Entwicklung und Optimierung von Display-Schnittstellen fuer embedded Linux Boards

-

Masterarbeit

im Fachgebiet Hard- und Softwareentwicklung

vorgelegt von:	Armin Schlegel
Studienbereich:	Fakultaet EFI
Matrikelnummer:	2020863
Erstgutachter:	Prof. Dr. Joerg Arndt
Zweitgutachter:	Peter Meier

© 2014

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

© Armin Schlegel

Zusammenfassung

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Quisque Natural-Programmierung blandit sed, hendrerit at, pharetra eget, dui. Sed lacus. Pellentesque malesuada. Cras gravida mi id sapien. Ut risus justo, fermentum non, scelerisque sit amet, lacinia in, erat. Proin nec lorem. Quisque porta, nisl at porta aliquam, felis libero consequat ipsum, vitae scelerisque dolor mi a odio. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Duis sollicitudin. Proin sollicitudin varius arcu. Morbi eleifend, metus sit amet placerat pharetra, dolor dui lobortis pede, vel imperdiet tellus eros imperdiet lorem. In hac habitasse platea dictumst. Curabitur elit mi, facilisis nec, ultricies id, aliquet et, magna. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam ac est. Mauris turpis enim, feugiat non, imperdiet congue, scelerisque non, purus. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Nullam dictum aliquet purus. Maecenas faucibus. Maecenas suscipit.

Abstract

Fusce neque est, tincidunt eu, nonummy nec, tempor iaculis, erat. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Vestibulum egestas, velit a rhoncus gravida, metus dolor pulvinar diam, sit amet placerat risus dolor sit amet elit. Maecenas eget purus ut est mattis porta. Suspendisse ut mi et mauris lobortis malesuada. Vestibulum dapibus. Duis hendrerit, elit eu venenatis eleifend, sapien ante volutpat odio, ac condimentum tellus massa ut massa. Etiam dapibus imperdiet metus. Sed sapien arcu, pulvinar quis, laoreet quis, venenatis non, justo. Aliquam est ante, pulvinar nec, accumsan sed, auctor sed, augue.

Ut adipiscing ligula. In mattis. Ut varius. In nec nulla at eros molestie viverra. Duis dolor risus, lobortis vel, dictum a, pellentesque id, lectus. Sed suscipit orci ac ligula venenatis condimentum. Maecenas et sem lacinia tortor cursus tempus. Mauris pellentesque risus at nulla. In arcu. Curabitur mattis mi quis dolor. In leo. Vivamus ut libero.

© Armin Schlegel

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Listings

1. Einleitung

1.1. Motivation

In der heutigen Zeit treten eingebettete Systeme (engl. embedded systems) immer stärker in den Vordergrund. Gerade in den Bereichen der Industrie, Telekommunikation oder Multimedia wächst der Bedarf an Lösungen die durch Zuverlässigkeit, Energiesparsamkeit und kompakter Bauform bestechen.
Obwohl eingebettete Systeme meist für den Anwender unsichtbar ihren Dienst verrichten, sind sie doch inzwischen allgegenwärtig. Im Bereich der Telekommunikation und Unterhaltungselektronik kommt ein solches System im Prinzip nicht mehr ohne ein Display aus. Die Möglichkeit zur Anzeige multimedialer Daten wird zur Kaufentscheidung. Auch hier gilt die Maxime: besser, schneller, größer.
Im Sektor der eingebetten Systeme spielen Betriebssystem wie Linux neben diversen anderen Systemen wie beispielsweise RTOS, OSEK, QNX oder auch Windows eine sehr große Rolle. In Verbindung zeigen eingebettete Linuxsysteme mit Displays ein großes Potential. Mit der beliebten ARM-Architektur lassen sich so kostengünstige, leistungsstarke Systeme aufbauen, die die gestellten Aufgaben gut erfüllen kann. Sieht man sich allein den Marktanteil von Smartphones welche auf Android-Basis arbeiten an, wird der Trend klar, dass Hersteller eine offene Basis bevorzugen. Brandt [2013]
Es ist ersichtlich, dass auch in Zukunft Linux auf eingebetteten Systemen eine immer größere Rollen spielen wird.

1.2. Ziel der Arbeit

Das Ziel dieser Arbeit ist zu zeigen, dass die Verwendung von Displays mit eingebetten Linux Systemen je nach Anforderung einfach oder über Umwege realisierbar ist.

1.3. Aufbau der Arbeit

Im ersten Teil der Arbeit werden theoretische Grundlagen gebildet, die für das Verständnis nötig sind. Hier werden Standards wie z.B. HDMI bzw. DVI, LVDS und RGB behandelt. Es wird ein Überblick über ausgewählte embedded Linux Boards gegeben und diese Klassifiziert mit welchen Displayschnittstellen diese ausgestattet sind bzw.. ausgestattet werden können. Der Zweite Teil behandelt das embedded Linux Board 'Gnublin', welches von Haus aus keine Displayschnittstelle vorgesehen hat. Hier werden zwei Varianten zur Ansteuerung von Displays erarbeitet. Die Ansteuerung wird hierbei vom Prozessor erledigt, da das 'Gnublin' keine dedizierten Grafikcontroller besitzt. Im dritten Teil wird für leistungsstärkere embedded Linux-Systeme mit HDMI-Schnittstelle eine Hardware entwickelt, RGB- oder LVDS-Panels anzuschließen. Um die Displays über die entwickelte Hardware anzusteuern, wird der dedizierte Grafikcontroller der Boards verwendet.

1.4. Typographische Konventionen

1.5. Verwendete Programme

Um Schaltpläne und Layouts zu erstellen, wurde das Programm Eagle von Cadsoft[1] verwendet. Im Rahmen von Teil B dieser Arbeit ist eine Bauteilbibliothek entstanden, um alle benötigten Bauteile im Schaltplan und Layout verwenden zu können. Diese Bibliothek befindet sich im Anhang auf der CD. Um 3D Bilder von Platinenlayouts zu erzeugen, wurde das Eagle Plugin Eagle3D[2] Für elektrische Simulationen wurde das Programm LTSpice von Linear Technology[3] verwendet. Die für den Teil B durchgeführte Simulation befindet sich im Anhang auf der CD. Zur Entwicklung der Programme für die Plattformen PC, ARM und AVR wurde Eclipse[4] verwendet. Die verwendeten Compiler sind allesamt Plattformabhaengige gcc-Versionen[5]. Tabelle 1.1 zeigt eine Übersicht der verwendeten Compiler für diese Arbeit.

Plattform	Compiler	Version
Linux 3.10.11-smp i686	gcc	4.8.1
Atmel ATMega88p	avr-gcc	4.3.3
ARM9 NXP LPC313x	arm-linux-gnueabi-gcc	4.6.4

Tabelle 1.1.: Verwendete Compiler

[1] http://www.cadsoft.com/
[2] http://sourceforge.net/projects/eagle3d.berlios/
[3] http://www.linear.com/designtools/software/
[4] https://www.eclipse.org/
[5] https://gcc.gnu.org/

2. Theoretische Grundlagen

In diesem Kapitel werden Theoretische Grundlagen geschaffen, die zum weiteren Verständnis der Arbeit benötigt werden. Zuerst werden ausgewählte Video-Schnittstellen erläutert und verglichen und bewertet welchen praktischen Nutzen diese für handelsübliche embedded Linuxsysteme bietet. Im Weiteren werden zwei Linux Boards verglichen und bewertet sowie deren praktische Einsatzgebiete beispielhaft dargelegt.

2.1. Video-Schnittstellen

Unter Video-Schnittstellen kann man die Schnittstellen verstehen, die direkt zur Anzeige von Bilddaten dienen und physikalisch mit einer Anzeigeeinheit verbunden sind. Hier können sowohl Hardware- als auch Softwarekomponenten enthalten sein.

2.1.1. VGA

Unter VGA versteht man Video Graphics Array und wurde 1987 von IBM entwickelt. Der Stecker hat 15 Pins und liefert neben analogen Farbinformationen Horizontale und Vertikale Synchronisationssignale. Aufgrund der limitierten Spezifikationen ist die Schnittstelle eher antik und selbst Intel als Chiphersteller will ab 2015 auf die Schnittstelle verzichten und digitalen Schnittstellen den Vorzug lassen (Knuppfer [2010]). Zwar ist die VGA-Schnittstelle noch nicht komplett obsolet, so wird sie den digitalen Schnittstellen trotzdem weichen müssen. Der Trend bei embedded Linuxsystemen ist zumindest der, dass handelsübliche Systeme direkt mit HDMI oder anderen digitalen Schnittstellen entwickelt werden. Die Funktionsweise der VGA-Schnittstelle ist in Abbildung 2.1 zu sehen. Es werden fünf analoge Leitungen benötigt: R, G, B, HSYNC[6] und VSYNC[7]. Die ersten drei stellen die Farbwerte Rot, Grün und Blau dar. Je nach Intensität der Farbkanäle lassen sich aus einer Mischung jede Farbe darstellen. Zur Steuerung der Intensität können Pegel zwischen 0V (absolut dunkel) und +0.7V (absolut hell) pro Farbkanal angenommen werden. Die Signale HSYNC und VSYNC werden zur Steuerung der Zeilen und Spalten verwendet.

[6]HSYNC: Horizontale Synchronisation
[7]VSYNC: Vertikale Synchronisation

Das Signal HSYNC zeigt an, wann eine Zeile vollständig ist. Während der HSYNC-Periode werden für jeden Pixel der Zeile zeitlich exakte Pulse auf den Farbleitungen angelegt. Sind alle Zeilen eines Bildes komplett, wird das VSYNC-Signal angestoßen, welches ein neues Bild von vorne beginnt (Valcarce [2011]).

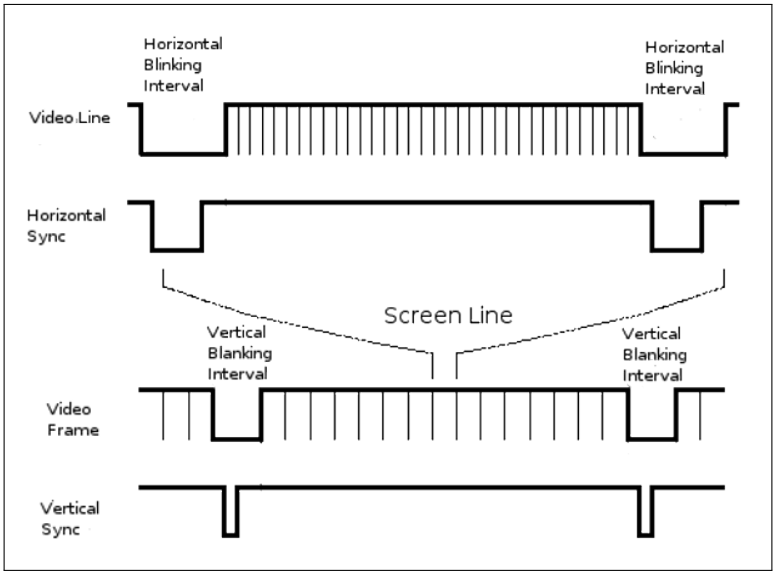


Abbildung 2.1.: VGA-Timing, Quelle: Valcarce [2011]

2.1.2. DVI

Hinter DVI steht der Begriff Digital Visual Interface und stellt ein digitale Schnittstelle zur Grafikanzeige dar. Der DVI Standard wurde 1999 von der DDWG[8] verabschiedet, da der Wunsch nach Leistungsstärkeren Schnittstellen vorhanden war. QXGA-Auflösungen[9] sind auf analogem Wege nicht mehr befriedigend erzielbar. Die DVI Schnittstelle beinhaltet neben den Digitalen Signalen zusätzlich analoge VGA Signale, was den Betrieb älterer Monitore und Displays zulässt. Zur digitalen Datenübertragung wird der TMDS[10] Standard verwendet, welcher die 24 Bit Farbinformationen[11] mittels eines Serializers in serielle Daten umwandelt. Je nach benötigter Bandbreite können drei oder sechs Aderpaare für Pixeldaten verwendet werden. Dies wird Single-Link bzw. Double-Link genannt und es lassen sich dabei max. 3.72 GBit/s[12] bzw. 7.44 GBit/s[13] übertragen. Um die Paare zuordnen zu können, wird ein weiteres Paar zur Synchronisation verwendet. Um die Übertragung

[8] Digital Display Working Group
[9] QXGA: 2048x1536
[10] Transition Minimized Differential Signaling - Differentielle Datenübertragung
[11] 24 Bit: je 8 Bit für Rot, Grün und Blau
[12] max. UXGA: 1600x1200@60Hz
[13] max. WUXGA: 1920x1200@60Hz

noch effizienter zu gestalten, gibt es die Möglichkeit bei High- sowie Low-Pegel des Taktsignals Daten zu übertragen[14] (Leunig [2002]).

2.1.3. HDMI

Gegeneber der DVI-Schnittstelle bietet die HDMI Schnittstelle dieselben Eigenschaften bezüglich der Videoübertragung verwendet zur ebenfalls TMDS. Hinzu kommt allerdings, dass sowohl Audio als auch Verschlüsselung unterstützt werden. Der Formfaktor der Stecker sind für den Hausgebrauch verkleinert worden. HDMI wurde als normierte Universallösung entwickelt und hat sich als solche etabliert (Extron [2014]). Nahezu jedes neu entwickelte Gerät mit Anzeigemöglichkeit, bietet eine HDMI-Schnittstelle - ebenso embedded Linux Boards wie z.B. bekannte Linux Boards wie Raspberry Pi oder Beagle Bone Black.

2.1.4. RGB

Der RGB-Bus, verwendet für kleine TFT-Panels bis ca. 7“, funktioniert prinzipiell analog zur VGA-Schnittstelle, mit dem Unterschied, dass die Datenleitungen komplett digital sind. So werden die Signale für Rot, Grün und Blau nicht mehr analog im Bereich von 0V bis +0.7V dargestellt, sondern durch einen üblicherweise acht Bit breiten Bus pro Farbkanal. Die Auflösung pro Farbkanal ist mit 255 Intensitätsstufen gerechnet ausreichend um ein gesamtes Farbspektrum von 16.777.216 [15] Farben zu erhalten. Dieser Farbmodus wird auch RGB888 genannt, da acht Bit für jede Farbe zur Kodierung, insgesamt also 24 Bit, zur Verfügung stehen. Neben dem 24 Bit Modus ist RGB565 noch weit verbreitet, der je fünf Bit für Rot und Blau und sechs Bit für Grün verwendet. Hier ergibt sich ein Farbspektrum von 65.536[16]Farben. Da digital übertragen wird, ist eine Taktleitung notwendig um die Synchronizität zu ermöglichen. Aufgrund der Verbreitung und Mächtigkeit der Schnittstelle besitzen

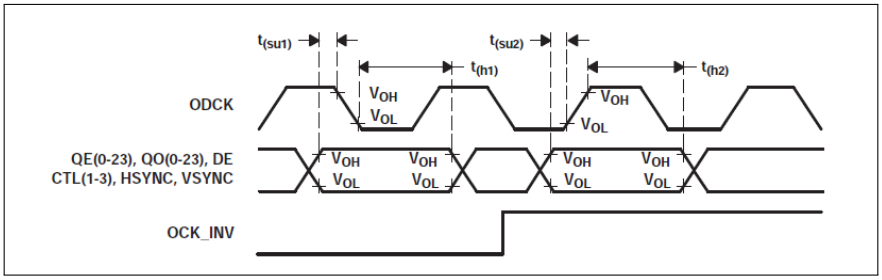


Abbildung 2.2.: RGB-Timing, Quelle: Texas Instruments [2011]

[14] *Double Data Rate*
[15] 16.777.216 Farben = 2^{24}
[16] 65.536 = 2^{16}

einige Prozessor, wie z.B. der Linuxfähige OMAP3530 von Texas Instruments, eine RGB-Schnittstelle. Abbildung 2.2 zeigt exemplarisch ein Timing-Diagramm der RGB-Schnittstelle des Bausteins TFP-401A von Texas Instruments.

2.1.5. LVDS

Um lange Strecken und große Bildformate übertragen zu können ist der parallele Datentransfer ungeeignet, da bei schnellem Takt z.B. das Übersprechen zu groß wird und das Signal schneller gestört wird. Deshalb ist die Praktik beliebt, große Datenmengen über eine differentielle Verbindung wie z.B. LVDS[17] zu übertragen. Die physikalische Funktionsweise der LVDS Leitung liegt darin, dass zweimal dasselbe Signal übertragen wird - mit positiver Spannung und mit negativer Spannung. Wirkt nun von außen eine Störung auf die LVDS Leitung, werden beide Leitungen - positive wie auch die negative - gleichermaßen gestört. Durch das Zusammenführen beider Signale am Ende, kompensieren sich diese Störungen im Idealfall zu Null. Wie auch LVDS arbeitet das zuvor genannte TMDS ähnlich, da es sich hierbei auch um eine differentielle Übertragungsart handelt. Der Unterschied liegt in der Verwendung. TMDS wird oft eingesetzt, sobald das Signal das Gerät verlässt - z.B. Desktop-Bildschirm mit Anschlusskabel. Befindet sich das Anzeigegerät allerdings im selben Gehäuse, so wird oft LVDS eingesetzt. Neben Bilddaten ist es natürlich auch möglich andere Nutzdaten wie z.B. Sensordaten zu übertragen. Aufgrund der hohen Geschwindigkeit und geringen Fehlerrate werden differentiellen Übertragungen werden gerne für Displays angewendet.

2.1.6. 8080-Interface

Das 8080-Interface ist eine antike Schnittstelle ursprünglich von Intel 8080 Prozessor. Sie wird bis heute verwendet, um Speicher, kleine TFT-Displays oder andere Bausteine mit einem Mikrocontroller zu betreiben. Eckdaten des 8080-Interface sind sowohl der Datenbus selbst als auch der Adressbus mit z.B. acht, 16 oder 32 Bit, je eine eine Leitung für Read-Enable, Write-Enable und Chip-Select. Durch die Verwendung der Chip-Select Leitungen ist es möglich mehrere Teilnehmer am selben Bus zu betreiben. Alle Teilnehmer, deren Chip-Leitung nicht aktiv ist, verhalten sich für andere Busteilnehmer unsichtbar. Erst mit Zuweisung der Chip-Selects werden diese sichtbar und übernehmen den Bus. Ein Hostsystem steuert als sog. Master die am Bus hängenden Slaves. Moechte das Hostsystem von einem Slave Daten lesen, wird ein Lesezyklus initiiert, der die Chip-Select Leitung aktiviert, die gewünschte

[17] LVDS: Low Voltage Differential Signaling

Adresse an den Bus anlegt, die Read-Enable Leitung aktiviert und nach einer festgelegten Zeit diese wieder deaktiviert. Der Slave legt die gewünschten Daten auf den Datenbus und der Host kann diese Daten korrekt lesen. Analog dazu funktioniert der Schreibzyklus ähnlich. Abbildung 2.3 zeigt das Timing Diagramm eines Schreib- und Lesezyklus des Displaycontrollers SSD1289. Das Signal D/C wird verwendet um zu unterscheiden, ob ein Daten oder ein Kommando auf dem Bus anliegen. Dazu kann beispielsweise eine Adressleitung des 8080-Bus verwendet werden. CS stellt das Chip-Select dar. WR und RD beziehen sich auf Write- bzw. Read-Enable. D0-D17 sind 18 Datenbits des Bus (Solomon Systech Limited [2007]).

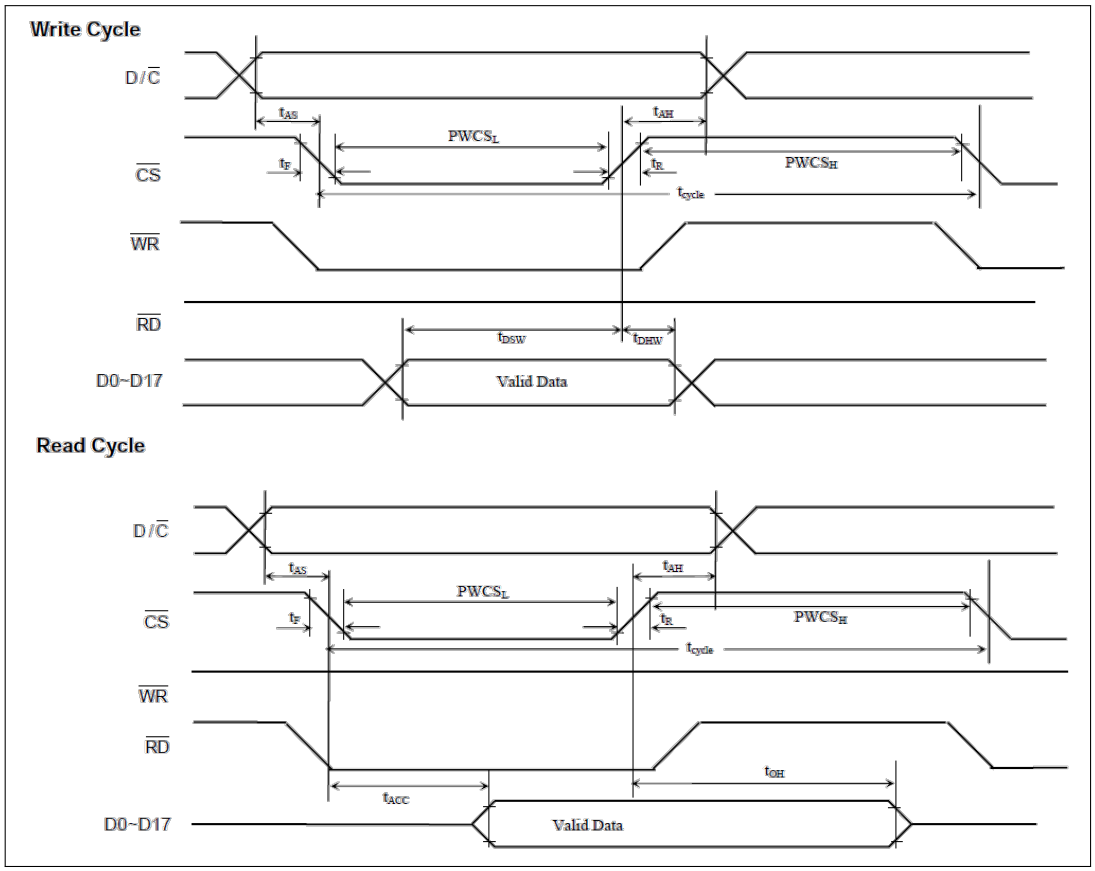


Abbildung 2.3.: 8080-Timing des SSD1289, Quelle: Solomon Systech Limited [2007]

Viele Mikrocontroller besitzen bereits ein 8080-Interface dediziert in Hardware - allerdings nicht alle. Als Ersatz kann das Protokoll mit GPIO[18] in Software implementiert werden. Dies ist allerdings wesentlich langsamer als eine Lösung, die bereits in Hardware läuft, da GPIO-Pins nicht dafür geschaffen sind, sich mit schneller Frequenz schalten zu lassen.

[18]GPIO: General Purpose In/Output

2.1.7. Bewertung der Video-Schnittstellen

Nachdem nun die wichtigsten Schnittstellen dargestellt wurden, werden diese im Folgenden mit dem Fokus auf die Masterareit hinsichtlich der Relevanz bewertet. Da die VGA-Schnittstelle antik und obsolet ist, spielt sie heutzutage nur noch eine geringe Rolle. Insbesondere im Bereich der embedded Systeme wird sie kaum verwendet. Für die Masterarbeit ist die VGA-Schnittstelle uninteressant, da diese von keiner, in der Masterarbeit behandelten, Hardware verwendet wird. Jedoch wurde diese eingangs behandelt, da diese den Übergang zum digitalen RGB-Bus schafft.
Die DVI- und HDMI-Schnittstellen, welche für den Bereich der Videoanzeige praktisch identisch sind, nehmen jedoch einen hohen Stellenwert in der Masterarbeit ein. Im zweiten Teil der Arbeit wird eine Hardware entwickelt, welche als Eingangssignale die TMDS der DVI-/HDMI-Schnittstelle nutzt. Ebenso spielen die RGB-Schnittstelle und LVDS eine große Rolle, da an diesen Schnittstellen der entwickelten Hardware TFT-Panels angeschlossen werden.

große Rolle: umschreiben

Neben den reinen Video-Schnittstellen weist das beschriebene 8080-Interface, das ursprünglich nicht zur Bildübertragung gedacht war, ein hohes Potential auf und besitzt für den ersten Teil der Masterarbeit hohen Stellenwert. Gerade im embedded Bereich besitzt diese Schnittstelle nach wie vor einen hohen Stellenwert, da vor allem kleine Displays damit hinreichend schnell und effizient betrieben werden können. Tabelle 2.1 zeigt nochmals eine kurze Übersicht der Relevanz der einzelnen Schnittstellen für die Masterarbeit.

Schnittstelle	Relevanz für Masterarbeit	Verwendung in der Masterarbeit
VGA	keine	-
DVI	mittel	Teil B
HDMI	hoch	Teil B
RGB	hoch	Teil B
LVDS	hoch	Teil B
8080-Interface	hoch	Teil A

Tabelle 2.1.: Relevanz der Display-Schnittstellen für die Masterarbeit

2.2. Betrachtete Embedded Linux Boards

In diesem Abschnitt werden die verwendeten Linux-Boards dargestellt, verglichen und hinsichtlich der Verwendbarkeit in der Masterarbeit bewertet. Da sich die Masterarbeit in zwei Teile gliedert, wird für beide Anwendungsfaelle ein typisches Linux-Board hergezogen, welches den Anforderungen gerecht werden muss, eine billige und effiziente Anzeige zu gestatten.

2.2.1. Raspberry Pi

Am wohl bekanntesten und mit einer riesigen Community hinter dem Projekt ist der Raspberry Pi von der Raspberry Pi Foundation [19]. Um die wichtigsten Eckdaten des Einplatinenrechners im Checkkartenformat zu nennen, besitzt er in der Ausfuehrung Model B einen ARM11-Core (Broadcom BCM2835), 512 Megabyte SDRAM, eine Broadcom VideoCore IV GPU sowie diverse Schnittstellen wie HDMI, USB 2.0, UART[20], SPI[21], I2C[22] sowie GPIO-Pins[23].
Der erschwingliche Preis macht den Raspberry Pi attraktiv und zieht die Community an, da man für rund 40 Euro einen kompletten Rechner bekommt.

Lehrstuhl	Professor
BWL	Maier
MB	Müller
Jura	Schmidt

2.2.2. Gnublin Extended

2.2.3. Bewertung der Linux-Boards

8080-Interface in SRAM erwaehnen!

[19] http://www.raspberrypi.org
[20] UART: Universal Asynchronous Receiver Transmitter - RS2332
[21] SPI: Serial Peripheral Interface - 4 Draht Bus
[22] I2C: Inter Integrated Circuit - 2 Draht Bus
[23] GPIO: General Purpose Input Output

3. Teil A

Im Folgenden Kapitel wird Teil A dieser Arbeit behandelt. Es wird die Ansteuerung von TFT Displays über den 8080-Bus auf Basis des Gnublin Linuxboards realisiert. Hierzu wurden verschieden große LCD Displays mit unterschiedlichen Controllern unter Verwendung des 8080-Interface und untersucht.

3.1. Untersuchte Displays mit 8080-Interface

Dieser Abschnitt behandelt die untersuchten Displays. Es wurden drei Displays aus China untersucht. Der Fokus bei der Bestellung lag vor allem darauf, dass die Pinbelegung der jeweiligen Displays übereinstimmen. So ist die Entwicklung von nur einer Adapterplatine zwischen Gnublin und Display nötig. Alle verwendeten Displays werden im 16 Bit Farbmodus betrieben. Die hieraus resultierende Farbtiefe beträgt 65.535 Farben.
Alle verwendeten Displays arbeiten dahingehend gleich, dass sie Kommandos und Daten auf dem Datenbus anliegen, diese jedoch durch eine gesonderte Leitung unterschieden werden. Soll dem Display also etwas mitgeteilt werden, so muss zuerst ein entsprechendes Kommando und im Anschluss die Nutzdaten gesendet werden. Um Pixeldaten an das Display zu senden, hat sich die Vorgehensweise etabliert, eine Rechteckige Region im RAM des Displays zu reservieren, das durch die 4 Eckpunkte des Rechtecks definiert sind. Werden im Anschluss Pixeldaten gesendet, inkrementiert der Controller die Adresse automatisch und springt bei einem Zeilenumbruch automatisch an die richtige Stelle im RAM. Der maximale Speicher im Controller beschränkt die maximale Auflösung der ansteuerbaren TFT-Panel. Trotz der Tatsache, dass sich die Displays auf elektrischer Seite nicht unterscheiden, so müssen diese allerdings alle speziell softwareseitig behandelt werden.

bild von Ramfenster hinzufuegen

3.1.1. 4.3“/5“ mit SSD1963

Die Wahl des Controllers SSD1963 von Solomon Systech liegt nahe, da dieser bereits mit einem 4.3“ Panel in einer vorausgehende Arbeit verwendet wird. Dort ist das Display mittels GPIO-Pins am Raspberry Pi angeschlossen. Die Software bezüglich der reinen Displayansteuerung ist somit bereits vorhanden (siehe **?**). Aufgrund eines

Problems, das in Abschnitt 3.3.1 näher beschrieben ist, wird für diese Arbeit zusätzlich ein anderes Display mit 5“ Panel aber selbem Controller untersucht. Abbildung 3.1 zeigt das Pinout der verwendeten Displays. Die Displays besitzen bei 4.3“ eine

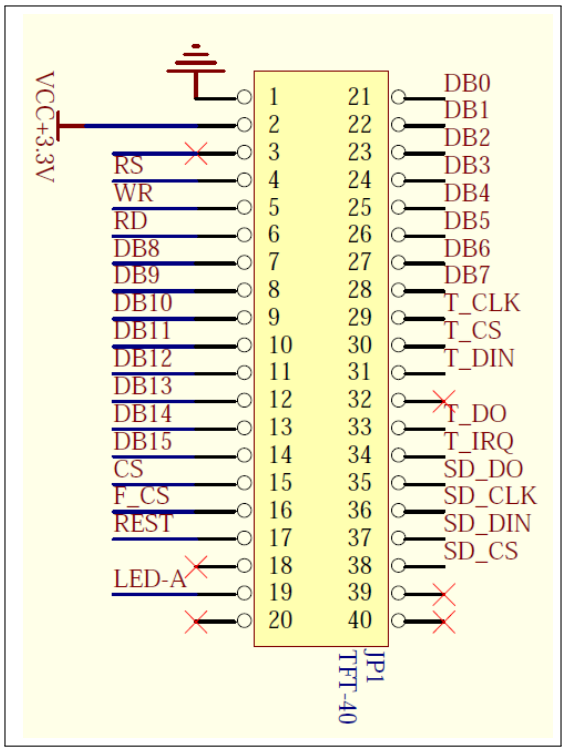


Abbildung 3.1.: 8080-Display Pinout, Quelle: Coldtears Electronics

Auflösung von 480x272 beziehungsweise bei 5“ 800x480 Pixeln. Neben den für die Initialisierung nötigen Kommandos besitzt der Controller folgende wichtigen Kommandos. Diese sind in Tabelle 3.1 beschrieben. Die zur Initialisierung notwendigen Kommandos werden hier nicht erläutert, da diese aus dem Datenblatt entnehmbar sind.

Kommando	**Hex-Code**	**Kommentar**
Set Column Address	0x2A	Eckpunkte des RAM-Fensters in X-Richtung
Set Page Address	0x2B	Eckpunkte des RAM-Fensters in Y-Richtung
Write Memory Start	0x2C	Alle Folgenden Pixeldaten werden im RAM-Fenster platziert

Tabelle 3.1.: Relevante Kommandos des SSD1963, Solomon Systech Limited [2008]

3.1.2. 3.2“ mit SSD1289

Das 3.2“ Display von Sainsmart wird mit einen SSD1289 von Solomon Systech betrieben. Dieses Display hat eine Aufloesung ovn 320x240 Farbpunkten. Das Pinout ist dasselbe, das in Abbildung 3.1MPMCStatic zu sehen ist. Analog zu Tabelle 3.1 besitzt der SSD1289 seine eigenen wichtigen Kommandos. Diese sind in Tabelle 3.2 erlaeutert. Die zur Initialisierung notwendigen Kommandos werden hier nicht erläutert, da diese aus dem Datenblatt entnehmbar sind.

Kommando	Hex-Code	Kommentar
Horizontal RAM address position	0x44	Eckpunkte des RAM-Fensters in X-Richtung
Vertical RAM address start position	0x45	Startpunkt des RAM-Fensters in Y-Richtung
Horizontal RAM address stop position	0x46	Endpunkt des RAM-Fensters in Y-Richtung
Set GDDRAM X address counter	0x4E	Zeiger im RAM-Fenster in X-Richtung
Set GDDRAM Y address counter	0x4F	Zeiger im RAM-Fenster in Y-Richtung
RAM Write Register	0x22	Alle Folgenden Pixeldaten werden im RAM-Fenster platziert

Tabelle 3.2.: Relevante Kommandos des SSD1289, Solomon Systech Limited [2007]

3.1.3. 5“ mit CPLD

Als drittes Display mit 8080-Interface kommt eine 5“ Display mit einer Auflösung von 800x480 Bildpunkten zum Einsatz, dass keinen univerell einsetzbaren Controller für variable Displaypanels im klassischen Sinn besitzt, sondern ein CPLD [24] als Controller mit zugeschnittenen Timings für das verwendete TFT-Panel. Der Vorteil eines solchen Displays ist, dass keine Initialisierungsroutine benötigt wird, um die Timings für das Panel einzustellen. Ein Reset setzt das Display betriebsbereit. Nachteilig stellt sich der Umstand ein, dass nur TFT-Panels exakter Größe und mit exakten Timings verwendet werden können. Für diese Arbeit ist allerdings die Verwendung von anderen Panels belanglos. Auch hier ist das Pinout des Displays analog zu dem Gezeigten in Abbildung 3.1.
Wichtige Kommandos zum Betrieb des Displays sind in Tabelle 3.3 einsehbar. Dieses Display trägt die Bezeichnung MD050SD.

[24] CPLD: Complex Programmable Logic Device

Kommando	Hex-Code	Kommentar
Beginning Row Address	0x02	Startpunkt des RAM-Fensters in X-Richtung
Ending Row Address	0x06	Endpunkt des RAM-Fensters in X-Richtung
Beginning Column Address	0x03	Startpunkt des RAM-Fensters in Y-Richtung
Ending Column Address	0x07	Endpunkt des RAM-Fensters in Y-Richtung
Writing Page Register	0x05	Alle Folgenden Pixeldaten werden im RAM-Fenster platziert

Tabelle 3.3.: Relevante Kommandos des MD050SD, ITEAD Studios [2013]

3.2. 8080-Interface mittels SRAM-Interface

Wie bereits in Abschnitt 2.2.2 erwaehnt, besitzt der Prozessor des Gnublin bereits ein externes 8080-Interface, auf welches zugegriffen wird. Im Folgenden wird auf das Konzept, die Idee und die Realisierung auf Hardware- und Softwareseite eingegangen.

3.2.1. Konzept

Im Gnublin stellt ein NXP LPC313x die zentrale Recheneinheit dar. Dieser besitzt ein sogenanntes EBI [25], worüber externe Bausteine wie Speicher, Ethernetcontroller oder ähnliche Bausteine angesprochen werden können.

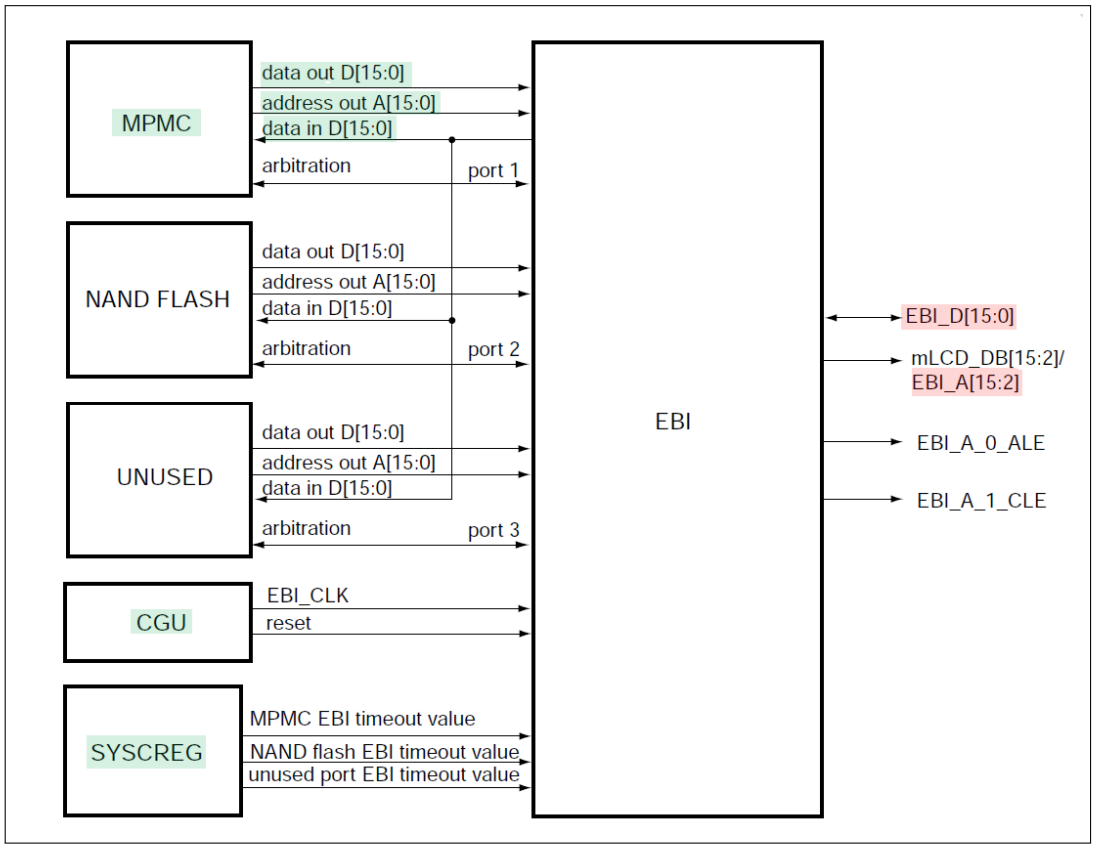


Abbildung 3.2.: NXP LPC313x EBI, Quelle: NXP Semiconductors [2010]

In Abbildung 3.2 ist ein Blockschaltbild des EBI zu sehen, bei welchem neben CGU [26] und SYSCREG[27], MPMC[28] sowie das NAND Flash an den Eingängen des EBI angeschlossen sind. Abgesehen von NAND Flash sind die Eingänge zum EBI für diese Arbeit relevant und grün markiert. An den Ausgängen des EBI sind Adress- und Datenbus zum Anschluss an externe Bausteine herausgeführt. Damit verschiedenartigen Bausteine an denselben Adress- und Datenpins angeschlossen werden kann, ist eine Priorisierung notwendig. Die Höchste Priorität besitzt der MPMC, gefolgt vom NAND Flash. Die Grundidee ist, das Display über den MPMC anzuschließen, da er so konfiguriert werden kann, dass er sich 8080-konform verhält. Die für diese Arbeit

[25] EBI: External Bus Interface
[26] CGU: Clock Generation Unit, Takterzeugung
[27] SYSCREG: System Control Register, Steuerregister
[28] MPMC: Multiport Memory Controller

interessanten Leitungen am Ausgang des EBI sind mit rot markiert. Hier wird der Datenbus selbst, sowie die oberen 13 Bit des Adressbus gezeigt.

3.2.2. MPMC - Multiport Memory Controller des NXP LPC313x

Der MPMC stellt die Möglichkeit zur Verfügung Bausteine wie dynamisches und statisches RAM anzubinden. Die Refresh-Zyklen werden bei Verwendung von dynamischen RAMs automatisch vollzogen. Das SDRAM-Interface bietet von Haus aus die Möglichkeit Displays mit 8080-Interface zu betreiben. Dies schließt allerdings die Verwendung von dynamischen RAMs aus. Soll ein Betriebssystem wie Linux auf dem System betrieben werden, ist allerdings die Verwendung von dynamischem RAM unerlässlich. Im Folgenden wird die Schnittstelle für das statische RAM SRAM-Interface benannt. Es besteht die Möglichkeit das Interface des statischen RAM zu verwenden, um ein Display zu betreiben, da es sich so konfigurieren lässt, dass es sich wie ein 8080-Interface verhält. Damit sich die verschiedenen Slaves an Adress- und Datenbus nicht überschneiden, regelt das EBI den Zugriff auf die Busse über Chip-Select Leitungen. Am Gnublin ist eine dieser Chip-Select-Leitungen für das SRAM-Interface nach außen gelegt. Die restlichen Anschlüsse wie Write-Enable, Read-Enable, Reset sind ebenfalls herausgeführt NXP Semiconductors [2010]. Ein Blockschaltbild des MPMC ist in Abbildung 3.3 zu sehen.

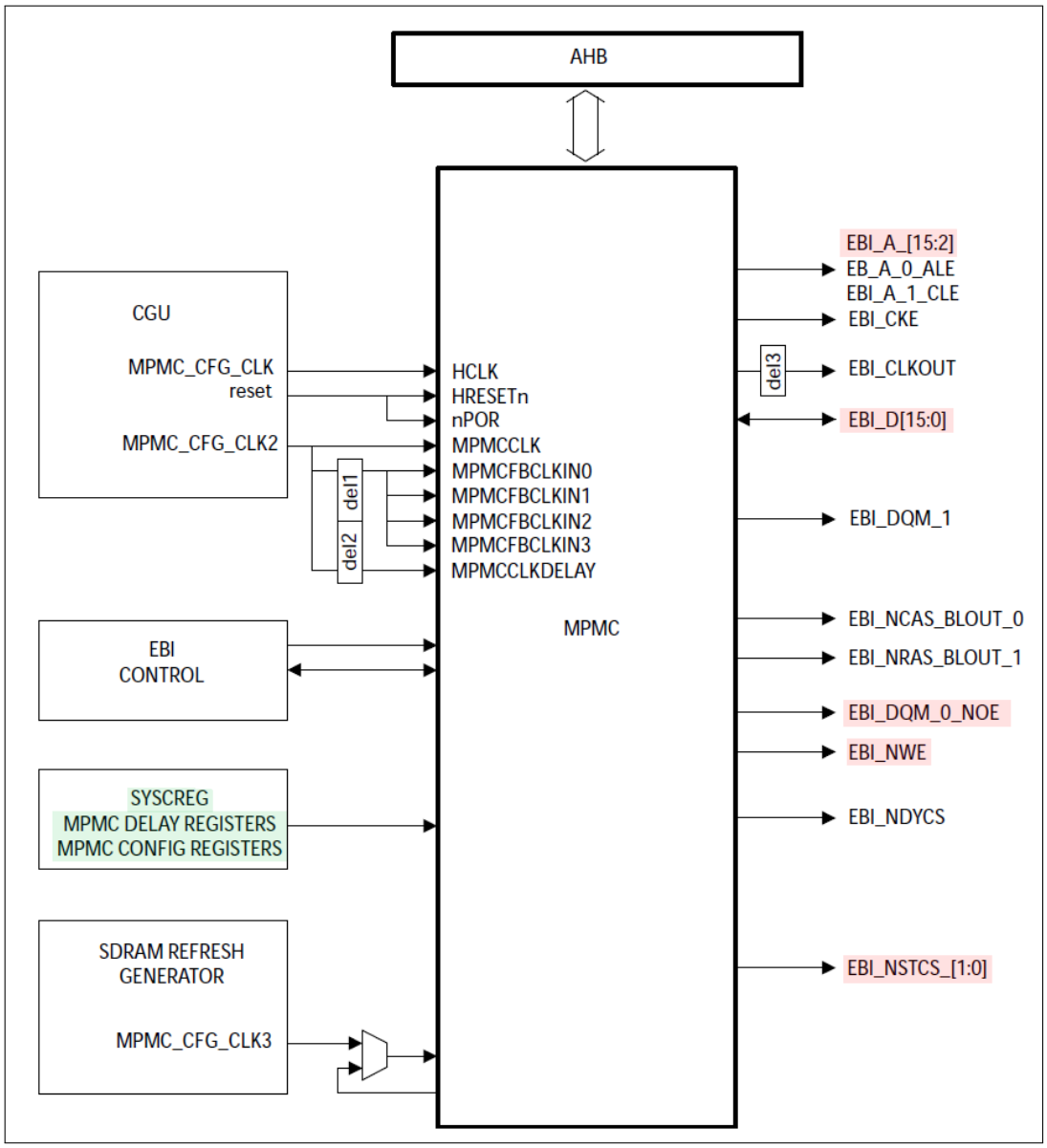


Abbildung 3.3.: NXP LPC313x MPMC, NXP Semiconductors [2010]

Die Register des MPMC werden so konfiguriert, dass die Schnittstelle kompatibel zum Display und dessen Timings wird. Entsprechend dem verwendeten Chip-Select-Signal werden die Register

- MPMCStaticConfig0
- MPMCStaticWaitWen0
- MPMCStaticWaitOen0
- MPMCStaticRd0
- MPMCStaticPage0
- MPMCStaticWr0
- MPMCStaticWaitTurn0

konfiguriert. Die Basisadresse des MPMC ist 0x1700 8000. Wie die Register zu beschreiben sind, geht aus NXP Semiconductors [2010] auf Seite 56 hervor und ist in Tabelle 3.4 gezeigt. Die Timings wurden so gewählt, dass die Timinganforderungen der Displaycontroller eingehalten werden.

Register	Offset	Wert	Beschreibung
MPMCStaticConfig0	0x200	0x81	• 16 Bit Modus • Aktiviert die Nutzung von EBI_nWE • CS low aktiv • keine ExtendedWait-Zyklen • Schreibpuffer deaktiviert • Geschütztes Schreiben deaktiviert • Page Mode deaktiviert
MPMCStaticWaitWen0	0x204	13	13 + 1 = 14 Wartezyklen ab Chip-Select bis Write-Enable
MPMCStaticWaitOen0	0x208	0	0 + 1 = 1 Wartezyklus ab Chip-Select bis Output-Enable
MPMCStaticRd0	0x20C	0	0 + 1 = 1 Wartezyklus ab Chip-Select bis Read-Enable
MPMCStaticPage0	0x210	0	0 + 1 = 1 Wartezyklus für sequential Page Mode Access
MPMCStaticWr0	0x214	15	15 + 2 = 17 Wartezyklen bis Write-Access
MPMCStaticWaitTurn0	0x218	0	0 + 1 = 1 Turnaround Cycles

Tabelle 3.4.: MPMC Register, NXP Semiconductors [2010]

Neben den MPMC-Registern muss das Register SYSCREG_AHB_MPMC_MISC konfiguriert werden. Wird Bit 7 des Registers auf der Adresse 0x1300 2864 mit dem Wert 0 eingestellt, so verändert sich das Adressierungsverhalten dahingehend, dass sich die Adressleitungen des EBI EBI_A[15:0] auf den für den Prozessor sichtbaren AHB[29] Adressbus AHB_A[16:1] verschiebt (siehe NXP Semiconductors [2010], S. 485f). Der Prozessor selbst, kann nun also 17 Bit adressieren, jedoch nur im Sprung von zwei Adressen, da das ursprüngliche LSB weggefallen ist.

[29] AHB: Advanced Microcontroller Bus Architecture

3.2.3. Hardwareverbindung zwischen SRAM-Interface und Display

In diesem Abschnitt wird die Verbindung zwischen dem Prozessor und dem Display behandelt. Eingangs wurde bereits erwähnt, dass beim Kauf der Displays Augenmerk auf Pinkompatibilität gelegt wurde. Das schlägt sich beim Entwurf der Adapterplatine positiv zu Buche, da nun lediglich ein Adapter benötigt wird.
Bereits dargestellt zeigt Abbildung 3.1 auf Seite 11 das Pinout der verwendeten Displays. Der Anschluss an den Prozessor stellt sich wir in Tabelle 3.5 dar. Anhand der gewonnenen Erkenntnisse aus Abschnitt 3.2.1 und Abschnitt 3.2.2 sowie des Schaltplans des verwendeten Gnublin Extended (siehe Sauter [2013b]) kann eine Zuordnung getroffen werden. Nicht verbundene Pins sind mit 'nc'[30] vermerkt.

Nr.	Pin Display	Pin Gnublin	Nr.	Pin Display	Pin Gnublin
1	GND	GND	21	DB0	LPC_DB0
2	+3V3	+3V3	22	DB1	LPC_DB1
3	nc	nc	23	DB2	LPC_DB2
4	RS	LPC_A15	24	DB3	LPC_DB3
5	WR	LPC_WE	25	DB4	LPC_DB4
6	RD	LPC_DQM0	26	DB5	LPC_DB5
7	DB8	LPC_DB8	27	DB6	LPC_DB6
8	DB9	LPC_DB9	28	DB7	LPC_DB7
9	DB10	LPC_DB10	29	nc	nc
10	DB11	LPC_DB11	30	nc	nc
11	DB12	LPC_DB12	31	nc	nc
12	DB13	LPC_DB13	32	nc	nc
13	DB14	LPC_DB14	33	nc	nc
14	DB15	LPC_DB15	34	nc	nc
15	CS	STCS0	35	nc	nc
16	nc	nc	36	nc	nc
17	RESET	GPIO19	37	nc	nc
18	nc	nc	38	nc	nc
19	LED-A	GPIO20	39	nc	nc
20	nc	nc	40	nc	nc

Tabelle 3.5.: Displayverbindung mit dem Gnublin, Coldtears Electronics, Sauter [2013b]

Das Daten-Interface des Displays ist mit den Pins DB[0:15] mit dem Datenbus verbunden. Die Signale Read-Enable RD und Write-Enable WR liegen auf den Pins LPC_DQM0 und LPC_WE. Als Chip-Select wird das Signal STCS0 verwendet. Diese Pins sind aus dem EBI herausgeführt (siehe Abbildung 3.2) und werden, sofern es das System von der Auslastung am Bus ermöglicht, für das Display zur Verfügung gestellt.

[30] nc: not connected

Das RS Signal am Display, welches zwischen Kommando und Daten unterscheidet, liegt auf dem Adresssignal A15. Die folgenden Angaben gehen von einer Registerkonfiguration nach Abschnitt 3.2.2 aus. Werden Daten gesendet, so ist der Pin logisch 1, was einem Wert auf dem Adressbus von 0x10000[31] entspricht. Bei Kommandos ist der Pin logisch 0 mit einem Adresswert von 0x00000[32]. Die unteren 16 Bits des Adressraums lassen sich also willkürlich verändern, da nur das MSB[33] vom Display verwendet wird.

Als RS-Pin ist die Adressleitung A15 (logisch verschoben auf A16) gewählt, da so möglicherweise DMA-Transfers[34] von bis zu 65.536 Bytes[35] möglich sind. Der DMA-Transfer könnte die Adressleitungen bei Daten von 0x10000 bis 0x1FFFF[36] bzw. bei Kommandos von 0x0000 bis 0x0FFFF[37] inkrementieren ohne die Gültigkeit der Wahl zwischen Kommando und Daten des Displays zu beeinträchtigen.

Das Display lässt sich Zusammenfassend also über zwei Pseudoregister für Kommando und Daten auf den Adressoffsets 0x00000 und 0x10000 mit der Basisadresse 0x20000000 ansprechen. Dies ist in Tabelle 3.6 nochmals übersichtlich dargestellt.

Register	Adresse	Typ
SRAM0_DISP_CTRL	0x20000000	Kommandos
SRAM0_DISP_DATA	0x20010000	Daten

Tabelle 3.6.: Adressen für SRAM-Zugriff, NXP Semiconductors [2010]

Die Untersuchung inwieweit DMA-Transfer praktisch mit der verwendeten Hardware möglich ist, ist allerdings nicht Bestandteil dieser Arbeit.

[31] 0x10000 = 0b0001 0000 0000 0000 0000
[32] 0x00000 = 0b0000 0000 0000 0000 0000
[33] MSB: Most Sigificant Bit, das höchstwertige Bit
[34] DMA: Direct Memory Access, Speichertransfer effizient und schnell direkt in Hardware
[35] $65.536 = 2^{16}$
[36] 0x1FFFF = 0b0001 1111 1111 1111 1111
[37] 0x0FFFF = 0b0000 1111 1111 1111 1111

3.2.4. Adapterplatine zwischen Gnublin Extended und Display

Der Adapter wird als Platine realisiert, die auf den Gnublin Extended aufgesteckt wird. Das Display wiederum wird mit der Adapterplatine steckbar verbunden. Der Schaltplan ist in Abbildung 3.5 gezeigt und stellt entsprechend Tabelle 3.5 die Verbindungen her.

Der grün markierte Bereich stellt die Verbindung zum Display dar, rot zum Gnublin Extended und im blauen Rechteck sind weitere kleine Bauteile untergebracht. Hier sind Pullup-Wiederstande mit 10 kΩ an den Leitungen STCS0, Reset und LED-A um definierte Pegel vorzugeben zu sehen sowie einen Blockkondensator mit 100 nF, der für eine bessere Spannungsversorgung des Displays sorgt.

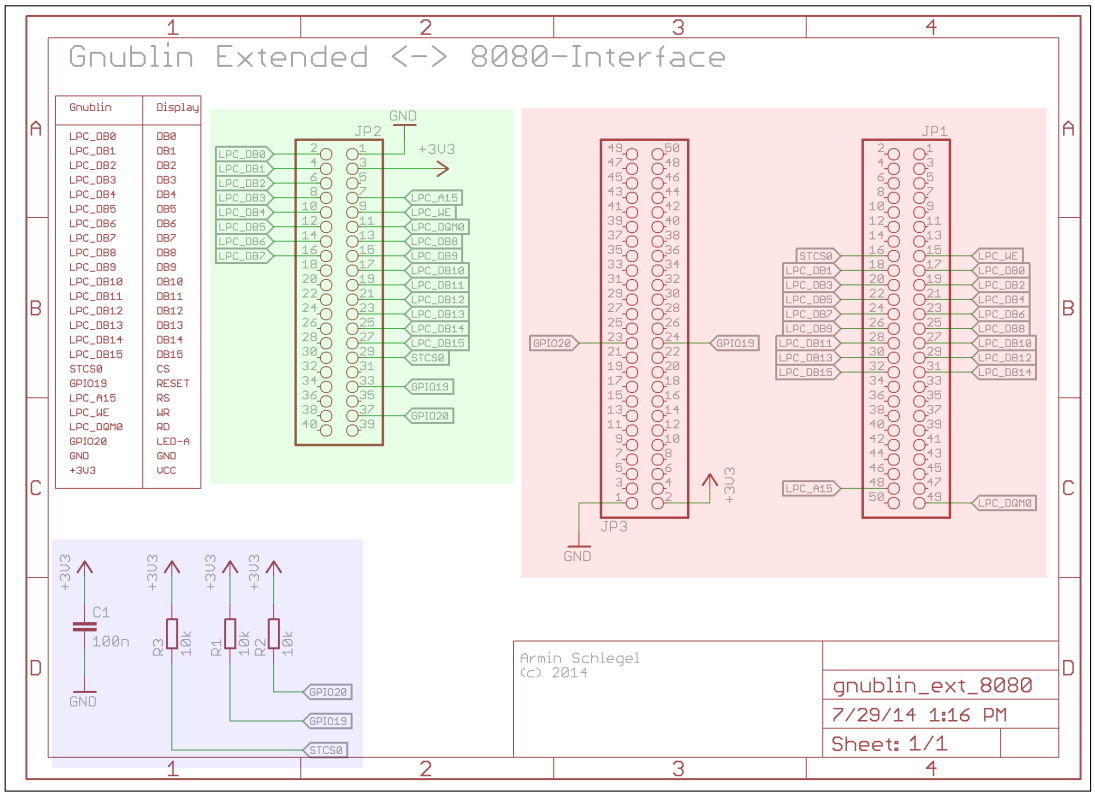
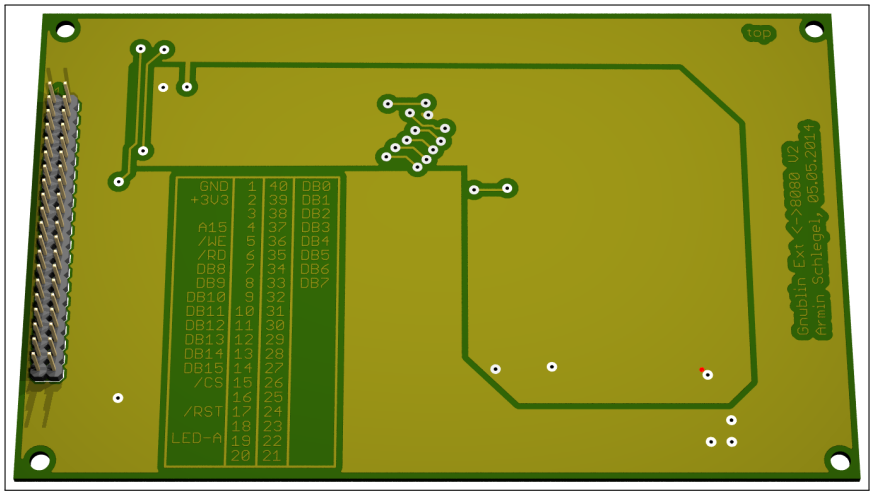
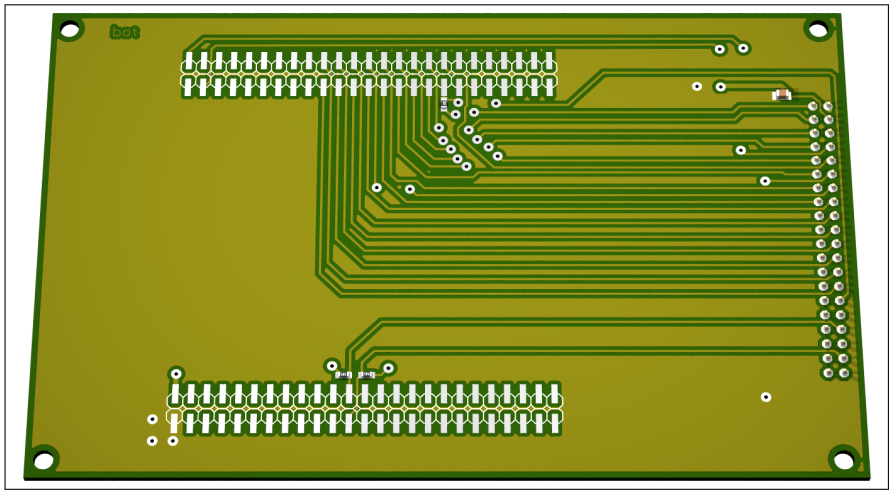


Abbildung 3.4.: Schaltplan Adapterplatine



(a) Adapterplatine Top Layer



(b) Adapterplatine Bottom Layer

Abbildung 3.5.: Adapterplatine zwischen Gnublin Extended und Display

Abbildung 3.5 (a) und (b) zeigt je ein gerendertes 3D Bild der Ober- und Unterseite der Adapterplatine. Auf der Oberseite wird das Display auf der linken Seite mit der 40 poligen Stiftleiste angeschossen. Mit der Unterseite wird die Platine auf das Gnublin Extended mit je zwei 50 poligen Buchsenleisten aufgesteckt.

Die Schaltplan und das Layout befinden sich in der CD im Anhang dieser Arbeit.

3.2.5. Software

Im Folgenden Abschnitt wird die Software behandelt, die nötig ist, um das Display zu betreiben. Die Softwareentwicklung ist in drei Teile auf gegliedert:

- Modifikationen im Bootloader APEX
- Userspace-Treiber basierend auf einem Treibers auf für den Raspberry Pi (siehe Schlegel [2013a] und Schlegel [2013b]) bei dem mittels GPIO-Pins ein 8080-Display zu betreiben
- Framebuffer-Treiber im Linux-Kernel

3.2.5.1. Anpassung des APEX-Bootloaders zur Verwendung des Displays

Der APEX-Bootloader[38] wurde ursprünglich für Prozessoren de Sharp LH Familie entwickelt, wurde allerdings auf eine Vielzahl von weiteren ARM basierten Prozessoren portiert, so auch für die verwendete NXP LPC313x CPU[39]. Die Aufgabe des Bootloaders ist es, grundlegende prozessorinterne Hardwareeinheiten wie z.B. CGU oder SD-RAM zu initialisieren um für den Linux-Kernel die notwendige Umgebung zu schaffen. Im Anschluss wird der Linux-Kernel geladen und gestartet. Es werden außerdem dem Linux-Kernel Bootparameter übergeben, die zum Start benötigt werden. Am Anfang des Bootloader-Codes werden die verwendeten MPMC-Register konfiguriert. Wie in Abschnitt 3.2.2 muss das SYSCREG-Register SYSCREG_AHB_MPMC_MISC nicht explizit beschrieben werden, da es im Resetzustand bereits richtig konfiguriert ist. Listing 3.1 zeigt die entsprechende Initialisierung der MPMC-Register für das MD050SD. Die Adressen von z.B. MPMC_STCONFIG0 sind in der `lpc313x.h` definiert. Die Modifikationen im Bootloader finden in der Datei `initialize.c` statt.

```
1  #elif defined(CONFIG_MACH_EPLPC3131_V1)
2  #if defined(CONFIG_DISP_SSD1963)
3    // ...
4  #elif defined(CONFIG_DISP_MD050SD)
5     /* LCD display, 16 bit */
6     MPMC_STCONFIG0  = 0x81;
7     MPMC_STWTWEN0   = 13;
8     MPMC_STWTOEN0   = 0;
9     MPMC_STWTRD0    = 0;
10    MPMC_STWTPG0    = 0;
11    MPMC_STWTWR0    = 15;
12    MPMC_STWTTURN0  = 0;
13 #elif defined(CONFIG_DISP_SSD1289)
14   // ...
15 #elif defined(CONFIG_DISP_NONE)
```

[38] https://gitorious.org/apex/
[39] CPU: Central Processing Unit, Prozessor

```
16    // ...
17 #endif
18 #endif
```

Listing 3.1: Bootloader: MPMC-Konfiguration

3.2.5.1.1. Boot-Logo im APEX-Bootloader Um dem Display beim Systemstart einen initialisierten Zustand zu geben und dem Benutzer bereits während dem Laden des Linux-Kernels ein Bild anzuzeigen, ist ein Bootlogo konfigurierbar. Hierfür bedarf es eines rudimentären Displaytreibers im Bootloader. Im Folgenden wird die Darstellung des Boot-Logos unter Verwendung des MD050SD dargestellt. Listing 3.2 zeigt den ersten Teil des Treibers, bei dem grundlegende Datentypen sowie Sendefunktionen für Daten und Kommandos gelistet sind.

```
1  #if defined(CONFIG_DISP_MD050SD) || defined(CONFIG_DISP_SSD1963) ||
       defined(CONFIG_DISP_SSD1289)
2  #define DISP_PHYS        (EXT_SRAM0_PHYS)
3  #define DISP_PHYS_CTRL   (DISP_PHYS + 0)
4  #define DISP_PHYS_DATA   (DISP_PHYS + 0x10000)
5
6  unsigned int width;
7  unsigned int height;
8  int pixel;
9
10 struct display {
11    volatile u16* ctrl;
12    volatile u16* data;
13 };
14
15 static struct display display;
16
17 static void display_send_cmd(u16 cmd)
18 {
19    *display.ctrl = 0x00FF & cmd;
20 }
21
22 static void display_send_data(u16 data)
23 {
24    *display.data = data;
25 }
26
27 #endif
```

Listing 3.2: Bootloader: Grundlegende Datentypen und Funktionen

Die Struktur `struct display` ab Zeile 10 von Listing 3.2 enthält zwei Zeiger `u16* ctrl` und `u16* data` auf die jeweiligen Adressen aus Tabelle 3.6 für Kommandos und Daten. In den Zeilen 17 und 22 sind die zwei Sendefunktionen `display_send_cmd(u16 cmd)` und `display_send_data(u16 cmd)` definiert. Hiermit werden die Kommandos und Daten an das Display gesendet. Wird auf eine der beiden Adressen ein Wert geschrieben, kümmert

sich das MPMC und das EBI automatisch um die restlichen Signale wie WR, RD und CS.

```
1  #if defined(CONFIG_DISP_MD050SD) || defined(CONFIG_DISP_SSD1963) ||
       defined(CONFIG_DISP_SSD1289)
2     display.ctrl = &__REG16 (DISP_PHYS_CTRL);
3     display.data = &__REG16 (DISP_PHYS_DATA);
4  #if defined(CONFIG_DISP_MD050SD)
5     GPIO_OUT_LOW(IOCONF_GPIO, _BIT(14)); //GPIO20 is LED_ENABLE
6
7     GPIO_OUT_LOW(IOCONF_GPIO, _BIT(13)); //GPIO19 is nRESET
8     udelay(20000);
9     GPIO_OUT_HIGH(IOCONF_GPIO, _BIT(13)); //GPIO19 is nRESET
10    udelay(20000);
11
12   /* Set Window from 0,0 to 479, 799 */
13    display_send_cmd(0x0002);
14    display_send_data(0);
15    display_send_cmd(0x0003);
16    display_send_data(0);
17
18    display_send_cmd(0x0006);
19    display_send_data(480 - 1);
20    display_send_cmd(0x0007);
21    display_send_data(800 - 1);
22
23   /* Clear the display with color black */
24    display_send_cmd(0x000F);
25
26    for(pixel = 0; pixel < 800 * 480; pixel++)
27    {
28       display_send_data(0x0000);
29    }
30
31    GPIO_OUT_HIGH(IOCONF_GPIO, _BIT(14)); //GPIO20 is LED_ENABLE
32
33 #if defined(CONFIG_LOGO_TUX)
34    width = boot_logo_tux[0];
35    height = boot_logo_tux[1];
36
37    display_send_cmd(0x0002);
38    display_send_data(480/2 - (height - 1)/2);
39    display_send_cmd(0x0003);
40    display_send_data(800/2 - (width - 1)/2);
41    display_send_cmd(0x0006);
42    display_send_data(480/2 + (height - 1)/2 + 1);
43    display_send_cmd(0x0007);
44    display_send_data(800/2 + (width - 1)/2 + 1);
45    display_send_cmd(0x000F);
46
47    for(pixel = 2; pixel < width * height + 2; pixel++)
48    {
49       display_send_data(boot_logo_tux[pixel]);
50    }
51 #endif
52 #endif
53 #endif
```

Listing 3.3: Bootloader: Display-Initialisierung und Bootlogo

Der Eigentliche Treiber und der Code für das Anzeigen des Bootlogos ist in Listing 3.3 zu sehen. In Zeile 2 und 3 werden den Adresszeigern die pysikalischen Adressen zum Schreiben von Kommandos und Daten zugewiesen. Von Zeile 5 bis 10 wird die Hintergrundbeleuchtung explizit abgschaltet und ein Reset auf das Display gegeben. Da das MD050SD einen CPLD-Controller besitzt, welcher eine auf genau dieses TFT-Panel zugeschnittene Programmierung enthält, fällt eine Initialisierungsroutine weg. Dies wäre bei Controllern wie dem SSD1963 nicht der Fall, da diese mit einer Vielzahl von Panels arbeiten können und demzufolge eine spezielle Initialisierung brauchen. Das MD050SD ist nach dem Reset also initialisiert und erwartet Kommandos. Von Zeile 13 bis 24 in Listing 3.3 wird ein Bereich im Display-RAM der vollen Bildschirmgröße reserviert und die Bereitschaft zum Datenempfang gestartet (siehe Tabelle 3.3). Alle Daten die im Anschluss gesendet werden, werden automatisch an die richtige Stelle im Display geschrieben. In einer Schleife werden ab Zeile 26 alle reservierten Pixel von zuvor mit der Farbe Schwarz beschrieben, um das automatisch angezeigte Testbild des Displays beim Start zu überschreiben. Im Anschluss wird die Hintergrundbeleuchtung wieder eingeschaltet. Ist über den Codeswitch `CONFIG_LOGO_TUX` das Bootlogo aktiviert, so wird dies ab Zeile 34 angezeigt. Die Größe des Logos wird in den Zeilen 34 und 35 ausgelesen und in den Folgenden angezeigt. Die Pixeldaten des Logos sind im Array `u16 boot_logo_tux[]` in den Dateien `boot_logo_tux.c` und `boot_logo_tux.h` hinterlegt.

3.2.5.1.2. Konfiguration des APEX-Bootloaders Der APEX-Bootloader besitzt zur Konfiguration dasselbe System wie der Linux-Kernel. Dieses System heißt KConfig und wird durch den Befehl `make menuconfig` aufgerufen, welches ein Konfigurationsmenü im aufrufenden Terminal erscheinen lässt. Hier sind neben Grundlegenden Konfiguration zum Beispiel für die Prozessorarchitektur oder Taktraten auch der Displaytreiber und das Bootlogo eingepflegt. Abbildung 3.6 zeigt exemplarisch den Unterpunkt `Platform Setup` bei dem das Display MD050SD und das Bootlogo ausgewählt sind.

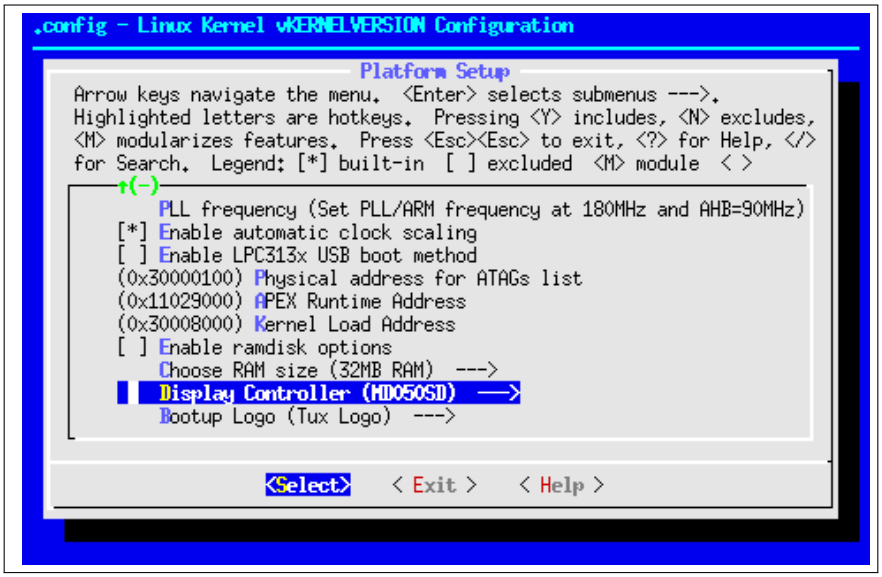


Abbildung 3.6.: APEX-Bootloader KConfig

Bevor der Bootloader konfiguriert werden kann, muss der Vanilla-Quellcode noch gepatcht werden. Listing 3.4 zeigt die einzelnen Schritte, um den Bootloader herunterzuladen und zu Patchen. Der Patch enthält alle nötigen Dinge zum Betrieb des MD050SD-Display und des Boot-Logos.

```
1 $ wget https://github.com/embeddedprojects/gnublin-distribution/raw/
    master/lpc3131/bootloader/apex/1.6.8/apex-1.6.8.tar.gz
2 $ tar xvfz apex-1.6.8.tar.gz
3 $ wget https://github.com/siredmar/master/raw/master/Teil_A/software/
    bootloader/apex_display.patch
4 $ cd apex-1.6.8
5 $ patch -p1 < ../apex_display.patch
```

Listing 3.4: Bootloader: Herunterladen und Patchen

Im Folgenden wird die Konfiguration des Bootloaders und die damit möglichen Einstellungsmöglichkeiten beschrieben. Im Konfigurationsmenü sind nach dem patchen im Untermenü `Platform Setup` die Optionen `Display Controller` und `Bootup Logo` verfügbar. Hier wird die Auswahl bezüglich Displaycontrollers und Logo getroffen. Wird ein Displaycontroller anders als MD050SD gewählt, so werden nur entsprechende Timings der MPMC-Register gesetzt und kein Boot-Logo angezeigt. Um den Kernel mit spezifishen Parametern starten zu können, werden für den Betrieb der unterschiedlichen Treiber (Framebuffer, User-Space) andere Bootparameter benötigt. So ist der originalen Parameterliste `console=ttyS0,115200n8 root=/dev/mmcblk0p3 rw rootfstype=ext4 rootwait` im Untermenü `Environment` für den Betrieb mit dem Framebuffer-Treiber ein `fbcon=rotate:0 fbcon=font:VGA8x16` hinzuzufügen um dem Kernel beim Start Informationen über den Kernel-Treiber `fbcon` mitzuteilen. Wird der User-Space-Treiber

verwendet, so übernimmt das Kernel-Modul `vfb`[40] die Aufgabe des Framebuffers. Die Parameterliste ist mit `console=tty0 video=vfb: vfb_enable=1` zu ergänzen (siehe Antonino Daplas [2005]).

Der Bootloader wird per `make apex.bin` kompiliert und mit `dd if=src/arch-arm/rom/apex.bin of=/dev/sdXY` [41] auf die SD-Karte des Gnublin geschrieben (siehe Sauter [2013a]).

3.2.5.2. Entwicklung eines Linux-Framebuffer-Treibers

3.2.5.2.1. Anpassungen für SSD1963 Controller

3.2.5.2.2. Anpassungen für SSD1289 Controller

3.2.5.2.3. Anpassungen für CPLD Controller

3.2.5.3. Entwicklung eines User-Space-Treibers

3.2.5.3.1. Anpassungen für SSD1963 Controller

3.2.5.3.2. Anpassungen für SSD1289 Controller

3.2.5.3.3. Anpassungen für CPLD Controller

3.2.5.4. Probleme bei der Entwicklung und Fehlersuche

3.2.5.4.1. Probleme mit SSD1963

3.2.5.4.1.1. Rolle des User-Space-Treibers

3.2.5.4.1.2. Debuggen mit Logik-Analyzer

[40] vfb: Virtual Frame Buffer
[41] /dev/sdXY: bezieht sich auf die Bootpartition auf der korrekten SD-Karte, z.b. /dev/sdb2

3.3. Known Bugs

3.3.1. Known Bugs SSD1963

3.3.2. Known Bugs SSD1289

3.3.3. Known Bugs CPLD Display

4. Teil B

4.1. Konzept

4.2. Hardwareentwicklung

4.2.1. Spannungsversorgung

4.2.2. HDMI-RGB-Bridge

4.2.3. RGB-LVDS-Bridge

4.2.4. EDID-Daten

4.3. Software

4.3.1. EDID-Daten auf embedded Seite

4.3.1.1. Konzept

4.3.1.2. Low-Level-Treiber

4.3.1.2.1. UART-Treiber

4.3.1.2.2. I2C-Treiber

4.3.1.3. Programmablauf

4.3.2. EDID-Daten auf PC Seite

4.3.2.1. Konzept

4.3.2.2. GTK GUI mit Glade

4.3.2.3. Programmablauf

4.4. Known Bugs

4.4.1. Hardware

4.4.1.1. HDMI-Stecker gekreuzt, CON2

4.4.1.2. LVDS-Steckerfootprint gespiegelt, CON6

4.4.1.3. +12V PWM Hintergrundbeleuchtung

4.4.1.4. +5V Kreis / Widerstand R13

4.4.1.5. USB D+/D- vertauscht R22, R23

4.4.2. Software

4.4.2.1. EDID Programmer

5. Zusammenfassung

Literaturverzeichnis

Antonino Daplas 2005

Antonino Daplas, Linux-Kernel: *The Framebuffer Console*. https://github.com/torvalds/linux/blob/master/Documentation/fb/fbcon.txt. Version: 2005, Abruf: 04.08.2014 3.2.5.1.2

Brandt 2013

Brandt, Mathtias: *Fast 80 Prozent Marktanteil für Android*. http://de.statista.com/themen/581/smartphones/infografik/1326/smartphone-absatz-weltweit. Version: 2013, Abruf: 20.05.2014 1.1

Coldtears Electronics

Coldtears Electronics, CE: *Schaltplan Gnublin Extended V1.7*. https://github.com/siredmar/master/raw/master/Recherche/Displays/8080/5Inch_SSD1963/schematic.pdf, Abruf: 29.07.2014 (document), 3.1, 3.5

Extron 2014

Extron: *DVI and HDMI: The Short and the Long of It*. http://www.extron.com/company/article.aspx?id=dvihdmi_ts. Version: 2014, Abruf: 20.05.2014 2.1.3

ITEAD Studios 2013

ITEAD Studios, ITEAD: *MD070SD Datasheet*. https://github.com/siredmar/master/raw/master/Recherche/Displays/8080/5Inch_MD050SD/DS_IM130820001.pdf. Version: 2013, Abruf: 22.05.2014 (document), 3.3

Knuppfer 2010

Knuppfer, Nick: *Leading PC Companies Move to All Digital Display Technology, Phasing out Analog*. http://newsroom.intel.com/community/intel_newsroom/blog/2010/12/08/leading-pc-companies-move-to-all-digital-display-technology-phasing-out-analog?cid=rss-258152-c1-262653. Version: 2010, Abruf: 20.05.2014 2.1.1

Leunig 2002

Leunig, Peter H.: *Der DVI-Standard - ein Überblick*. http://www.leunig.de/_pro/downloads/DVI_WhitePaper.pdf. Version: 2002, Abruf: 20.05.2014 2.1.2

NXP Semiconductors 2010

NXP Semiconductors, NXP: *LPC3130/31 User Manual*. https://github.com/siredmar/master/raw/master/Recherche/Gnublin/datasheets/user.manual.lpc3130.lpc3131.pdf. Version: 2010, Abruf: 28.07.2014 (document), 3.2, 3.2.2, 3.3, 3.2.2, 3.4, 3.2.2, 3.6

Sauter 2013a

Sauter, Benedikt: *Bootloader uebersetzen und installieren*. http://wiki.gnublin.org/index.php/Bootloader_übersetzen_und_installieren. Version: 2013, Abruf: 04.08.2014 3.2.5.1.2

Sauter 2013b

Sauter, Embedded P.: *LPC3130/31 User Manual*. https://github.com/siredmar/master/raw/master/Recherche/Gnublin/datasheets/GNUBLIN_EXTENDED_V1_7.pdf. Version: 2013, Abruf: 28.07.2014 (document), 3.2.3, 3.5

Schlegel 2013a

Schlegel, Armin: *Ansteuerung eines TFT-Displays mit dem Raspberry Pi über die GPIO-Pins*. https://github.com/siredmar/siredmar_projects/raw/master/embedded/RPi/RPI_SSD1963/doc/RPI_SSD1963_24_11_2013.pdf. Version: 2013, Abruf: 28.07.2014 3.2.5

Schlegel 2013b

Schlegel, Armin: *Ansteuerung eines TFT-Displays mit dem Raspberry Pi über die GPIO-Pins - Quellcode*. https://github.com/siredmar/siredmar_projects/tree/master/embedded/RPi/RPI_SSD1963/sw. Version: 2013, Abruf: 28.07.2014 3.2.5

Solomon Systech Limited 2007

Solomon Systech Limited, SSD: *SSD1289 Datasheet*. http://www.kosmodrom.com.ua/el/STM32-TFT/SSD1289.pdf. Version: 2007, Abruf: 22.05.2014 (document), 2.1.6, 2.3, 3.2

Solomon Systech Limited 2008

Solomon Systech Limited, SSD: *SSD1963 Datasheet*. https://github.com/siredmar/master/raw/master/Recherche/Displays/8080/5Inch_SSD1963/SSD1963datasheet.pdf. Version: 2008, Abruf: 22.05.2014 (document), 3.1

Texas Instruments 2011

Texas Instruments, TI: *TI PanelBus™ DIGITAL RECEIVER - TFP401A-EP*. www.ti.com/litv/slds160a. Version: 2011, Abruf: 22.05.2014 (document), 2.2

Valcarce 2011

Valcarce, Javier: *VGA Video Signal Format and Timing Specifications*. http://www.javiervalcarce.eu/wiki/VGA_Video_Signal_Format_and_Timing_Specifications. Version: 2011, Abruf: 22.05.2014 (document), 2.1.1, 2.1

Eidesstattliche Erklärung

Ich, Armin Schlegel, Matrikel-Nr. 2020863, versichere hiermit, dass ich meine Masterarbeit mit dem Thema

> *Entwicklung und Optimierung von Display-Schnittstellen fuer embedded Linux Boards -*

selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, wobei ich alle wörtlichen und sinngemäßen Zitate als solche gekennzeichnet habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Mir ist bekannt, dass ich meine Masterarbeit zusammen mit dieser Erklärung fristgemäß nach Vergabe des Themas in dreifacher Ausfertigung und gebunden im Prüfungsamt der Ohm-Hochschule abzugeben oder spätestens mit dem Poststempel des Tages, an dem die Frist abläuft, zu senden habe.

Nuernberg, den 4. August 2014

Armin Schlegel

A. Anhang

Toolchain fuer NXP LPC313x