OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

Computer Architektur

Sommersemester 2025

Projektzielsetzung

Version 1.0

Teammitglieder:

Fabian Becker

Jendrik Jürgens

Nicolas Koch

Franz Krempl

Daniel Sowada

Michael Specht

Professor:
Dr. Daniel Münch

Abgabedatum:
30.04.2025 11:30 Uhr

27. April 2025

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

1. Projektzielsetzung

Projekttitel:
Integriertes Demosystem für Sensor- und Displayansteuerung auf FPGA-Basis
Teammitglieder:
Fabian Becker, Jendrik Jürgens, Nicolas Koch, Franz Krempl, Daniel Sowada, Michael Specht
Projektbeschreibung:
Ziel des Projekts ist die Entwicklung und Integration eines funktionsfähigen Demosystems, das zwei getrennte Komponenten – den Sonarsensor (PMOD MAXSONAR) und das LCD-Display (PMOD CLP) – auf einer gemeinsamen FPGA-Plattform (Digilent Arty A7-100) vereint. Die Messwerte des Sensors sollen in Echtzeit auf dem Display ausgegeben werden. Dazu werden eigene IP-Cores für beide Komponenten entworfen, implementiert, getestet und in die Hardwareplattform integriert.

Geplante Arbeitspakete und Zuständigkeiten:

1. Systemintegration bestehender Demosysteme
   - Integration der Software- und Hardware-Demoprojekte zu einem Gesamtsystem (HW & SW)

     Zuständig: Alle Teammitglieder
2. Entwicklung Custom IP für Sensor (UART / MAXSONAR)
   - Konzept (Blockdiagramm, Registermapping) auf Basis der `tut_int` Vorlage
   - Reduzierte Funktionalität orientiert an Xilinx UART Lite IP

     Zuständig: Fabian Becker, Nicolas Koch
3. Entwicklung Custom IP für Display (PMOD CLP, Nibble-Mode)
   - Umsetzung von Timinganforderungen in Hardware
   - Anzeige von Daten auf LCD über FSM und spezifizierte Steuerregister

     Zuständig: Jendrik Jürgens, Michael Specht
4. Erstellung und Test der IP-Testbenches (Core und AXI)
   - Entwicklung mit Fokus auf Polling (Interrupt optional bei Zeitreserve)
   - Simulation und Validierung des Verhaltens

     Zuständig: Alle Teammitglieder
5. Treibersoftware
   - Schreiben von Treibern für die beiden IPs unter Verwendung der SW-Templates von `tut_int`
   - SW-basierte Initialisierung und Datentransfer

Zuständig: Franz Krempl, Daniel Sowada

6. Integration in vollständiges Demosystem
   - Zusammenführung aller Komponenten zu einem lauffähigen System
   - Validierung auf der realen Hardwareplattform

   Zuständig: Alle Teammitglieder

Systemblockbild:

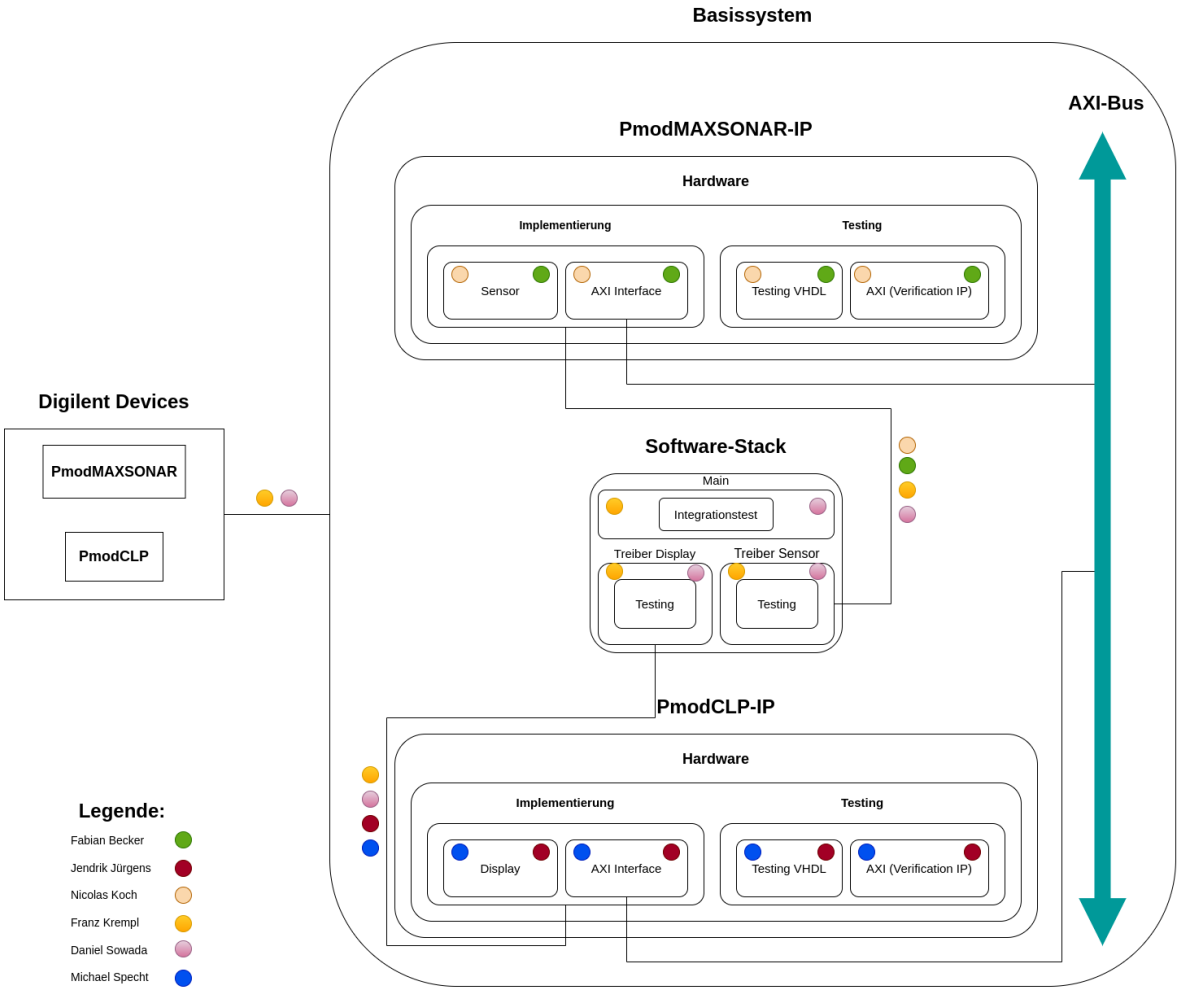

Abbildung 1.1: Systemblockbild

- **Exakte Timings für das Display (PmodCLP) in VHDL:**
  Die Ansteuerung im Nibble-Mode erfordert die präzise Umsetzung aller Timingbedingungen in einer FSM, da keine automatische Verzögerung durch die CPU gegeben ist.
- **Entwicklung AXI4-Lite-kompatibler IP-Cores:**
  Für Sensor und Display werden eigenständige IPs mit klar strukturiertem Registermapping und AXI-Anbindung erstellt, basierend auf einer gemeinsamen IP-Vorlage.
- **Zuverlässiger Datenfluss zwischen Sensor, CPU und Display:**
  Die Software muss synchronisiert mit den IPs arbeiten, um Sensordaten korrekt auszulesen und anzuzeigen – inklusive Fehlerbehandlung und Statusabfrage.

- **Simulation und Verifikation:**
  Funktion und Schnittstellen werden über VHDL-Testbenches (Core + AXI) geprüft, um Designfehler frühzeitig zu erkennen.

Ziel:
Ein lauffähiges Demosystem mit eigenentwickelten, erweiterbaren IP-Cores, das Messwerte des Sonar Sensors auf einem LCD-Display ausgibt – mit Tests und einer funktionalen Ergebnisvorführung.

2. PmodCLP

Der PmodCLP besteht aus einem Samsung KS0066 LCD Controller und einem Sunlike LCD Panel, worüber Informationen dargestellt werden können. Es ist möglich 32 Positionen auf dem 16x2 LCD Panel zu nutzen. Pro Position werden die Zeichen dabei mit einer Auflösung von 5x8 angezeigt.

Das System besteht im Wesentlichen aus drei Komponenten. Der character-generator ROM (CGROM) hält 192 vordefinierte Zeichen, darunter 93 ASCII Charaktere. Anschaulich gesehen können die Zeichen über eine matrixartige Struktur indiziert werden, welche im Datenblatt festgelegt ist. Neben den nicht-volatilen Daten im CGROM ist es möglich bis zu 8 eigene Zeichen volatil im character-generator RAM (CGRAM) zu halten. Um nun Zeichen aus diesen beiden Repositories auf dem Panel anzeigen zu können, gibt es den data RAM (DDRAM). Hier können bis zu 80 Zeichencodes gespeichert werden. Er fungiert als Indexspeicher für Daten innerhalb des CGROM oder CGRAM. Wird ein Index aus der matrixartigen Struktur in den DDRAM geladen, erscheint das entsprechende Zeichen auf dem Display.

Das Display selbst verfügt über 2 Zeilen á 16 Positionen. Insgesamt stehen jedoch nicht 32 Speicherplätze zur Verfügung, sondern 39, um beispielsweise Scrolling zu verwenden.

Wichtige Schnittstellen des Samsung KS0066 LCD Controller sind

- *DB4-DB7*: Datenbits im Nibble-Mode zur Codierung von Befehlen/Zeichen
- *RS (Register Select)*: High für Daten, Low für Instruktionen
- *RW (Read/Write)*: High = Read, Low = Write
- *E (Enable)*: High für Read, Falling Edge für Write

Um diese nutzen zu können wird folgendes Mapping auf dem FPGA hinterlegt:

```
set_property -dict {PACKAGE_PIN D13 IOSTANDARD LVCMOS33}[get_ports{clp_db_tri_io[4]}];
    #db04
set_property -dict {PACKAGE_PIN B18 IOSTANDARD LVCMOS33}[get_ports{clp_db_tri_io[5]}];
    #db05
set_property -dict {PACKAGE_PIN A18 IOSTANDARD LVCMOS33}[get_ports{clp_db_tri_io[6]}];
    #db06
set_property -dict {PACKAGE_PIN K16 IOSTANDARD LVCMOS33}[get_ports{clp_db_tri_io[7]}];
    #db07
set_property -dict {PACKAGE_PIN E2 IOSTANDARD LVCMOS33}[get_ports{clp_cb_tri_o[0]}];
    #lcd_rs
set_property -dict {PACKAGE_PIN D2 IOSTANDARD LVCMOS33}[get_ports{clp_cb_tri_o[1]}];
    #lcd_rw
set_property -dict {PACKAGE_PIN H2 IOSTANDARD LVCMOS33}[get_ports{clp_cb_tri_o[2]}];
    #lcd_e
```

Listing 2.1: Pin-Zuordnung im Constraints-File

Die Zuordung in Aufzählung 2.1 greift auf zwei Header des PmodCLP-Boards zurück. Die Daten-Bits db04-db07 sind an die untere Hälfte des Headers J1 gebunden. Die Steuersignale Register Select, Read/Write und Enable hingegen an Header J2.

2.1. Ansatz Custom IP

Im ersten Schritt soll die IP funktional fertiggestellt werden. Sie soll dabei mittels Polling eingesetzt werden. Sobald die Funktionalität des Gesamtsystems vollumfänglich gegeben ist, wird anstatt Polling über Interrupts kommuniziert.

Das Projektteam hat sich auf folgenden Entwurf geeinigt:

- *Submodul 1*: Timing Controller
  Um das Display ordnungsgemäß betreiben zu können, müssen diverse Timing-Constraints beachtet werden. Dies kann bspw. mit einem Zähler, der als Timer in Abhängigkeit vom Systemtakt fungiert, umgesetzt werden.

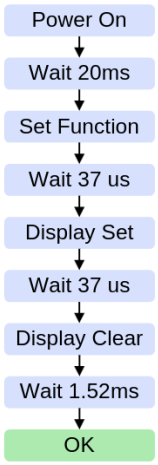

Abbildung 2.1: Startup Sequence

  Graphik 2.1 zeigt beispielhaft die Initialisierung des Displays und die damit verbundenen Timing-Anforderungen.

- *Submodul 2*: LCD Controller
  Hier werden Befehle interpretiert und die Daten in die entsprechenden Register geschrieben, um die Werte anschließend auf dem Display darzustellen.

- *Submodul 3*: AXI Slave Interface
  Nachdem die Zuverlässigkeit der IP mittels Tests sichergestellt wurde, soll die IP an den internen Systembus (AXI) angebunden werden.

Das Registermapping ist an *at_ doc.pdf* aus *02b_ tut_ vhdl_ v03* angelehnt.

2.2. Registermapping

2.2.1. I/Os

Signal Name	I/O	Initial State	Description
ap_clk(s00_axi_aclk)	I	NA	AXI Clock
ap_rst_n (s00_axi_aresetn)	I	NA	AXI Reset, active-Low
s_axi_control* (s00_axi*)	NA	NA	AXI4-Lite Slave Interface signals
interrupt	I	0x0	Indicates that the condition for an interrupt has occurred. (new sensor value available) 0 = No interrupt has occurred 1 = Interrupt has occurred
db4_7	O	0xFF	4 data bits, necessary in nibble mode.
register_select	O	0x1	Register Select: High for Data Transfer, Low for Instruction Transfer
read_write	O	0x1	Read/Write signal: High for Read mode, Low for Write mode
read_write_enable	O	0x1	Read/Write Enable: High for Read, falling edge writes data

Tabelle 2.1: PmodCLP Überblick I/O

2.2.2. Registerbereich

Address Offset	Register Name	Description
0x00	GCSR	General/Global Control and Status Register
0x04	GIER	Global Interrupt Enable Register
0x08	IPIER	IP Interrupt Enable Register
0x0C	IPISR	IP Interrupt Status Register
0x10	IDR	ID Register
0x14	VERR	Version Register
0x18	SCSR0	Special Control and Status Register
0x1C	CCR	Character Control Register
0x20	CDR	Character Data Register
0x24	DCR	Display Control Register

Tabelle 2.2: PmodCLP Controller Register Space Overview

Bit	Name	Access Type	Reset Value	Description
0x00 GCSR - General/Global Control and Status Register				
0	ap_start	R/W	0	Asserted when the kernel can start processing data or format display. Cleared on handshake with ap_done being asserted.

Continued on next page

Bit	Name	Access Type	Reset Value	Description
1	ap_done	R	0	Asserted when the kernel has completed processing data (with or without error). Cleared on read.
2	ap_idle	R	0	Asserted when the kernel is idle.
3	*reserved (ap_ready)*	R	0	Asserted by the kernel when it is ready to accept new data (used only by AP_CTRL_CHAIN)
4	*reserved (ap_continue)*	R/W	0	Asserted by the XRT to allow kernel keep running (used only by AP_CTRL_CHAIN)
5:6	reserved			
7	auto_restart *(optional)*	R/W	0	Used to enable automatic kernel restart. This bit determines whether the display reloads the last sensor value and continues running or clears the display.
31:8	reserved			

Tabelle 2.3: General/Global Control and Status Register (GCSR)

Bit	Name	Access Type	Reset Value	Description
0x04 GIER - Global Interrupt Enable Register				
0	gie	R/W	0	When asserted, along with the IP Interrupt Enable bit, the interrupt is enabled.
31:1	reserved			

Tabelle 2.4: Global Interrupt Enable Register (GIER)

Bit	Name	Access Type	Reset Value	Description
0x08 IPIER - IP Interrupt Enable Register				
0	ipie	R/W	0	When asserted, along with Global Interrupt Enable bit, the interrupt is enabled. (default: uses the internal ap_done signal to trigger an interrupt)
31:1	reserved			

Tabelle 2.5: IP Interrupt Enable Register (IPIER)

Bit	Name	Access Type	Reset Value	Description
0x0C IPISR - IP Interrupt Status Register				
0	ipis	R/W	0	Toggle on write. (write 1 to clear(W1C))
31:1	reserved			

Tabelle 2.6: IP Interrupt Status Register (IPISR)

Bit	Name	Access Type	Reset Value	Description
0x10 IDR - ID Register				
31:0	ID	R	0x80010744	Distinct ID for PmodCLP Controller

Tabelle 2.7: ID Register (IDR)

Bit	Name	Access Type	Reset Value	Description
0x14 VERR - Version Register				
31:0	VER	R	0x80001000	Version

Tabelle 2.8: Version Register (VERR)

Bit	Name	Access Type	Reset Value	Description
0x18 SCSR0 - Special Control and Status Register				
0	lcd_initialized	R	0	1 indicates the LCD has been properly initialized and is ready for commands.
8:1	lcd_state *(for debugging)*	R	0x00	Current state of LCD controller. (init: 0x01, ready: 0x02, write: 0x03, read: 0x04, clear display: 0x05, return home: 0x06, display control: 0x07, display line switch: 0x08)
9	lcd_error_flag	R	0	Indicates an error occurred during last operation. In combination with 'write_char' and 'read_char' two errors can occur: invalid symbol (write) or invalid ddram pos (read).
31:10	reserved			

Tabelle 2.9: Special Control and Status Register (SCSR0)

Bit	Name	Access Type	Reset Value	Description
0x1C CCR - Character Control Register				
7:0	ddram_pos_last_written	R	0xFF	Last updated DDRAM address/position for character (00H-27H for line 1, 40H-67H for line 2)
15:8	ddram_pos_to_read	R/W	0xFF	DDRAM address/position for character (00H-27H for line 1, 40H-67H for line 2) for software testing purpose (e.g. read symbol which was written before).
16	write_char	R/W	0	Write 1 to initiate character write either to specified DDRAM position (if ddram_pos is in valid range) or default last_ddram_pos.
17	read_char	R/W	0	Write 1 to read character either from specified DDRAM position (if ddram_pos is in valid range) or default last_ddram_pos.
31:18	reserved			

Tabelle 2.10: Character Control Register (CCR)

Bit	Name	Access Type	Reset Value	Description
0x20 CDR - Character Data Register				
7:0	symbol_to_write	R/W	0xFF	Character symbol to write to LCD.
15:8	symbol_read	R	0x3F	Character symbol which was read ('?' is default value if no read operation was triggered before). For software testing purpose.
31:16	reserved			

Tabelle 2.11: Character Data Register (CDR)

Bit	Name	Access Type	Reset Value	Description
0x24 DCR - Display Control Register				
0	clear_display	R/W	0	Write 1 to clear the display
1	return_home	R/W	0	Write 1 to return cursor to home position
2	cursor_on	R/W	0	Enable cursor visibility (0 = off, 1 = on)
3	cursor_blink	R/W	0	Enable cursor blinking (0 = no blink, 1 = blink)
4	display_line *(optional)*	R/W	0	Indicates selected line on display (0 = line 1, 1 = line 2)

Continued on next page

Bit	Name	Access Type	Reset Value	Description
31:5	reserved			

Tabelle 2.12: Display Control Register (DCR)

3. PmodMAXSONAR

Der Pmod MAXSONAR besitzt einen MaxBotix® LV-MaxSonar®-EZ1™ Ultraschall Sensor, welcher Entfernungen von 15cm - 648cm mit einer Genauigkeit von 2.54cm messen kann. Nach einer Power-On-Phase von 250 ms und einer weiteren Wartezeit von ca. 100 ms für die Kalibrierung und erste Messung, kann der Sensor in alle 49 ms eine Distanzmessung durchführen.

Der Sensor sendet ein Ultraschall-Signal aus, welches von einem Objekt reflektiert wird. Der Sensor misst die Zeit, die das Signal benötigt, um zum Sensor zurückzukehren und berechnet daraus die Entfernung zum Objekt. Dieser Wert wird auf drei Arten an den Ausgängen dargestellt, zum einen als Spannungssignal, als Pulsweite oder als digitale Zahl per UART. In diesem Projekt wird die Entfernung per UART abgegriffen.

Die wichtigen Schnittstellen sind daher:

- *TX (Transmit Data)*: Pin zur Initialisierung der Sensorkalibrierung und Distanzmessungen
- *RX (Receive Data)*: Empfang des Distanzwertes als 5 ASCII-Zeichen mit dem Format "Rxxx\r"; xxx entspricht dem Distanzwert in zoll

Um den Sensor nutzen zu können, müssen die Pins TX und RX mit dem FPGA verbunden werden.

```
set_property -dict {PACKAGE_PIN U16 IOSTANDARD LVCMOS33} [get_ports {tx_out}];
    #pmodmaxsonar rx
set_property -dict {PACKAGE_PIN V15 IOSTANDARD LVCMOS33} [get_ports {rx_in}];
    #pmodmaxsonar tx
```

Listing 3.1: Pin-Zuordnung im Constraints-File

3.1. Ansatz Custom IP

Im ersten Schritt soll die IP funktional fertiggestellt werden. Sie soll dabei mittels Polling eingesetzt werden. Sobald die Funktionalität des Gesamtsystems vollumfänglich gegeben ist, wird anstatt Polling über Interrupts kommuniziert.

Das Projektteam hat sich auf folgenden Entwurf geeinigt:

- *Submodul 1*: Startup & Calibration Timer
  Während der Startup und Kalibrierungsphase können keine Messungen durchgeführt werden. Daher wird ein Timer implementiert, der die Zeit bis zur ersten Messung überwacht.
- *Submodul 2*: UART Receiver (FSM)
  Mittels einer FSM wird der UART Empfang realisiert, dieser empfängt jeweils 1 Byte und meldet den Empfang an den ASCII-Decoder.

Die Baudrate mit Oversampling wird durch einen Clockdivider aus dem Systemtakt generiert. Bei einem Fehler im Empfangsprozess wird ein Fehler-Flag gesetzt, und die IP beendet den Vorgang mittels des *ap_done* Signals.

- *Submodul 3*: ASCII Decoder (FSM) *optional*
  Der ASCII Decoder empfängt stückweise die 5 ASCII-Zeichen im Format "R$x_0x_1x_2\backslash$r"
  $x_0 \in \{0,..,2\}, x_{1,2} \in \{0,...,9\}$.
  Wird dieses Format verletzt, wird ein Fehler-Flag gesetzt, und die IP beendet den Vorgang mittles des *ap_done* Signals.
  Nach erfolgreicher Konvertierung steht die Distanz als 8-Bit Zahl in der Einheit zoll (inch) zur Verfügung.

- *Submodul 4*: AXI Slave Interface
  Nachdem die Zuverlässigkeit der IP mittels Tests sichergestellt wurde, soll die IP an den internen Systembus (AXI) angebunden werden.

Das Registermapping ist an *at_doc.pdf* aus *02b_tut_vhdl_v03* angelehnt.

3.2. Registermapping

3.2.1. I/Os

Signal Name	**I/O**	**Initial State**	**Description**
ap_clk(s00_axi_aclk)	I	NA	AXI Clock
ap_rst_n (s00_axi_aresetn)	I	NA	AXI Reset, active-Low
s_axi_control* (s00_axi*)	NA	NA	AXI4-Lite Slave Interface signals
interrupt	I	0x0	Indicates that the condition for an interrupt has occurred. (new sensor reading) 0 = No interrupt has occurred 1 = Interrupt has occurred
rx_in	I	NA	UART receive from sensor
tx_out	O	1	UART transmit to sensor

Tabelle 3.1: PmodMAXSONAR I/O

3.2.2. Registerbereich

Address Offset	**Register Name**	**Description**
0x00	GCSR	General/Global Control and Status Register
0x04	GIER	Global Interrupt Enable Register
0x08	IPIER	IP Interrupt Enable Register
0x0C	IPISR	IP Interrupt Status Register
		Continued on next page

Address Offset	Register Name	Description
0x10	IDR	ID Register
0x14	VERR	Version Register
0x18	SCSR0	Special Control and Status Register
0x1C	DIST0	Distance Value Register
0x20	UCSR	UART Control and Status Register
0x24	ADSR	ASCII Decoder Status Register *(optional)*

Tabelle 3.2: PmodMAXSONAR Register Space Overview

Bit	Name	Access Type	Reset Value	Description
0x00 GCSR - General/Global Control and Status Register				
0	ap_start	R/W	0	Asserted when the kernel is able to process data. Cleared on handshake with ap_done being asserted.
1	ap_done	R	0	Asserted when the kernel has completed a sensor reading (with or without error). Cleared on read.
2	ap_idle	R	0	Asserted when the kernel is idle.
3	*reserved (ap_ready)*	R	0	Asserted by the kernel when it is ready to accept new data (used only by AP_CTRL_CHAIN)
4	*reserved (ap_continue)*	R/W	0	Asserted by the XRT to allow kernel keep running (used only by AP_CTRL_CHAIN)
5:6	reserved			
7	auto_restart	R/W	0	Used to enable automatic kernel restart. This bit determines whether only one sensor reading is processed or the sensor reading is continously updated. 0 = single reading 1 = free running mode
31:8	reserved			

Tabelle 3.3: General/Global Control and Status Register (GCSR)

Bit	Name	Access Type	Reset Value	Description
0x04 GIER - Global Interrupt Enable Register				
0	gie	R/W	0	When asserted, along with the IP Interrupt Enable bit, the interrupt is enabled.
31:1	reserved			

Continued on next page

Bit	Name	Access Type	Reset Value	Description

Tabelle 3.4: Global Interrupt Enable Register (GIER)

Bit	Name	Access Type	Reset Value	Description
0x08 IPIER - IP Interrupt Enable Register				
0	ipie	R/W	0	When asserted, along with Global Interrupt Enable bit, the interrupt is enabled. (default: uses the internal ap_done signal to trigger an interrupt)
31:1	reserved			

Tabelle 3.5: IP Interrupt Enable Register (IPIER)

Bit	Name	Access Type	Reset Value	Description
0x0C IPISR - IP Interrupt Status Register				
0	ipis	R/W	0	Toggle on write. (write 1 to clear(W1C))
31:1	reserved			

Tabelle 3.6: IP Interrupt Status Register (IPISR)

Bit	Name	Access Type	Reset Value	Description
0x10 IDR - ID Register				
31:0	ID	R	0x534F4E52	Distinct ID for PmodMAXSONAR (ASCII for SONR)

Tabelle 3.7: ID Register (IDR)

Bit	Name	Access Type	Reset Value	Description
0x14 VERR - Version Register				
31:0	VER	R	0x80001000	Version

Tabelle 3.8: Version Register (VERR)

Bit	Name	Access Type	Reset Value	Description
0x18 SCSR0 - Special Control and Status Register				

Continued on next page

Bit	Name	Access Type	Reset Value	Description
0	powerup_done	R	0	Asserted when the powerup time of the sonar sensor has passed
1	config_done	R	0	Asserted when the configuration of the sonar sensor is done
2	read_valid	R	0	Signals if the current values in the DIST0 Register are valid. In single reading mode the flag is asserted when completed a sensor read without an error. When a new reading starts the flag is cleared. In free running mode the flag is set when the first error free sensor reading was conducted. If there are errors while reading sensor, the flag should be cleared.
5:3	reserved			
6	reset_ip	R	0	Stops the whole IP and resets all values
7	freeze_ip	R	0	Stops the whole IP but does not reset the values (useful for debugging)
31:8	reserved			

Tabelle 3.9: Special Control and Status Register (SCSR0)

Bit	Name	Access Type	Reset Value	Description
0x1C DIST0 - Distance Value Register				
7:0	dist_in	R	0x00	Distance in inches • 0x00 = no valid reading yet • 0x06 - 0xFF • 6in - 255in
15:8	dist_char_1 *(optional)*	R	0x00	First Digit of Distance value ($x_1 * 10^2$)
23:16	dist_char_2 *(optional)*	R	0x00	Second Digit of Distance value ($x_2 * 10^1$)
31:24	dist_char_3 *(optional)*	R	0x00	Third Digit of Distance value ($x_3 * 10^0$)

Tabelle 3.10: Distance Value Register (DIST0)

Bit	Name	Access Type	Reset Value	Description
0x20 UCSR - UART Control and Status Register				
0	ur_error	R	0	Set to one, if the UART Receiver moves into the error state. Cleared on reset.
1	ur_bd_rate *(optional)*	R/W	0	Baudrate of the UART Receiver 0 = 4800 Baud 1 = 9600 Baud
2	ur_os_rate *(optional)*	R/W	0	Oversampling Rate of UART Receiver 0 = 8x Oversampling 1 = 16x Oversampling
6:3	ur_data_bits *(optional)*	R/W	0x1	Number of Databits 0x1 = 5 Bits 0x2 = 6 Bits 0x4 = 7 Bits 0x8 = 8 Bits
7	reserved			
15:8	ur_data	R	0x00	Current status of the UART Receive Buffer (for Debug Purposes)
31:16	reserved			

Tabelle 3.11: UART Receiver Status Register (URSR)

Bit	Name	Access Type	Reset Value	Description
0x24 ADSR - ASCII Decoder Status Register				
0	ad_error *(optional)*	R	0	Set, if the ASCII Decoder moves into error state (the packet structure „Rxxx\r“ violated). Cleared on reset.
5:1	ad_err_pos *(optional)*	R	0x00	One-hot bitmask of the ASCII character which caused the error 0x01 = R 0x02 = first number 0x04 = second number 0x08 = third number 0x10 = cariage return
7:6	reserved			
15:8	ad_err_char *(optional)*	R	0x00	ASCII character which caused the error
31:16	reserved			

Tabelle 3.12: ASCII Decoder Status Register (ADSR)