8bit	unsigned	Systemstatus
	Beispiel:	getSystemReady() - Liefert den aktuellen Systemstatus RDY_UC = P2.13 des µC (0 = 3-State-On oder 1 = System ready)											

E.2 Software-Modul PSI (securepsi.c/.h)

Steuerung des Kommunikationstransfers und der Daten über das Parallele Software Interface.

Tab. E.2: C-Bibliothek securepsi.h/.c

Fkt.	Routine	Beschreibung	Parameter							Rückgabewert			
			Pos.	Typ	Größe	Vorz.	Konstanten Bezeichung	Wert	Funktion	Typ	Größe	Vorz.	Bemerkung
Init	**initPSI**	Initialisierung des PSI-Betriebsmodus	1	BOOL	8bit	unsig.	PSI_IRQ_MODE_ON PSI_IRQ_MODE_OFF	0x01 0x00	IRQ Polling				
	Beispiel:	initPSI(PSI_IRQ_MODE_ON) - Aktiviert den IRQ-Modus und konfiguriert die Hardware-Funktionen.											
	initPSIEint2	Initialisierung des Edge-Sensitiven ext. Interrupts EINT2											
	Beispiel:	initPSIEint2() - Aktivierung der Hardware-Peripherie EINT2 (P2.12)											
	initPSITimer0	Initialsierung des HW-Timers0 in Mikrose-kunden	1	WORD	16bit	unsig.							
	Beispiel:	initPSITimer0(23) - Periodischer Timer mit einer Frequenz von ~44 KHz											
Set	**setPSIAddress**	Setzt die 3bit Kompo-nenten- adresse des FPGAs	1	BYTE	8bit	unsig.							
	Beispiel:	setPSIAdress(0x02) - Setzt die Komponentenadresse auf 2 (P1.15..13)											
	setPSIData	Setzt das 10bit Datum	1	WORD	16bit	unsig.							
	Beispiel:	setPSIData(0x00F0) - Setzt die Datenbits auf 240d (P0.(30 , 29 , 22..19 , 14 , 13 , 5) und P1.10)											
	setPSIRequest	Setzt die 1.+3. Stufe des 4-PHs	1	BOOL	8bit	unsig.	FALSE TRUE	0x00 0x01	Low High				
	Beispiel:	setPSIRequest(TRUE) - Den Signalpegel des PSI_REQ-Signals = P1.12 auf High setzen											

Set	**setPSIReadWrite**	Setzt die Daten- trans-ferrichtung	1	BOOL	8bit	unsig.	FALSE TRUE	0x00 0x01	B -> A A -> B				
	Beispiel:	setPSIReadWrite(FALSE) - Konfiguriert die Hardware und setzt den PSI_RW-Signalpegel = P1.11 (B -> A)											
Get	**getPSIAck**	Liefert den PSI_ACK-Signalpegel								BOOL	8bit	unsig.	FALSE TRUE
	Beispiel:	getPSIAck() - Gibt den aktuellen Pegel des PSI_ACK-Signals = P1.8 ,der 2.+4. Stufe des 4-PHs, zurück											
	getPSIAddress	Liefert die 3bit Adres-se (PSI_ADDR)								BYTE	8bit	unsig.	K.-Adresse
	Beispiel:	getPSIAddress() - Gibt die aktuelle Adresse der FPGA-Komponente zurück (P1.15..13)											
	getPSIData	Liefert das Datum PSI_DATA								WORD	16bit	unsig.	Datum
	Beispiel:	getPSIData() - Gibt das 10bit bidirektionale Datum zurück (P0.(30 , 29 , 22..19 , 14 , 13 , 5) und P1.10)											
	getPSIResult	Liefert den Result-Signalpegel PSI_RESULT								BOOL	8bit	unsig.	FALSE TRUE
	Beispiel:	getPSIResult() - Gibt den aktuellen Wert des PSI_RESULT-Signals = P2.12 zurück											
	getPSIRequest	Liefert den Pegel des PSI_REQ-Signals								BOOL	8bit	unsig.	FALSE TRUE
	Beispiel:	getPSIRequest() - Gibt den aktuellen Wert des PSI_REQ-Signals = P1.12 zurück											
	getPSIReadWrite	Liefert den Pegel des PSI_RW-Signals								BOOL	8bit	unsig.	FALSE TRUE
	Beispiel:	getPSIReadWrite() - Gibt die aktuelle Datentransferrichtung PSI_RW = P1.11 zurück											
R/W	**readPSIData**	Automatisches einle-sen des 10bit Datums	1	BYTE	8bit	unsig.			Adresse	WORD	16bit	unsig.	10bit Datum
	Beispiel:	readPSIData(0x02) Liest über das 4-P-H das Datum (P0.(30 , 29 , 22..19 , 14 , 13 , 5) und P1.10) der 2 FPGA-Komponente ein											
	writePSIData	Automatisches senden des 10bit Datums	1 2	BYTE WORD	8bit 16bit	unsig. unsig.			Adresse Datum				
	Beispiel:	writePSIData(0x04, 0x000F) - Sendet das PSI_DATA = 15d, an die Komponentenadresse 4											

ISR	**isr_PSITimer0** **isr_PSIEint2**	ISR für den periodischen Timer ISR für den EINT2												
Timer	**start_PSITimer0**	Startet den Hardware-Timer												
	stopp_PSITimer0	Stoppt den Hardware-Timer												
PT	**isr_PSITimer0 __ISR__**	Prototyp												
	isr_PSIEint2 __ISR__	Prototyp												

E.3 GPIO-Funktionen (gpio.c/.h)

Konfiguration und Steuerung der FastGPIO-/GPIO-Funktionen des ARM7^{TM} TDMI.

Tab. E.3: C-Bibliothek gpio.h/.c

Fkt.	Routine	Beschreibung	Parameter							Rückgabewert			
			Pos.	Typ	Größe	Vorz.	Konstanten			Typ	Größe	Vorz.	Bemerkung
							Bezeichung	Wert	Funktion				
Init	**initGPIOPortSpeed**	Aktivierung des Fast- oder GPIO-Modus für P0..1(31..00)	1	DWORD	32bit	unsigned	GPIO_SPEED_SLOW	0x11111110	GPIO-Modus				
							GPIO_SPEED_FAST	0x00000001	FGPIO-Modus				
	Beispiel:	initGPIOPortSpeed(GPIO_SPEED_FAST) - Schaltet den FastGPIO-Modus für die Ports P0 und P1 ein											
Set	**setGPIOPin**	Setzt den Pin des gewählten Ports, auf den vorgegebenen Ausgangspegel	1	BYTE	8bit	unsigned	PORT0, PORT1..PORT4	0x00..0x04	Portnummer				
			2	BYTE	8bit	unsigned	PIN0, PIN1..PIN31	0x00..0x1F	Pinnummer				
			3	BOOL	8bit	unsigned	TRUE	0x01	Pinwert = '1'				
							FALSE	0x00	Pinwert = '0'				
	Beispiel:	setGPIOPin(PORT1, PIN11, TRUE) - Setzt P2.11 auf High											
	setGPIOPort	Setzt die Pinwerte (31..00) des gewählten Ports	1	BYTE	8bit	unsigned	PORT0, PORT1..PORT4	0x00..0x04	Portnummer				
			2	DWORD	32bit	unsigned							
	Beispiel:	setGPIOPort(PORT0, 0x000000FF) - Setzt die Pegel P0.7..0 auf High und P0(31..08) auf Low											
	setGPIOPinDir	Setzt die Datenflussrichtung des Pins vom gewählten Port	1	BYTE	8bit	unsigned	PORT0, PORT1..PORT4	0x00..0x04	Portnummer				
			2	BYTE	8bit	unsigned	PIN0, PIN1..PIN31	0x00..0x1F	Pinnummer				
			3	DWORD	32bit	unsigned	GPIO_PIN_DIR_IN	0x00	Input				
							GPIO_PIN_DIR_OUT	0x01	Output				
	Beispiel:	setGPIOPinDir(PORT2, PIN3, GPIO_PIN_DIR_IN) - Schaltet P2.3 auf Eingang (lesend)											
	setGPIOPortDir	Schaltet die Pins 31..00 unabhängig, des gewählten Ports, auf Ein- oder Ausgang	1	BYTE	8bit	unsigned	PORT0, PORT1..PORT4	0x00..0x04	Portnummer				
			2	DWORD	32bit	unsigned	GPIO_PORT_DIR_IN	0x00000000	Alle als Input				
							GPIO_PORT_DIR_OUT	0xFFFFFFFF	Alle als Output				
	Beispiel:	setGPIOPortDir(PORT0, 0x00000003) - Setzt P0.1..0 auf Ausgang (schreibend) und die restlichen auf Eingang											

Get	**getGPIOPin**	Liefert den Signalpegel des Pins vom gewählten Port	1	BYTE	8bit	unsigned	PORT0, PORT1..PORT4	0x00..0x04	Portnummer	DWORD	32bit	unsig.	Pinpegel
			2	BYTE	8bit	unsigned	PIN0, PIN1..PIN31	0x00..0x1F	Pinnummer				
	Beispiel:	getGPIOPin(PORT1, PIN3) - Liefert den Pegel von P1.3 zurück											
	getGPIOPort	Liefert alle 32 Signalpegel des gewählten Ports	1	BYTE	8bit	unsigned	PORT0, PORT1..PORT4	0x00..0x04	Portnummer	DWORD	32bit	unsig.	Portpegel
	Beispiel:	getGPIOPort(PORT2) - Gibt die Signalpegel P2.31..0 zurück											
	getGPIOPinDir	Gibt die Transferrichtung des Pins vom gewähltem Port zurück	1	BYTE	8bit	unsigned	PORT0, PORT1..PORT4	0x00.0x04	Portnummer	DWORD	32bit	unsig.	0 = Input
			2	BYTE	8bit	unsigned	PIN0, PIN1..PIN31	0x00..0x1F	Pinnummer				1 = Output
	Beispiel:	getGPIOPinDir(PORT1, PIN2) - Liefert die Datentransferrichtung von P1.2											
	getGPIOPortDir	Gibt alle 32 Transferrichtungen des gewählten Ports zurück	1	BYTE	8bit	unsigned	PORT0, PORT1..PORT4	0x00..0x04	Portnummer	DWORD	32bit	unsig.	0 = Input 1 = Output
	Beispiel:	getGPIOPortDir(PORT0) - Liefert die Transferrichtungen von P0.31..0											

F. Grundkonfigurationen

F.1 LPC-2468-Stick (ARM7™ TDMI)

Alle möglichen Grundeinstellungen, wie z.B. Taktfrequenz [5] und GPIO-Modus oder Andere, werden mit dem StartEasy-Tool [HITEX] vorgenommen. Die Software erstellt nach den vorgenommenen Einstellungen das Projekt, das die wichtigsten Systemdateien enthält, wie z.B. für den Bootvorgang der Startupcode.

Tab. F.1: LPC-2468-Stick Grundkonfiguration

Funktion	Wert/Einstellung
PLL state	Enabled
CLK source	4MHz (Systemoszillator)
Multiplier (M)	72
Divider (N)	2
CPU CLK divider	4
USB CLK divider	6
MAM module	Parity enabled
GPIO-/FastGPIO modus (P0 und P1)	FastGPIO

Eine andere Möglichkeit ist das manuelle abändern der Konfigurationsdateien im Editor. Beim Erstellen des Codes sollte darauf geachtet werden, dass keine Dateien durch das Tool ungewollt überschrieben werden. Vorgenommene Einstellungen:

```
Datei: startup.s

Makro: Start_init_s: (Micro controller specific code)

GPIO-Output: RegWrite SCS, 0x21 (für FastGPIO oder 0x20 für GPIO)
(SCS = System Control and Status)
Frequenzfaktoren (Mul/Div): setup_PLL, 72, 2
Taktfrequenz (Mux): RegWrite CCLKCFG, 3 (für 72MHz oder 5 für 48MHz)
```

Die konfigurierten Einstellungen ermöglichen den Betrieb des µControllers mit der Taktfrequenzen 72MHz (SYSCLK) und 48MHz für die USB-Schnittstelle. Für die Berechnungen der Einstellungen von Divider und Multiplier, kann der PLL Konfiguration verwendet werden (../ Erprobungsplattform / Mikrocontroller / LPC24xx_PLL-Konfiguration.xls).

Bei Rücksetzung des ARM7™ TDMI der folgenden Peripheriegeräte sind standardmäßig deaktiviert (Stromsparmodus):

- Analog zu Digital Konverter
- beide CAN-Controller
- Hardware-Timer 2 und 3
- UART 2 und 3
- I²S-Schnittstelle
- SD-Karte (LPC-COM-Board)
- General Purpose DMA
- Ethernet-MAC (LPC-COM-Board)
- USB Controller (LPC-COM-Board)

F.2 NEXYS2-Board (Spartan3E™)

Alle Konfigurationen der IOBs und externen Peripherie werden über das standardisierte UCF-Template voreingestellt. Dieses beinhaltet sämtliche verfügbaren Pins und entsprechende Informationen über die Kontaktierungen oder angeschlossen Hardware-Komponenten der NEXSELv2-Plattform.

```
..
#-- Ext. Takt/Oszillator Socket
#-------------------------------------------------------------------------------------------------
#NET "SOCKET"    LOC = "U9"; # Bank = 2, Pin name = IO_L13P_2/D4/GCLK14, Type = DUAL/GCLK


#-- Steuerleitung/Control Signal (SECURITYv2-Board)
#-------------------------------------------------------------------------------------------------
NET "RDY_FPGA"   LOC = "B16"; # Bank = 0, Pin name = IO_L01P_0, Type = I/O, J1A-45


#-- 37x I/O-Pins (FX2-100 Connector)
#-------------------------------------------------------------------------------------------------
#- I/O-Block: 1 (SECURITYv2-Board 3-State-Block: 1)
NET "IOB1<0>"    LOC = "A4"; # Bank = 0, Pin name = IO_L24P_0, Type = I/O, J1A-07, uC: X3(P0.5)
NET "IOB1<1>"    LOC = "C3"; # Bank = 0, Pin name = IO_L25P_0, Type = I/O, J1A-08, uC: X2(P1.10)
NET "IOB1<2>"    LOC = "C4"; # Bank = 0, Pin name = IO, Type = I/O, J1A-09, uC: X2(P0.13)
NET "IOB1<3>"    LOC = "B6"; # Bank = 0, Pin name = IO_L20P_0, Type = I/O, J1A-10, uC: X2(P0.14)
NET "IOB1<4>"    LOC = "D5"; # Bank = 0, Pin name = IO_L23N_0/VREF_0, Type = VREF, J1A-11, uC: X3(P0.19)
NET "IOB1<5>"    LOC = "C5"; # Bank = 0, Pin name = IO_L23P_0, Type = I/O, J1A-12, uC: X3(P0.20)
NET "IOB1<6>"    LOC = "F7"; # Bank = 0, Pin name = IO_L19P_0, Type = I/O, J1A-13, uC: X2(P0.21)
NET "IOB1<7>"    LOC = "E7"; # Bank = 0, Pin name = IO_L19N_0/VREF_0, Type = VREF, J1A-14, uC: X2(P0.22)
..
```

G. Hardware-Module des ARM7™ TDMI

G.1 Power-Einstellungen

Mit dem Stromverbrauch des µControllers besteht ein direkter Zusammenhang mit der Anzahl aktiver Peripherien und der Taktfrequenz. Der LPC-2468-Stick [HITEX] ist keine Ausnahme. Die Einfachheit und niedrige Gate-Zählung der ARM7™ TDMI tragen zu seinem niedrigen Stromverbrauch bei. Dieser hat vier verschiedene Power-Down-Modi, mit zwei separaten Steuerleitungen zur Steuerung der Netzspannung. Neben dem Power-Control-Modi kann die SYSCLK zu jeder Peripherie gestoppt werden .Dies kann für ein dynamisches Power-Management genutzt werden.

Power-Domains

Innerhalb der µController-Struktur [5] gibt es zwei getrennte Domains. Eine besteht aus der Echtzeituhr und 2k low-power SRAM als Batterie-RAM. Die zweite Power-Domain besteht aus der ARM7™-CPU und der restlichen Peripherie. Diese separate Spannungsversorgungs-Domänen erlauben den Großteil der Hardware abzuschalten, während kritischen System-Variablen dem Batterie-RAM beibehalten werden.

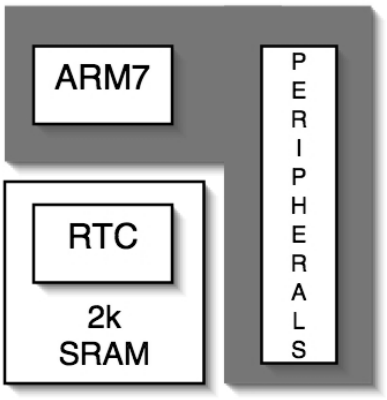


Abb. G.1: ARM7 TDMI Power-Domains

Globale Power-Modi

Die vier globalen Power-Modi werden durch das PCON-Register gesteuert. Die PM0-, PM1- und PM2-Bits schalten nachfolgende Modi:

- Idle
- Sleep
- Power-Down
- Deep-Power
- Down-Mode.

Abb. G.2: PCON-Register [HITEX, 5]

Power-Down-Modus hält den Prozessor- und die Periphere-CLK an. Welche externe Interrupts verwenden, um den Prozessor neu zu starten oder die Peripheriegeräte.

Wenn der **Idle-Modus** eingestellt ist, werden die drei Power-Control-Bits auf 001 gesetzt. Im **Standby-Modus** wird die SYSCLK gestoppt und die Peripherie ist weiterhin aktiv. Ein Reset oder ein Interrupt von einer Peripherie bewirkt, dass der CPU-Takt wieder aktiv ist und die sequentielle Befehlsabarbeitung weiterläuft.

Wird der **Sleep-Modus** aktiviert, werden die drei Power-Control-Bits auf 101 gesetzt. Im Sleep-Modus sind alle Taktungen an die CPU und Peripheriegeräte gestoppt, außer der Echtzeituhr. Die externen Oszillatoren sind dann ausgeschaltet und der PLL wird angehalten [5]. Doch damit eine beschleunigte Takt-Wiederaufnahme erfolgt, wird in den Standby-Modus geschaltet, während der On-Chip-RC-Oszillator noch aktiv ist. Die Inhalte des SRAMs und der Register bleiben erhalten

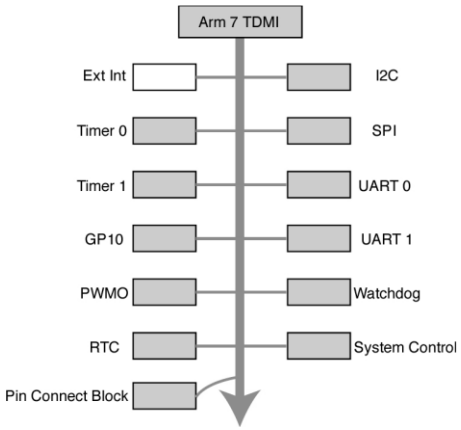


Abb. G.3: PCON-Register Sleep-Modus

Wenn der **Power-Down-Modus** eingestellt ist, werden die Power-Control-Bits auf 010 gesetzt. Dieser Modus arbeite wie der Sleep-Modus, außer dass die FLASH-Speicher aktiv bleiben. Wenn ein Neustart des µControllers erfolgt, beträgt die Startzeit zusätzlich 100µsec, bevor auf den FLASH-Speicher zugegriffen werden kann.

Ist der **Deep-Power-Down-Modus** aktiviert, werden die Power-Control-Bits auf 110 gesetzt. Bei diesem Modus ist keine Netzspannungsversorgung vorhanden, diesbezüglich können die Inhalte des SRAMs und der CPU-Register verloren gehen. Wenn die Stromversorgung des VBAT geschaltet ist, taktet die RTC weiter und der Inhalt der RAM-Batterie bleibt erhalten. Die CPU kann durch einen Reset oder einen RTC-Alarm (Interrupt) in den Idle-Modus schalten.

Das **Peripheral-Power-Control-Register** (PCONP) enthält ein CLK-Bit für jede Benutzer-Peripherie. Wenn das Gating-Bit auf 1 gesetzt ist, wird die Hardware-Clock aktiviert. Über dieses Register können die Peripherie-Module aktiviert oder deaktiviert werden. Das Peripheral-Power-Control-Register kann auch verwendet werden, um dynamisch zur Laufzeit Module ein- oder abzuschalten.

G.2 ADC

Der A/D-Wandler [5] des µControllers ist ein 10-Bit-sukzessiv Approximation-Wandler mit einer Berechnungszeit von 2,44µs pro Data. Dieser arbeitet mit 6 oder 8 Multiplexer-Eingängen je nach Konfiguration. Die Programmierschnittstelle für die A/D-Wandler ist in Abb. G.4 dargestellt.

Über das Steuerregister ADCR wird die Konfiguration des Konverters vorgenommen Umwandlung aktiviert. Zuerst müssen für die Konvertierung durch den ADC die Peripherie CLK voreingestellt werden. Die PCLK muss mit dem Divider-Faktor auf 4,5MHz konfiguriert werden. Dieses ist die maximale Taktung des A/D-Wandlers.

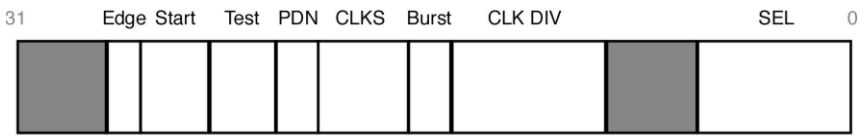


Abb. G.4: ADC-Steuerregister

PCLK geteilt durch den Wert von CLKdiv und dekrementiert um 1, ergibt die Gleichung zur Berechnung des A/D-Wandler-Takts Adclk.

```
CLKdiv = (PCLK / Adclk) - 1;
```

Der A/D-Wandler kann Messungen über die externen Pins des LPC-2468-Sticks, wenn diese als GPIO konfiguriert sind, durchführen. Mithilfe des PINSEL-Registers werden diese in den ADC-Modus geschaltet. Die Genauigkeit der Umwandlung wird über das CLKS-Register vordefiniert. Der ADC verfügt über eine maximale Auflösung von 10Bit. Dieser kann so programmiert werden, dass jede Konvertierung bis minimal 3 Bits berechnet wird. Die Umwandlungsauflösung ist gleich der Anzahl der Taktzyklen pro Konvertierung inkrementiert um 1. Daher benötigt der Wandler für ein 10Bit-Ergebnis

11 ADCLK-Zyklen und vier für ein 3-Bit-Ergebnis, sobald dieser konfiguriert ist. Der ADC verfügt über zwei Umwandlung-Modi, Hardware und Software. Im Hardware-Modus kann die Anzahl von Kanälen, die der A/D-Wandler durchläuft, konfiguriert werden. In diesem Modus wird eine Konvertierung für jeden Kanal gemacht bis Umwandlung gestoppt ist. Am Ende jeder Konvertierung wird das Ergebnis in die Datenregister ADDR0-ADDR7 für jeden Kanal geschrieben.

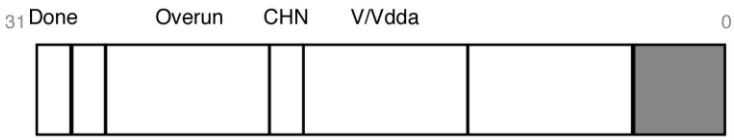


Abb. G.5: ADC-Datenregister

Am Ende jeder Umwandlung wird das Done-Bit gesetzt. Ein Interrupt kann auch erzeugt werden, wenn die entsprechenden Bist im Interrupt-Enable-Register konfiguriert wurden. Die Umrechnung ergibt sich in den gespeicherten V/VDDA-Bits, als ein Verhältnis der Spannung auf dem analogen Kanal. Durch die Spannung am analogen von der Stromversorgung getrenntem Pin. Die Nummer des Kanals, für die die Umwandlung wurde, auch neben dem Ergebnis gespeichert. Dieser Wert wird in der CHN-Feld (vgl. Abb. G.5) gespeichert.

Der A/D-Wandler hat einen zweiten Software-Konvertierungs-Modus. In diesem Fall ist ein Kanal für die Konvertierung ausgewählt und mit den SEL-Bits 0x01 wird der Start der Umwandlung unter Software-Steuerung begonnen. Vgl. Tab G.1 für weitere Konvertierungs-Starteinstellungen. Diese Einstellung speichert die Ergebnisse in das ADDR in der gleichen Weise wie der Hardware-Modus. Das Ende der Umsetzung kann durch einen Interrupt signalisiert werden, oder durch eine Abfrage der Bits im ADDR..

```
int main(void) // Konvertierung im Hardware-Modus
{
VPBDIV = 0x00000002; // PCLK  mit 30MHz
IODIR1 = 0x00FF0000; // P1.16..23 als Outputs
ADCR = 0x00270607; // 10bit Konvertierung und AIN0 auf 3MHz
VICVectCntl0 = 0x00000032; // A/D setzen auf 0
VICVectAddr0 = (unsigned)AD_ISR; // Addresse des IRQ an die VIC binden

VICIntEnable = 0x00040000; // Interrupt aktiviert
  while(1){
    ; // Mache nichts… Test
  }
}

void AD_ISR (void){
  unsigned val,chan;
  static unsigned result[4];
  val = ADCR;
  val = ((val >> 6) & 0x03FF); // ADC-Ergebnis
  chan = ((ADCR >>0x18) & 0x07);
  result[chan] = val;
}
```

Software-Steuerung der Konvertierung:

```
VPBDIV = 0x02; // PCLK mit 30MHz
IODIR1 = 0x00FF0000; // P1.16..23 als Outputs
ADCR = 0x00270601; // ADC mit 10bit und AIN0 auf 3MHz
ADCR |= 0x01000000; // Start ADC

while(1){
do{
  val = ADDR; // Lesen des A/D-Datenregister
}
```

Der A/D-Wandler startet die Konvertierung oder stoppt diese, nach setzen der Start-Bits des SEL-Registers (vgl. Tab. G.1).

Start Bits	Conversion Trigger
0 0 0	Halted
0 0 1	Start Now
0 1 0	PO - 16
0 1 1	PO - 22
1 0 0	MAT 0-1
1 0 1	MAT 0-3
1 1 0	MATT 1-0
1 1 1	MAT 1-1

Tab. G.1: ADC-SEL-Register

G.3 SPI

Die SPI-Schnittstelle [5] ist eine einfache Peripherie. Über diese können Daten seriell geschrieben und gelesen werden, welche auf dem SPI-Bus anliegen. Die SPI-Schnittstelle bietet keine automatische Datentransferfluss- und Zugriffssteuerung. Die Verwaltung und Steuerung muss programmiert werden.

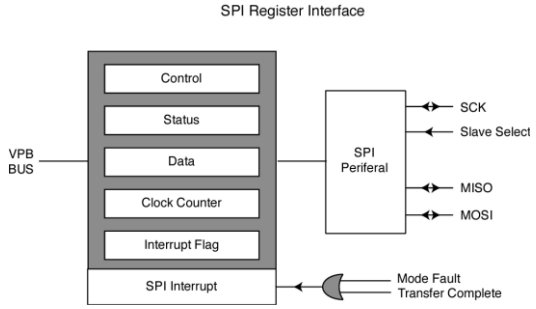


Abb. G.6: SPI-Interface [HITEX]

Die SPI-Peripherie hat vier externe Pins. Einen seriellen Takt SCK, einen Slave-Select-Pin CS und zwei Datenbits für Master-In-Slave-Out MISO und Master-Out-Slave-In MOSI. Die Taktquelle hat eine maximale Frequenz von 400Kbits/s. Im Master-Modus oder im Slave-Modus kann eine externe Taktquelle verwendet werden Der SPI-Bus ist eine serieller High-Speed-Datentransfer und benötigt im Gegensatz zu I²C keine Adressierung. Eine externe Peripherie wird von einem Slave-Select-Pin ausgewählt. Normalerweise, wenn der LPC-2468-Stick im Master-Modus arbeitet, kann dieser mit einem GPIO-Pin als Slave agieren und über SC die gewünschte SPI-Peripherie auswählen. Wenn die SPI-Peripherie im Slave-Modus ist, hat es einen Slave-Select-Eingang. Die beiden Datenübertragung Pins MISO und MOSI sind auf dem Remote-Gerät angeschlossen. Das Schaltbild (vgl. Abb. G.7) zeigt eine typische Konfiguration für eine Verbindung zu einem EEROM Gerät.

Die Programmierer-Schnittstelle für die SPI-Peripherie (vgl. Abb. G.7) hat fünf Register. Das SCK-Zählerregister bestimmt die Baudrate. PCLK wird einfach durch den Wert von SCK entgegen der SPI-Bitrate unterteilt. Dieses Register muss einen minimalen Wert von acht besitzen. Das Steuerregister wird verwendet, um den Betrieb des SPI-Bus zu konfigurieren. Die Größe der Datenübertragung ist standardmäßig auf acht Bits voreingestellt. Allerdings kann durch das Setzen von BitEnable, die Datenpaketgröße zwischen 8Bits bis 16Bits festgelegt werden.

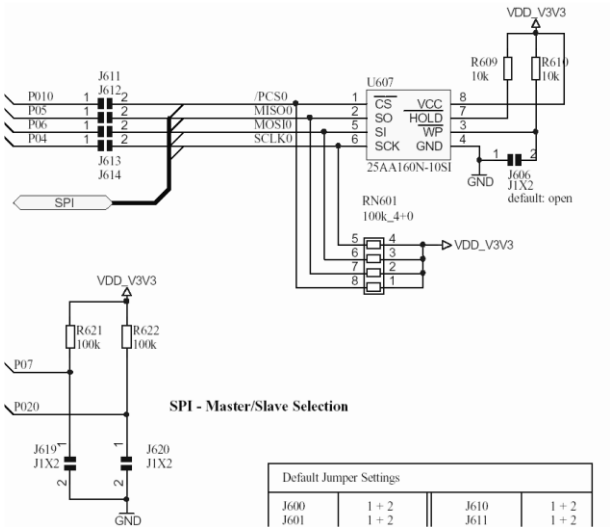


Abb. G.7: SPI-EEPROM [HITEX-Schematics]

Das Schaltbild zeigt, wie eine SPI-Schnittstelle mit einem externen EEROM auf den SPI-Bus geschaltet ist. Es sollte beachtet werden, dass die Pins P0.7 und P0.20 an Pull-Up angeschlossen werden, damit die SPI-Peripherie als Master arbeitet.

BITS	Interrupt Enanble	LSB First Endian Control	Mode Select	Clock Polarity	Clock Phone	Bit Enable

Abb. G.8: SPI-Protokoll

Aufgrund der Einfachheit der SPI-Datenübertragung und des vielfältigen Spektrums an SPI-Peripherien, können der SPI-Takt und Datenleitungen so eingestellt werden, dass diese in verschiedenen Konfigurationen betrieben werden. Als erstes müssen die Polarität und Phase des SCK-Taktes definiert werden. Die Polarität kann aktiv hoch oder aktiv low sein. Die Phase von CPMA skann versetz oder mit der Flanke von SCK erfolgen.

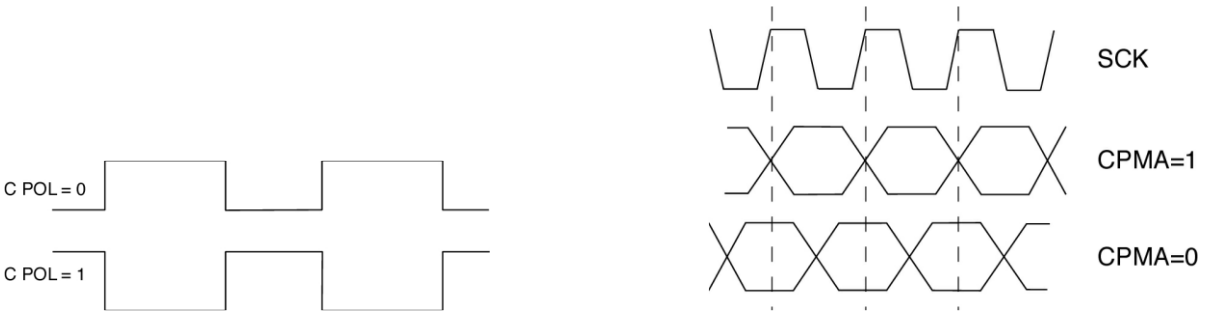


Abb. G.9: SPI-SCK

Für die Datentransferrichtung kann das signifikante Bit am Anfang oder am Ende übertragen werden.

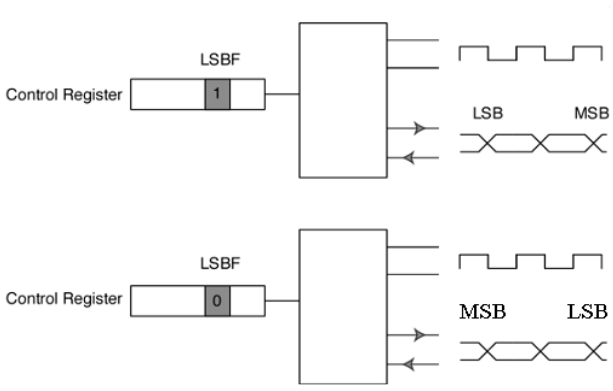


Abb. G.10: SPI-Transfer

Jede dieser Konfiguration muss im Steuerregister anhand der Bitkombinationen eingestellt für die entsprechende Kommunikation eingestellt werden. Sobald die Bitrate eingestellt wurde und die Flusskontrolle konfiguriert ist, kann die Datenübertragung beginnen. Zur Kommunikation mit dem SPI-EEPROM (vgl. Abb. G.7), zuerst ist der GPIO-Pin zu setzen, um den Speicher für die Kommunikation zu enablen. Mit einem Schreibzugriff auf das den SPI-Datenregister, wird ein Byte gesendet und mit einen lesendem Zugriff, wird ein Byte im Register gespeichert, die von der SPI-Peripherie transferiert wurde.

H. PSI-Transferfrequenzen und -latenzzeiten

Sequentielle Messungen der max. Latenzzeiten WCR (Write-Calc-Read) des PSI Kommunikationstransfers und Berechnung der resultierenden Frequenzen. Alle nachfolgend gelisteten Werte wurden am Port P1.0 (Testbit) des LPC2468-Stick mit dem Tektronix eScope TDS-3054-DPO (Konfiguration s. Softwareprojekte) gemessen. Alle Projekte sind auf der [CD-ROM] gespeichert.

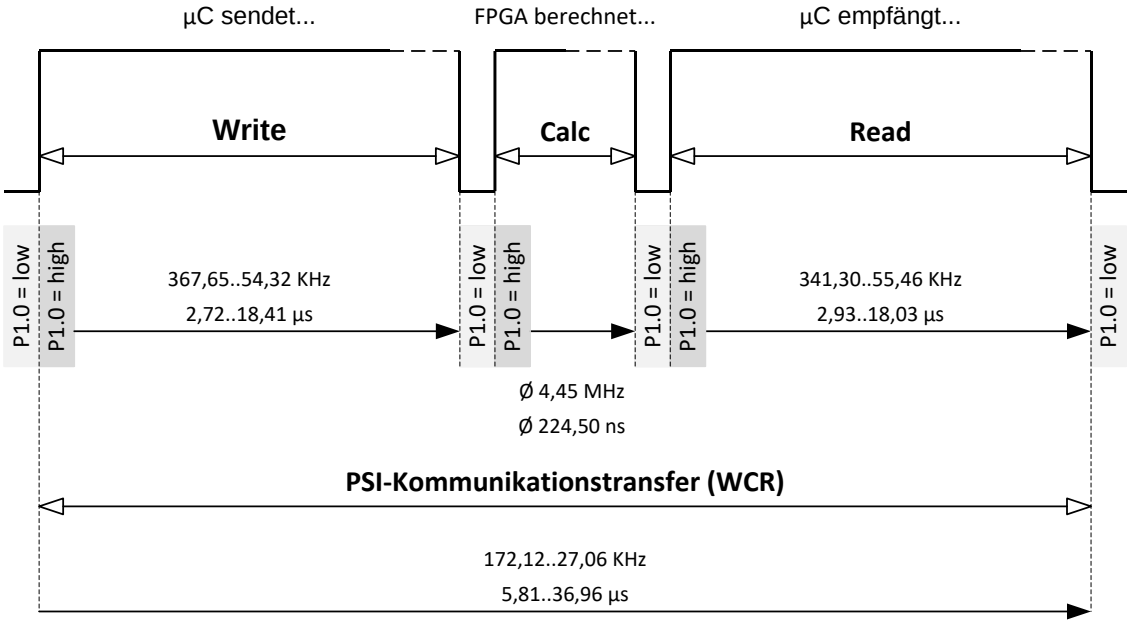


Abb H.1: Messung des PSI-Kommunikationstransfers

PSI-Kommunikation über Funktionsaufrufe (Software-Steuerung)

Projekte/Programmcodes
FPGA: PSI_FSM_EQUILIZER_V1
µC: PSI_Messung1

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**Startup.S setupPLL** MULxDIV [M, N]	**Startup.S CCLKCFG** MUX	**PSI Senden (µC)** Impulsfreq. [KHz / µs]	**Equilizer Berechnung (FPGA)** Taktfrequenz: 50,0 MHz [MHz / ns]	**PSI Empfangen (µC)** Impulsfreq. [KHz / µs]
48	71, 1	5	~54,32 / 18,41	~4,42 / 226,21	~55,46 / 18,03
72	71 ,1	3	~81,57 / 12,26	~4,44 / 224,98	~83,20 / 12,02
Sonderfall					
78	68, 0	5	~104,71 / 9,55	~4,44 / 225,43	~106,72 / 9,37

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**PSI-Transfer 1x Zyklus (WCR)** Impulsfreq. [KHz / µs]	**PSI-Handshake 4-P-H (inkl. RW)** Impulsfreq. [KHz / µs]
48	~27,06 / 36,96	~85,69 / 11,67
72	~40,65 / 24,60	~129,20 / 7,74
Sonderfall		
78	~**52,11 / 19,19**	~165,02 / 6,06

Steuerfunktionsaufrufe ohne dynamische FPGA-Prozessorelementadresse

Projekte/Programmcodes
FPGA: PSI_FSM_EQUILIZER_V1
µC: PSI_Messung1

Einmalige statische Adresszuweisung des FPGA-PEs (PE : Prozessorelement), als Initialisierung vor dem Programmstart:

```
_PSIAdress = 0x00; /* FPGA-Komponentenadresse – deklariert in der config.h */
…
/* Konfiguration der FPGA-PE-Adresse */
setPSIAddress(_PSIAdress);
```

Verkürzung der Latenzzeit um 0,94 µs/Adresskonfiguration (Funktionsaufruf).

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**Startup.S setupPLL** MULxDIV [M, N]	**CCLKCFG** MUX	**PSI Senden (µC)** Impulsfreq. [KHz / µs]	**Equalizer Berechnung (FPGA)** Taktfrequenz: 50,0 MHz [MHz / ns]	**PSI Empfangen (µC)** Impulsfreq. [KHz / µs]
48	71, 1	5	~64,10 / 15,60	~4,44 / 225,08	~65,32 / 15,31
72	71 ,1	3	~96,15 / 10,40	~4,45 / 224,69	~97,94 / 10,21
Sonderfall					
78	68, 0	5	~123,30 / 8,11	~4,43 / 225,50	~124,69 / 8,02

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**PSI-Transfer 1x Zyklus (WCR)** Impulsfreq. [KHz / µs]	**PSI-Handshake 4-P-H (inkl. RW)** Impulsfreq. [KHz / µs]
48	~31,71 / 31,54	~85,69 / 11,67
72	~47,60 / 21,01	~129,20 / 7,74
Sonderfall		
78	~**60,90 / 16,42**	~165,02 / 6,06

Software-Steuerung mit parallelen I/O-Steuerregisterzugriffen

Projekte/Programmcodes
FPGA: PSI_FSM_EQUILIZER_V1
µC: PSI_Messung6

Messung der Laufzeiten des gesamten PSI-Kommunikationstransfers. Steuerung der Transferrichtung der IOBs des SECURITYv2.1-Boards über Funktionsaufrufe:

```
/* SECURITYv2.1-Board IOB1..2 */
SetIOBDir(IOB1, IOB_DIR_B_2_A);
SetIOBDir(IOB2, IOB_DIR_B_2_A);
…
/* SECURITYv2.1-Board IOB1..2 */
SetIOBDir(IOB1, IOB_DIR_A_2_B);
SetIOBDir(IOB2, IOB_DIR_A_2_B);
```

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**Startup.S setupPLL** MULxDIV [M, N]	**Startup.S CCLKCFG** MUX	**PSI Senden (µC)** Impulsfreq. [KHz / µs]	**Equalizer Berechnung (FPGA)** Taktfrequenz: 50,0 MHz [MHz / ns]	**PSI Empfangen (µC)** Impulsfreq. [KHz / µs]
48	71, 1	5	~109,76 / 9,11	~4,43 / 225,60	~109,17 / 9,16
72	71 ,1	3	~164,75 / 6,07	~4,42 / 226,18	~164,75 / 6,07
Sonderfall					
78	68, 0	5	~207,47 / 4,82	~4,41 / 226,62	~205,34 / 4,87

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**PSI-Transfer 1x Zyklus (WCR)** Impulsfreq. [KHz / µs]	**PSI-Handshake 4-P-H (inkl. RW)** Impulsfreq. [KHz / µs]
48	~53,97 / 18,53	~137,17 / 7,29
72	~80,84 / 12,37	~260,42 / 3,84
Sonderfall		
78	~**100,81 / 9,92**	~337,84 / 2,96

Datentransfersteuerung über I/O-Registerzugriffe (Hardware-Steuerung)

Projekte/Programmcodes
FPGA: PSI_FSM_EQUILIZER_V2
µC: PSI_Messung7

Datentransfer über das 4-Phasen-Handshake mit Adressierung des FPGA Prozessorelements. Ersetzung der vier Funktionsaufrufe (setzen der Datentransfersichtung des SECURITYv2.1-Boards):

```
/* SECURITYv2.1-Board IOB1..2 */
SetIOBDir(IOB1, IOB_DIR_A_2_B … IOB_DIR_B_2_A);
SetIOBDir(IOB2, IOB_DIR_A_2_B … IOB_DIR_B_2_A);

/* SECURITYv2.1-Board IOB1..2 (A <- B) */
FIO2CLR |= 0x00000008; /* IOB1, P2[3] */
FIO2CLR |= 0x00000010; /* IOB2, P2[4] */
…
/* SECURITYv2.1-Board IOB1..2 (A -> B) */
FIO2SET |= 0x00000008; /* IOB1, P2[3] */
FIO2SET |= 0x00000010; /* IOB2, P2[4] */
```

Aus dem Projekt PSI_Messung6 durch FastGPIO-Registerzugriffe. Verkürzung der Latenzzeit um 1,5 µs/Funktionsaufruf.

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**Startup.S setupPLL** MULxDIV [M, N]	**Startup.S CCLKCFG** MUX	**PSI Senden (µC)** Impulsfreq. [KHz / µs]	**Equalizer Berechnung (FPGA)** Taktfrequenz: 50,0 MHz [MHz / ns]	**PSI Empfangen (µC)** Impulsfreq. [KHz / µs]
48	71, 1	5	~149,25 / 6,70	~4,46 / 224,31	~161,03 / 6,21
72	71 ,1	3	~228,31 / 4,38	~4,47 / 223,82	~221,24 / 4,52
Sonderfall					
78	68, 0	5	~289,02 / 3,46	~4,44 / 225,02	~295,86 / 3,38

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**PSI-Transfer 1x Zyklus (WCR)** Impulsfreq. [KHz / µs]	**PSI-Handshake 4-P-H (inkl. RW)** Impulsfreq. [KHz / µs]
48	~77,05 / 12,98	~332,23 / 3,01
72	~112,74 / 8,87	~492,61 / 2,03
Sonderfall		
78	~**145,77 / 6,86**	~641,03 / 1,56

Hardware-Steuerung ohne dynamische FPGA-Prozessorelementadresse

Projekte/Programmcodes
FPGA: PSI_FSM_EQUILIZER_V2
µC: PSI_Messung7

Einmalige statische Adresszuweisung des FPGA-PEs (PE : Prozessorelement) als Initialisierung vor dem Programmstart:

```
_PSIAdress = 0x00; /* FPGA-Komponentenadresse – deklariert in der config.h
*/
…
/* Konfiguration der FPGA-PE-Adresse */
FIO1CLR |= 0x0000E000;
FIO1SET = (FIO1PIN & 0xFFFF1FFF) |
          (((_PSIAdress & 0x04) << 11) |
          ((_PSIAdress & 0x02) << 13) |
          ((_PSIAdress & 0x01) << 15));
```

Verkürzung der Latenzzeit um 0,76 µs/Adresskonfiguration.

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**Startup.S setupPLL** MULxDIV [M, N]	**Startup.S CCLKCFG** MUX	**PSI Senden (µC)** Impulsfreq. [KHz / µs]	**Equalizer Berechnung (FPGA)** Taktfrequenz: 50,0 MHz [MHz / ns]	**PSI Empfangen (µC)** Impulsfreq. [KHz / µs]
48	71, 1	5	~189,75 / 5,27	~4,43 / 225,49	~180,51 / 5,54
72	71 ,1	3	~284,90 / 3,51	~4,45 / 224,71	~270,27 / 3,70
Sonderfall					
78	68, 0	5	~367,65 / 2,72	~4,42 / 226,01	~341,30 / 2,93

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**PSI-Transfer 1x Zyklus (WCR)** Impulsfreq. [KHz / µs]	**PSI-Handshake 4-P-H (inkl. RW)** Impulsfreq. [KHz / µs]
48	~92,94 / 10,76	~332,23 / 3,01
72	~139,86 / 7,15	~492,61 / 2,03
Sonderfall		
78	~**172,12 / 5,81**	~641,03 / 1,56

Übersicht der Laufzeiten des PSI-Kommunikationtstransfers (WCR)

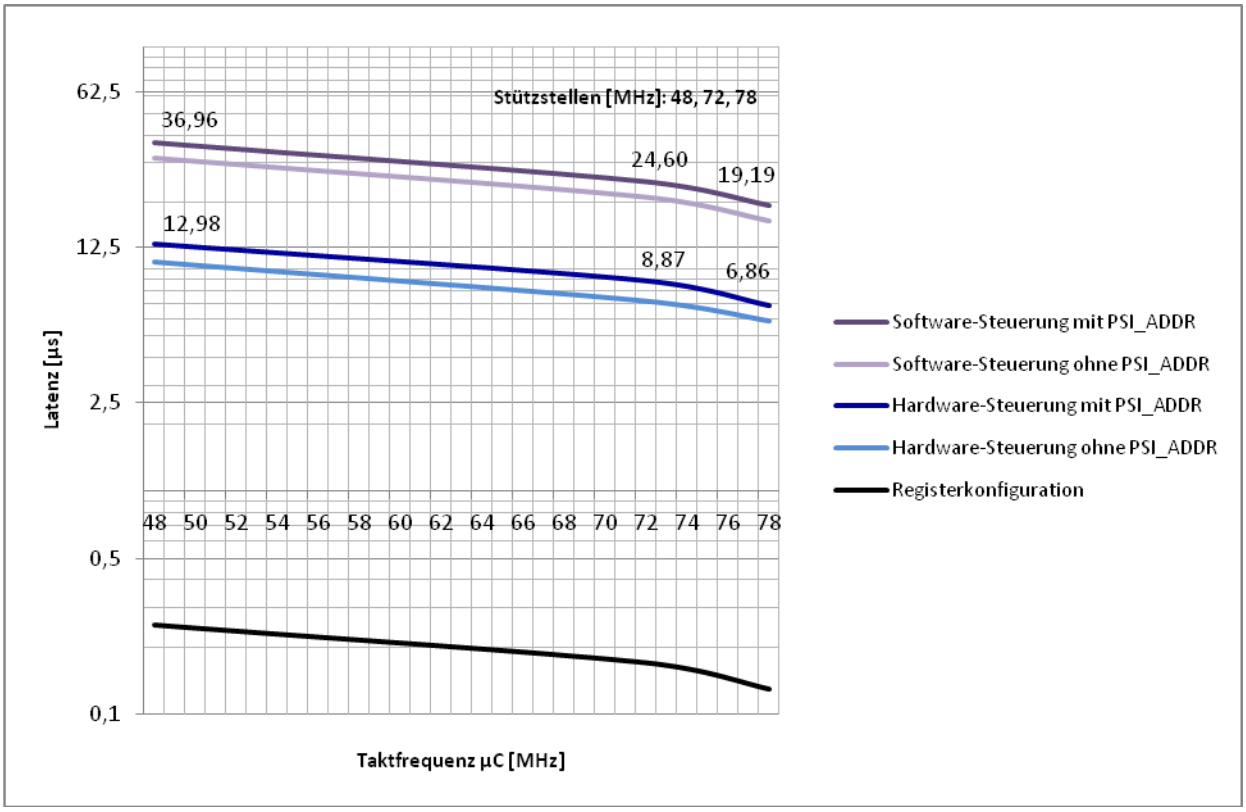


Abb. H.2: Laufzeiten des PSI-Kommunikationstransfers (WCR)

Ausführungszeiten von Funktionen und I/O-Systemregistern

Messungen der max. Latenzzeiten der PSI-Funktionsaufrufe oder äquivalente Registerzugriffe, und Berechnung der resultierenden Frequenzen. Alle nachfolgend aufgelisteten Daten wurden am Port P1.0 und P1.1 (Testbits) des TI-/LPC-Boards (LPC2468-Stick) mit dem Tektronix eScope TDS-3054-DPO (Konfiguration s. Projekte) gemessen.

Für alle Messungen, wurde über die jeweilige pre-compiler-Anweisungen (#define) die Softwaresequenz ausgewählt, und die festgelegten Standardfrequenzkonfigurationen verwendet:

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**Startup.S setupPLL** MULxDIV [M, N]	**Startup.S CCLKCFG** MUX	**Registerkonfiguration P1.0 zu P1.1** Impuls $t_{P0.0,1}$ [ns]
48	71, 1	5	246,50
72	71 ,1	3	164,61
Sonderfall			
78	68, 0	5	127,58

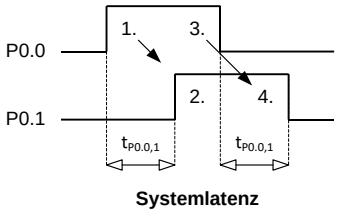


Abb H.3: Ausführungszeit der Systemregister

PSI-Kommunikationstransfer über Software-Steuerung

Projekte/Programmcodes
FPGA: PSI_FSM_EQUILIZER_V2
µC: PSI_Messung8

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**PSI_RW** 1bit set +Impuls	**PSI_REQ** 1bit set +Impuls	**PSI_ACK** 1bit get +Impuls	**PSI_ADDR** 3bit set +Impuls	**PSI_DATA** 10bit set +Impuls	**PSI_DATA** 10bit get +Impuls
48	5,46 µs	1,83 µs	1,75 µs	3,03 µs	4,14 µs	3,15 µs
72	3,64 µs	1,23 µs	1,16 µs	2,02 µs	2,76 µs	2,10 µs
Sonderfall						
78	**2,89 µs**	**903,10 ns**	**892,32 ns**	**1,55 µs**	**2,26 µs**	**1,59 µs**

PSI-Kommunikationstransfer über Hardware-Steuerung

Projekte/Programmcodes
FPGA: PSI_FSM_EQUILIZER_V2
µC: PSI_Messung8

LPC2468-Stick SystemCLK Taktfrequenz [MHz]	**PSI_RW** 1bit set +Impuls	**PSI_REQ** 1bit set +Impuls	**PSI_ACK** 1bit get +Impuls	**PSI_ADDR** 3bit set +Impuls	**PSI_DATA** 10bit set +Impuls	**PSI_DATA** 10bit get +Impuls
48	1,56 µs	493,11 ns	432,64 ns	1,54 µs	2,51 µs	1,46 µs
72	1,04 µs	329,81 ns	288,32 ns	1,03 µs	1,67 µs	974,72 ns
Sonderfall						
78	**817,60 ns**	**263,50 ns**	**225,09 ns**	**806,51 ns**	**1,31 µs**	**763,53 ns**

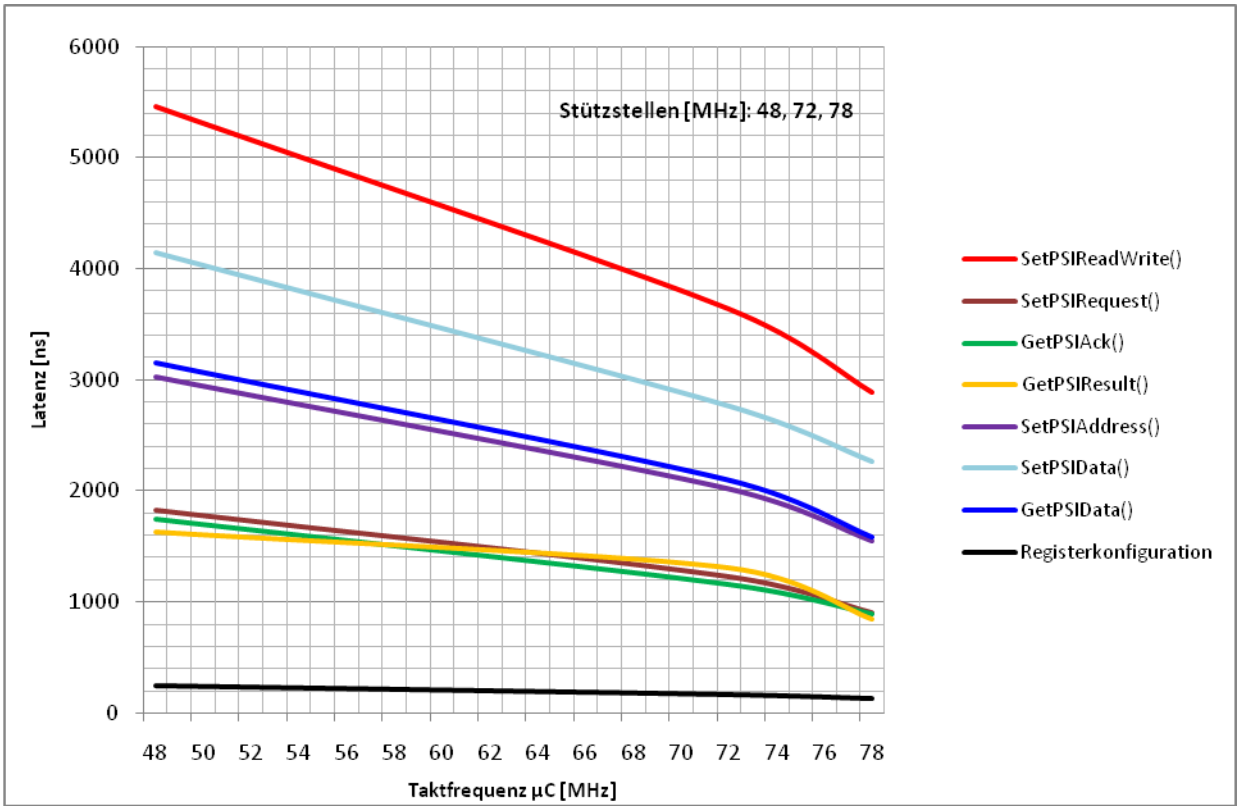


Abb. H.4: PSI-Kommunikationstransfer über Funktionsaufrufe (securepsi.c/.h)

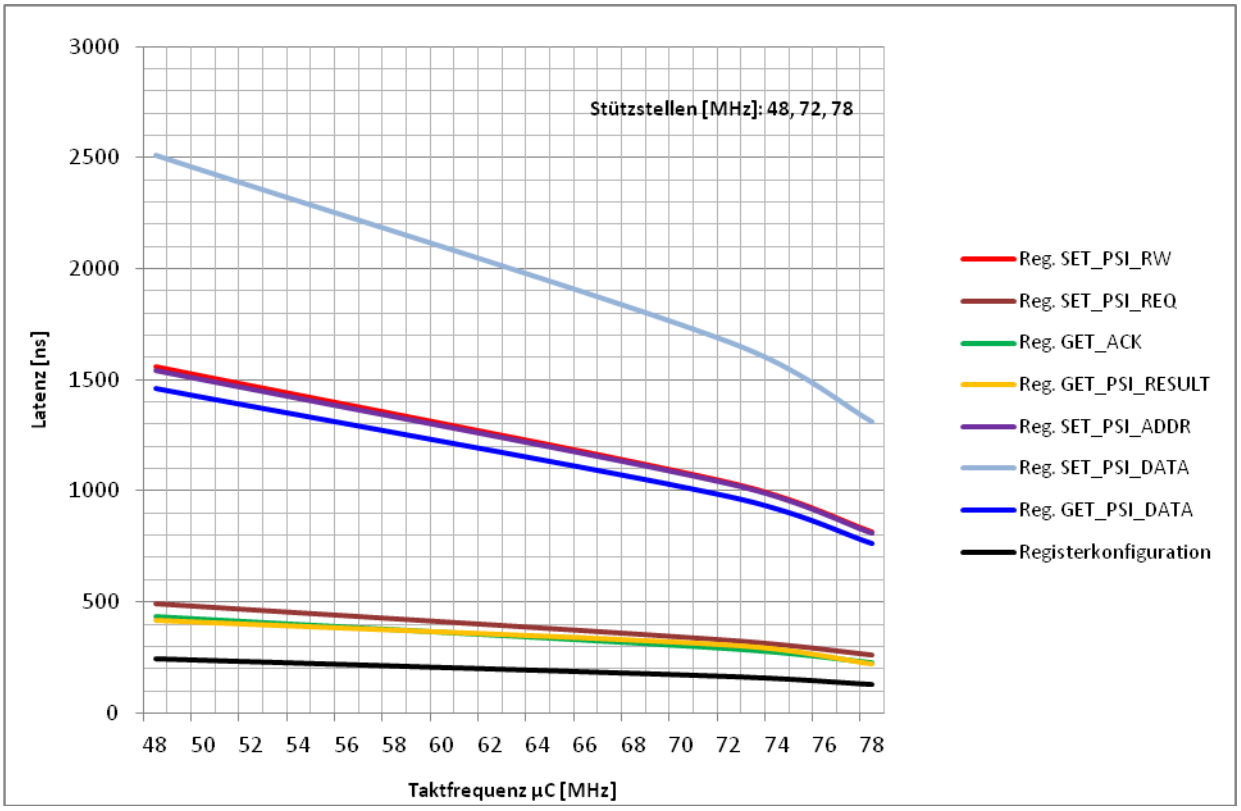


Abb. H.5: PSI-Kommunikationstransfer über Hardware-Steuerung

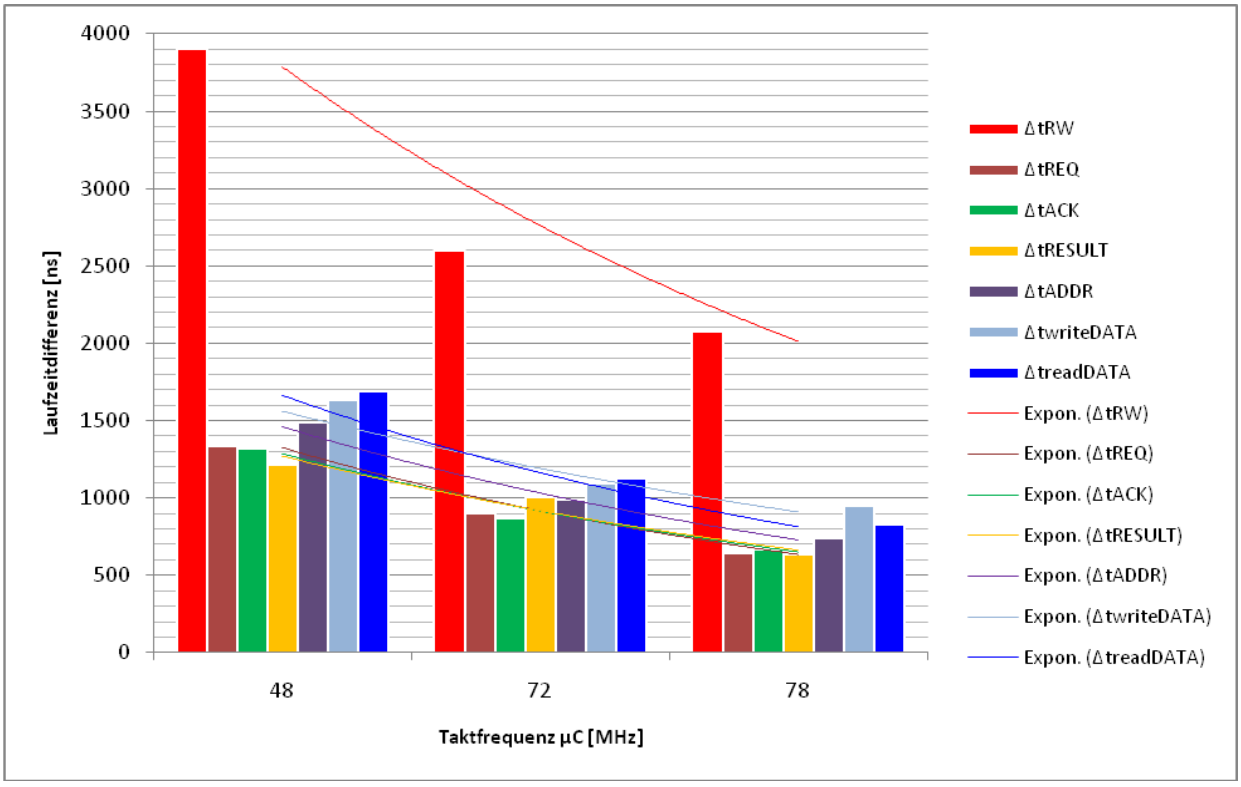


Abb. H.6: Laufzeitdifferenzen der Hardware-/Software-PSI-Kommunikationssequenzen

I. Funktions- und Kommunikationstest der NEXSELv2-Plattform

I.1 Grundkonfiguration der Development-Boards

Für die Testkommunikation der gekoppelten Development-Boards sind nachfolgend aufgeführte Einstellungen im **Offline-Modus** zu überprüfen und oder vorzunehmen. Alle Softwareprojekte sind auf der [CD-ROM] hinterlegt. Dieser Funktionstest ermöglicht fehlerhafte Hardwarekomponenten oder I/O-Pins - die ggf. während der Fertigung oder im Betrieb entstanden sind - festzustellen. Alle Messungen erfolgten ohne Compiler-Optimierungen durch die HiTop53-Software [HITEX].

NEXYS2-Board-Jumper und Schalter:

Name	*Info/Bemerkung*	*Konfiguration*
JP7	POWER SELECT	5-6 (USB)
JP9	MODE	2-3 (ROM)
JP4	CONNECT	offen
JP8	JUMPER STORAGE	1-2
	(One-hot codiert)	
SW7	IOB1	LOW
SW6	IOB2	LOW
SW5	IOB3	LOW
SW4	IOB4	LOW
SW3	IOB5	LOW
	(Bitcodierung)	
SW2	Alle Bits	HIGH
SW1	Gerade/ungerade Bits	LOW
	(Steuersignal)	
SW0	RDY_FPGA	LOW

SECURITYv2.1-Board-Jumper und Schalter:

Name	*Info/Bemerkung*	*Konfiguration*
	(Datentransferrichtung)	(dynamische Ansteuerung)
JP1	IOB1	1
JP2	IOB2	1
JP3	IOB3	1
JP4	IOB4	1
JP5	IOB5	1
	(En-/Disable)	(MUX-Ansteuerung)
JP6	IOB1	1
JP7	IOB2	1
JP8	IOB3	1
JP9	IOB4	1
JP10	IOB5	1
	(Kopplung (16bit Betrieb))	
JP11	IOB1 und IOB2	2 (8bit Betrieb)
SP1	Betrieb: stand-alone	links (3-State-On)

Die aktuelle Schalterkonfiguration des NEXYS2-Boards wird auf der 7-Segementanzeige dargestellt:

Segment	*Funktion*	*Wert*
4 (MSB/links)	Transferrichtung	I (input) oder O (output)
3	IOB-Nr.	1..5 oder E (error), wenn mehr als ein oder kein IOB ausgewählt wurde
2	Bitkombination	A, I oder II (alle, gerade oder ungerade Bits)
1	RDY_FPGA	r , wenn das Steuersignal auf high ist (System aktiv)

Die Datenübertragung der Testprogramme auf die SoC-Komponenten und die Netzspannungsversorgung der Development-Boards erfolgt über USB.

I.2 Softwaretest 1 (OUTput: ARM7™ TDMI und INput: FPGA Spartan 3E)

Signalmusterreihenfolge des Testprogrammablaufs:

```
Step   Zeit      RDY_uC    IOBxDir   IOB1..4Pins   IOB5Pins
       (delay)
1..............TRUE
       2500ms
2..............FALSE
       2500ms
3..............TRUE
4.......................FALSE
       2500ms
5.................................01010101......10101
       2500ms
6.................................10101010......01010
       2500ms
7.................................11111111......11111
       2500ms
8.................................00000001......00001
       1250ms
9.................................00000010......00010
       1250ms
10................................00000100......00100
       1250ms
11................................00001000......01000
       1250ms
12................................00010000......10000
       1250ms
13................................00100000......00000
       1250ms
14................................01000000......00000
       1250ms
15................................10000000......00000
       1250ms

Wiederholung ab Step: 5
```

Programmierung/flashen der SoC-Komponenten:

NEXYS2-Board (.bit): ..\ uC_2_FPGA_-_V_1_1\uc_2_fpga (mit Adept.exe [DIGILENT])

SECURITYv2.1-Board (.htp): ..\ uC_2_FPGA_V_1_0_3\Template_HiTOPv53

Für die Testkommunikation über das SECURITYv2.1-Board Interface ist der 3-State-On-Modus abzuschalten und das Steuersignal per Schalter zu setzen.

NEXYS2-Board-Schalter:

Name	*Info/Bemerkung*	*Konfiguration*
SW0	RDY_FPGA	HIGH

SECURITYv2.1-Board-Schalter:

Name	*Info/Bemerkung*	*Konfiguration*
SP1	Betrieb: System aktivert	rechts

I.3 Softwaretest 2 (INput : ARM7™ TDMI und OUTput: FPGA Spartan 3E)

Signalmusterreihenfolge des Testprogrammablaufs:

```
Step  Zeit     RDY_uC  IOBxDir  IOBx  SECURITYv2    LPC-Port
      (delay)                        Board Jumper  P1[7] P1[6] P1[5] P1[4] P1[3] P1[2] P1[1]
P1[0]
                                     J8[1|2]
1.............TRUE
      2500ms
2.............FALSE
      2500ms
3.............TRUE
4.....................FALSE
      2500ms

Nachfolgend manueller Betrieb (über die NEXYS2-Schaltereinstellungen)

                                1     [1]           0/1   0/1   0/1   0/1   0/1   0/1   0/1
0/1
                                2     [1]           0/1   0/1   0/1   0/1   0/1   0/1   0/1
0/1
                                3     [2]           0/1   0/1   0/1   0/1   0/1   0/1   0/1
0/1
                                4     [1]           0/1   0/1   0/1   0/1   0/1   0/1   0/1
0/1
                                5     [1]                               0/1   0/1   0/1   0/1
0/1
```

Programmierung/flashen der SoC-Komponenten:

NEXYS2-Board (.bit): ..\ FPGA_2_uC_-_V_1_1\fpga_2_uc (mit Adept.exe [DIGI])

SECURITYv2.1-Board (.htp): ..\ FPGA_2_uC_V_1_0_2\Template_HiTOPv53

Für die Testkommunikation über das SECURITYv2.1-Board Interface ist der 3-State-On-Modus abzuschalten und das Steuersignal per Schalter zu setzen.

NEXYS2-Board-Schalter:

Name	*Info/Bemerkung*	*Konfiguration*
SW0	RDY_FPGA	HIGH

SECURITYv2.1-Board-Jumper und Schalter:

Name	*Info/Bemerkung*	*Konfiguration*
JP8	(Test: IOB1,2,4,5) IOB3 En-/Disable	2 (deaktiviert)
JP8	(Test: IOB3) IOB3 En-/Disable	1 (aktiviert)
SP1	Betrieb: System aktivert	rechts

J. Laborgeräte

Tektronix – TDS 544A

Abb. J.1

500MHz 4-Kanal Digital-Speicher-Oszilloskop mit 3GS/s für den Laboreinsatz.

Bedienungshandbuch und technische Daten:
..\ Messgeräteanleitungen \ Tektronix \
Tektronix.pfd

Technology Limited – picoScope ADC-200 VI

Abb. J.2

10MHz 8bit 2-Kanal eScope mit 20MS/s für den Labor- oder Feldeinsatz.

Manual und Data Sheets:
..\ Messgeräteanleitungen \ Technology_Limited \
adc2xx.pfd

FLUKE – UDM 177 True RMS

Abb. J.3

Echteffektivwert Universal-Digitalmultimeter nach EN61010 Kat. III 1000V und IV 600V bei DC ±(0,09% + 2) für Feld- und Laboreinsatz.

Bedienungshandbuch und technische Daten:
..\ Messgeräteanleitungen \ FLUKE \
0900766b800bcd8e.pfd

TENMA – 72-2055

Abb. J.4

1000V Universal-Digitalmultimeter für Feld- und Laboreinsatz.

Reference und Data Sheets:
..\ Messgeräteanleitungen \ TENMA \
UM72-2055.pfd

Philips – PM 2525

Abb. J.5

Universal-Digitalmultimeter für Feld- und Laboreinsatz.

Manual und Data Sheets:
..\ Messgeräteanleitungen \ Philips \
PHILI_252545818.pfd

KONTRON – 8020

Abb. J.6

Universal-Funktionsgenerator für Feld- und Laboreinsatz.

Manual und Data Sheets:
..\ Messgeräteanleitungen \ KONTRON \
KONTRON_8020.pfd

Rohde & Schwarz – NGT 20

Abb. J.7

3fach DC-Netzversorgungsgerät mit zweimal 0..20V/0..1A und einmal 0..6V/0..5A OVP für den Laboreinsatz.

Manual und Data Sheets:
..\ Messgeräteanleitungen \ Rohde&Schwarz \
Power_Supplies_Component_bro_en.pfd

K. TI-Development-Variante

Das MODULsystem umfasst zwei Basiseinheiten und verschiedene Peripherie-Steckkartenmodule (vgl. Abb. K.1). Die Basiseinheiten sind technisch durch eine Schutzschaltung vor fehlerhafter Ansteuerung oder einem Kurzschluss geschützt. Durch die Halbleitersockel der Basiseinheiten, sind beschädigte ICs oder SoCs auswechselbar. Die modulare Technik und Verwendung von steckbaren Einheiten gewährleistet eine maximale Flexibilität des Gesamtsystems. Dieses ist unabhängig von Stückzahlen oder proprietären Development-Boards anderer Hersteller und kann den unterschiedlichen Lehrinhalten der HAW, durch die Erweiterung von Software-/Hardware-Modulen (ggf. Studien- oder Abschlussarbeiten), entsprechend angepasst werden. Die Netzspannungsversorgung kann separat oder über ein Zusatzmodul alle verbundenen Systemeinheiten versorgen. Grafisch ist eine mögliche Auswahl an Steckkartenmodulen dargestellt.

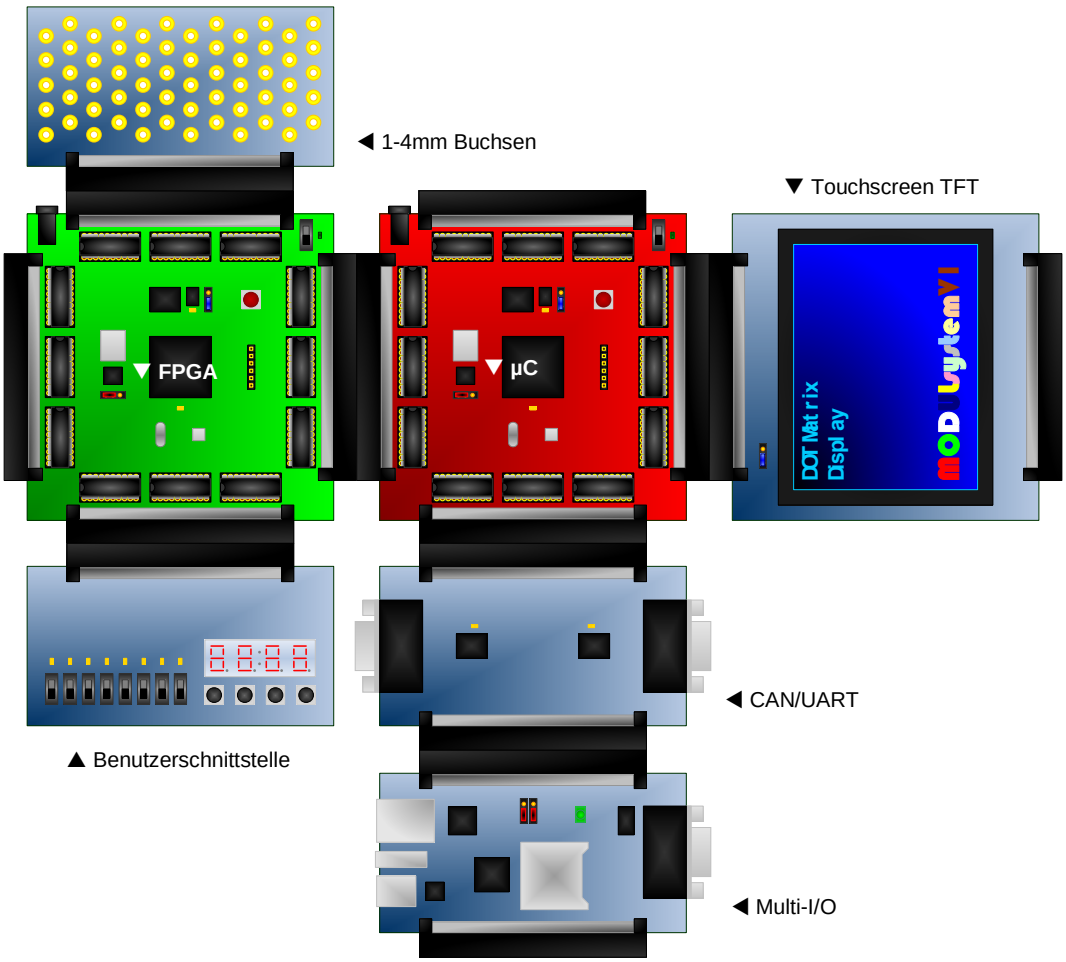


Abb. K.1: TI Development-Board-Variante

L. CD-ROM-Inhalt

Der im Rahmen dieser Bachelorarbeit erzeugte Quellcode und weitere Dateien befinden sich auf der CD-ROM.

3E-Verzeichnisstruktur

+Datenblätter
    -Developments-Boards
    -Hersteller
+Erprobungsplattform
    -FPGA
    -Mikrocontroller
-Fotos
+Grafiken
    -Messungen
+Messgerätanleitungen
    -FLUKE
    -KONTRON
    -Philips
    -Rodhe&Schwarz
    -Technology_Limited
    -Tekronix
    -TENMA
+Projekte
    -FPGA
    -Mikrocontroller
+SoC-Komponenten
    -ARM7_TDMI_(LPC2468FET208)
    -FPGA_Spartan_3E_(XC3S1200E)
+Software
    -FPGA
    -Mikrocontroller
-Tabellen

Versicherung über Selbständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 20. März 2011	
Ort, Datum	Unterschrift