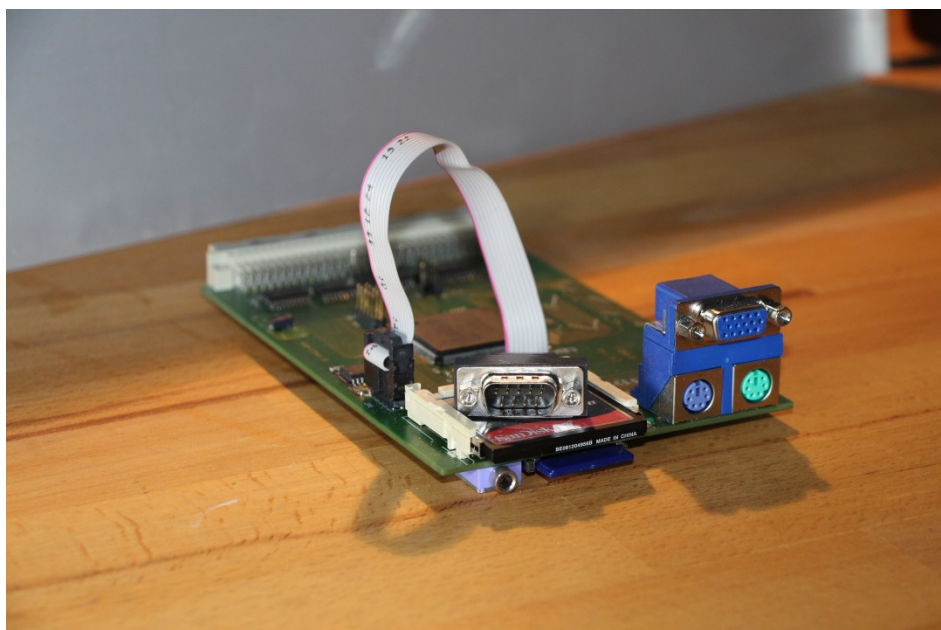
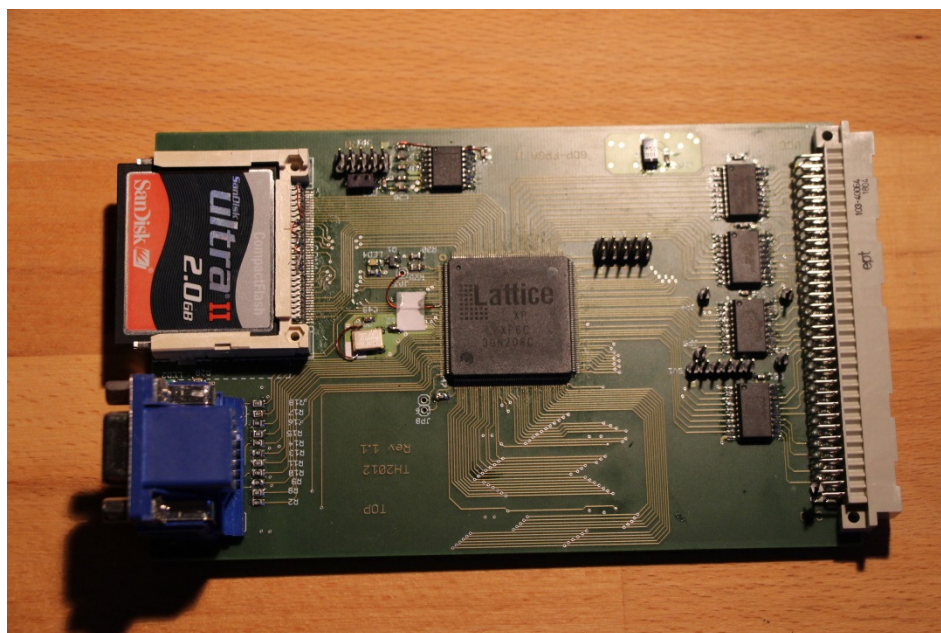


GDPFPGAII

GDP Baugruppe (nicht nur) für den NDR-Klein-Computer (ECB-Variante)

Stand: Februar 2015

Autor: Torsten Hemmecke



Wichtiger Hinweis:

Die in dieser Anleitung wiedergegebenen Schaltungen und Verfahren werden ohne Rücksicht auf die Patentlage oder Lizenzrechte Dritter mitgeteilt. Sie sind ausschließlich für private Zwecke und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden. *)

Alle Schaltungen und technische Angaben in dieser Anleitung wurden vom Autor sorgfältig erarbeitet bzw. zusammengestellt. Trotzdem sind Fehler nicht auszuschließen. Daher kann der Autor weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernehmen. Für die Mitteilung eventueller Fehler ist der Autor jederzeit dankbar.

Die Rechte an Firmennamen, Logos und Warenzeichen, die in dieser Anleitung genannt werden, liegen bei den jeweiligen Inhabern.

**) Bei gewerblicher Nutzung ist vorher die Genehmigung des möglichen Lizenz- oder Rechteinhabers einzuholen.*

Inhalt

Inhalt.....	3
1 Vorwort.....	5
2 Änderungshistorie	6
3 Kurzbeschreibung der Funktion	7
4 Technische Daten.....	8
5 Prinzipbeschreibung.....	9
5.1 Blockschaltbild.....	9
5.2 Beschreibung des Blockschaltbildes.....	9
5.3 Das FPGA (XPF6C).....	10
5.3.1 Blockschaltbild.....	10
5.3.2 Funktionsbeschreibung	10
5.3.3 VHDL-Quellcode	11
5.3.4 Programmierung	11
5.3.5 Register der GDPHS / Änderungen zum Original	11
5.4 Das CPLD.....	20
5.4.1 Blockschaltbild.....	21
5.4.2 Funktionsbeschreibung	21
5.4.3 Programmierung	21
6 Aufbauanleitung	22
6.1 Umgang mit IC's	22
6.2 Aufbau Schritt für Schritt.....	22
7 Inbetriebnahme	23
7.1 Betrieb mit CPU68K20 (M68020)	23
7.2 Betrieb mit CPU68K00 (M68000)	23
8 Version mit LFXP2C	24
8.1 Blockschaltbild.....	24
9 Fehlerkorrekturen.....	25
10 Quellenhinweise	26
11 Anhänge	27
11.1 Steckerbelegungen.....	27
11.2 Schaltplan (LFXP6)	28
11.3 Stückliste (LFXP6)	32
11.4 Bestückungsplan (LFXP6).....	34

11.5	Layout Oben (LFXP6)	36
11.6	Layout Unten	36
11.7	Schaltplan (LFXP2C)	37
11.8	Stückliste (LFXP2C)	41
11.9	Bestückungsplan (LFXP2C)	43
11.10	Layout Oben (LFXP2C)	45
11.11	Layout Unten (LFXP2C)	45

1 Vorwort

Die Baugruppe GDPFPGAII entstand im Rahmen des Projektes „NKC Redesign“ das zum Ziel hat, den legendären NDR-Klein-Computer mit heute (Stand 2013) erhältlichen Bauelementen neu aufzubauen und bis hin zum Einsatz mit einem kleinen Linux-Kernel. Das ganze System verwendet einen ECB Bus, wobei die Belegung des Busses zum MC-Computer kompatibel ist und darüber hinaus 16/32bit Erweiterungen vorsieht.

Die GDPFPGAII soll volle Software-Kompatibilität zum original NKC bieten. Sie besteht aus einem 8/16-Bit ECB Bus Interface, einem XILINX XC9536 CPLD zur teilweisen Adressdekodierung eines LFXP6C3QN208 FPGA . Das FPGA hat 16-Bit breiten Zugriff auf 1MB SDRAM (8ns) als Video-Speicher, SDCARD und CF-CARD (GIDE) Interface sowie PS2-Mouse/-Tastatur, einen RS232 Port (über Steck-Verbinder) und einen VGA Stecker.

Das FPGA realisiert in der Version 1.0 (LFXP6) und 2.0 (LFXP2C) die GDP64-HS von Andreas als Wishbone-Modul. Als weiteres WB-Modul ist die GIDE mit CPU-Wait-Generierung implementiert. Der Speicher ist ebenfalls als Modul realisiert und es kann direkt darauf zugegriffen werden.

Zusammen mit einer CPU68K00 oder CPU68K20 und einer passenden Backplane steht damit ein vollständiges NKC System zur Verfügung, das bereits JADOS über eine SD-CARD bootet.

Prinzipiell können mit der Karte auch andere Hardware-Varianten realisiert werden, z.B. ein VGA Core oder ein „echtes“ ATA.

2 Änderungshistorie

Hardware-Revision:

1.3 – keine Fehler bekannt (LFXP6C ist EOP, Nachfolger ist LFXP2C)

2.1- **Ungetestet** (für LFXP2C)

FPGA-Revision:

LFXP6C-150223 – Für Hardware-Revision 1.3 (LFXP6C-3QN208C)

LFXP2C-150303 – Für Hardware-Revision 2.1 (LFXP2C-8E-5QN208C)

3 Kurzbeschreibung der Funktion

Die Baugruppe GDPFPGAII stellt alle notwendige Peripherie für den NKC zur Verfügung: GDPHS (A.Voggeneder) mit SD-Card Interface, PS2-Mouse/Keyboard, Serielle Schnittstelle (RS232), 2x Audio-Ausgang, GDPHS mit VGA Ausgang sowie eine GIDE. Außerdem ist der Zugriff auf den Bildschirmspeicher direkt mit 8/16 Bit möglich. Die einzelnen VHDL Module sind durch WishBone verbunden, was Erweiterungen erleichtert. Die Hardware ist auf einer Europakarte aufgebaut und wird mit einem 16-Bit ECB-Bus-Anschluß betrieben. Durch die Verwendung von CPLD und FPGA Baustein ist Raum für Erweiterungen gegeben.

Zusätzlich zu den in den originalen NKC Karten und den Erweiterungen von A.Voggeneder sind noch ein Hardware-Cursor (CTRL2.6) und die Möglichkeit die Tastatur in ScanCode-Mode zu betreiben (port 0x67) implementiert.

4 Technische Daten

Europakarte 160 x 100 mm 4 Lagen (2 Supply Planes)

FPGA: LFXP6C-3QN208C in der HW-Version 1.x

FPGA: LFXP2C-8E-5QN208C in der HW-Version 2.x

ECB-Bus (8/16-bit)

Betriebsspannung 5V

Adressraum voll ausdekodiert

1MB SRAM (8/10ns – z.B. CY7C1041CV33) , 2MB mit LFXP2C

VGA/GDPFPGA

SD-CARD

CF-CARD/ATA

PS2Mouse/Keyboard

RS232

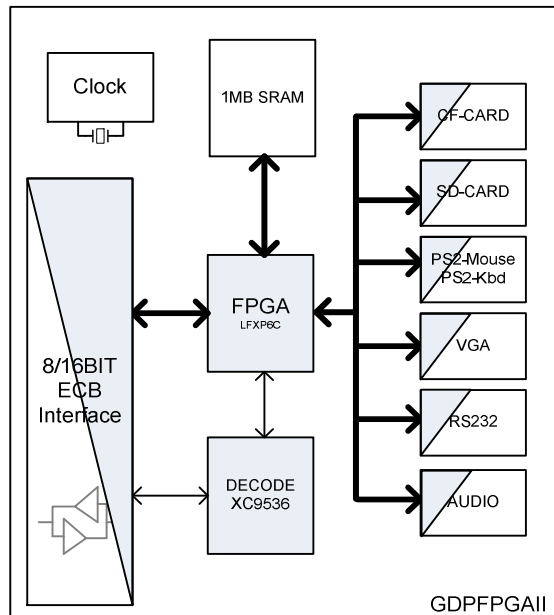
AUDIO

Ethernet (LFXP2C)

5 Prinzipbeschreibung

Die folgende Beschreibung bezieht sich auf die Variante mit LFXP6C. Die Unterschiede in der LFXP2C Variante werden im Abschnitt Version mit LFXP2C erläutert.

5.1 Blockschaltbild



5.2 Beschreibung des Blockschaltbildes

Die Karte besteht aus den Blöcken ECB-Interface, Adress-Dekodierung, SRAM (1MB) sowie der Peripherie (CF-Card, SD-Card, PS2, VGA-Anschluß, RS232, Audio).

Die Hardware des Businterface besteht im Wesentlichen aus den 74LVC245 Bustreibern, die die Umsetzung der Signale zwischen 5V und 3.3V Welt realisieren. Das CPLD XC9563 dekodiert den Adress-Bus vollständig und generiert das Card-Select für IO- und MEM-Zugriffe. Außerdem werden einige Signale wie INT, WAIT ... zwischen 5V ECB Bus und 3.3V FPGA konvertiert. Das Signal F_NMI wird dazu „mißbraucht“ dem CPLD die Verwendung des M68000 zu signalisieren. Damit werden dann nur die Adressleitungen bis A22 dekodiert. Soll das Signal F_NMI in seinem ursprünglichen Sinne verwendet werden, müssen die Adressleitungen A24...29 auf der CPU68K00 auf L gezogen werden damit die Adressdekodierung korrekt funktioniert.

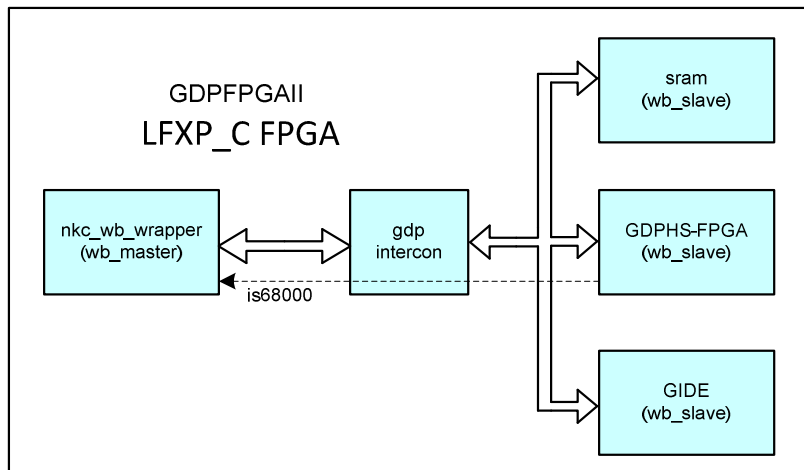
Die Adressierung der Komponenten entspricht den originalen Projekten. Der Zugriff auf das SRAM erfolgt Memory-Mapped. Das SRAM wird dazu ab 0x70.0000*cpu als 4MB Block (cpu=2 für 68000, cpu=4 für 68020) 16Bit breit in den Speicher eingeblendet. Im CPLD der CPU68K20 muß dazu eine Anpassung vorgenommen werden um DTACK0 und DTACK1 korrekt zu erzeugen.

Die farblich hervorgehobenen Blöcke sind im CPLD/FPGA realisiert.

Die Karte ist in allen Funktionen mit CPU68K00-12MHz und CPU68K020-25MHz getestet.

5.3 Das FPGA (XPF6C)

5.3.1 Blockschaltbild



5.3.2 Funktionsbeschreibung

Die Funktionalität des FPGA umfaßt in der vorliegenden Version die der GDPHS64NEUv2 von Andreas, die GIDE und den direkten Zugriff auf den Bildschirmspeicher. Alle Module sind via WishBone miteinander verbunden um ein Standard-Modell für zukünftige Erweiterungen zu haben und die Anpassung neuer Module zu vereinfachen.

Die Anbindung an den NKC Bus erfolgt über ein Wrapper-Modul, das die notwendige Anpassung an den WishBone Bus vornimmt.

Das Modul gdp_intercon verbindet alle Master und Slave Module miteinander.

Abweichend zur GDPHS64NEU wird der Bildschirmspeicher nicht direkt an das GDP-Modul angebunden, sondern ebenfalls über den WishBone-Bus als eigenes Modul angekoppelt. Dadurch kann direkt auf dem Bildschirmspeicher zugegriffen werden, und zwar 8 oder 16 Bit breit.

Das Signal is68000 wird im Option-Register 0x67 (NEU) erzeugt, um die Busdekodierung zwischen 68020 (dynamic bus sizing) und 68000 umzuschalten. Das Register wird weiter Unten beschrieben.

5.3.3 VHDL-Quellcode

Im Unterverzeichnis FPGA\src befinden sich die VHDL Quellcodes für die GDP-FPGA sowie die CPU68K20 mit den notwendigen Änderungen der DTACK Signale. Das Unterverzeichnis FPGA\jed enthält die fertigen Jedec-Files zum Programmieren der CPLDs/FPGAs.

Im Unterverzeichnis FPGA\src integriert ist eine GIT Versionskontrolle, mit der alle Varianten (auch die LFXP2) selektiert werden können.

Das Verzeichnis FPGA\src\Xilinx das Projekt für den Xilinx CPLD (Adressdekodierung).

Das Verzeichnis FPGA\src\GDPFPGAII\vhdl\GDP64HSFPGA enthält die für die GDP-HS notwendigen Dateien von Andreas mit diversen Änderungen/anpassungen.

5.3.4 Programmierung

Als Entwicklungs-/Programmierungsumgebung wurde die Lattice Diamond 2.0(64-bit) IDE eingesetzt. Ein Programmierkabel für Lattice-FPGA/CPLD Bausteine ist z.B. hier beschrieben:

http://www.fpga.com.cn/lattice/lattice_cable.pdf

5.3.5 Register der GDPHS / Änderungen zum Original

5.3.5.1 Zusammenfassung der GDPHS Register

Entnommen aus der Dokumentation von Andreas GDPHS64NEUv2

5.3.5.1.1 SPI

\$FFFFFF00 - SPI-Control-Register

Schreiben

7 6 5 4 3 2 1 0

| | | | | | | |

| | | | | ----- Clockdivider

| | | | | 000 = 40MHz/2 => 20,0MHz

| | | | | 111 = 40MHz/16 => 2,5MHz

| | | | | ----- frei

| | | | | ----- frei

| | | | | ----- Slave Select

| 01 = Slave 0, 10 = Slave 1

----- SPI-Controller enable

Lesen

7 6 5 4 3 2 1 0

| | | | | | | |

| | | | | | | --- IDLE 1 = Controller bereit für Daten

| | | | | | ----- Write Collision 1 = Datenverlust

----- frei

\$FFFFFF01 - SPI-Daten-Register

5.3.5.1.2 SOUND

\$FFFFFF50 - \$FFFFFF51

Funktion wie SOUND-Baugruppe

5.3.5.1.3 GDP

\$FFFFFF60 - Seitenumschaltung / XOR / DMA

Schreiben

7 6 5 4 3 2 1 0

| | | | | | | |

| | | | | | | --- XOR-MODE (1 = aktiv)

| | | | ----- frei

| | ----- Leseseite (0 - 3)

----- Schreibseite (0 - 3)

Lesen

Ergebnis der DMA-Abfrage (nach Befehl \$0f an GDP)

Das Ergebnis ist invertiert!

Die Pixeladresse wird in den X- und Y-Registern übergeben

Bei der S/W-Variante werden 8 Pixel geliefert,

bei der Farb-Variante 2

\$FFFFFF61 - Scrollport

Schreiben

Hardware-Scrollwert in 2er Schritten (LSB wird ignoriert)

GDP-Register

\$FFFFFF70 - Status / CMD

\$FFFFFF71 - CTRL 1

```

$FFFFFF72 - CTRL 2
7 6 5 4 3 2 1 0
| | | | | | | |
| | | | | | ----- Art der Vektoren
| | | | | ----- 1 = Buchstaben kursiv
| | | | ----- 1 = Buchstaben vertikal
| | | ----- 1 = USER-Zeichensatz
| | ----- 1 = Transparent-Mode
| ----- 1 = Hardware-Cursor enable
----- frei

```

USER-Zeichensatz

Der USER-Zeichensatz befindet sich im RAM des FPGA, dieses ist mit dem deutschen NKC-Zeichensatz vorbelegt. Es stehen 96 Zeichen zur Verfügung. Diese beginnen bei dem Zeichen \$20 entsprechend SPACE und enden mit dem Zeichen \$7F.

Schreiben eines neuen Zeichens:

Setzen des Bits 4 in CTRL2

Laden der 1. Zeichenadresse in die X-Register (\$FFFFFF78 + 79)

Diese Adresse errechnet sich wie folgt:

$(NR - \$20) * 5$

NR = ASCII-Nummer des Zeichens (z.B. \$41 fürs 'A')

3. Schreiben von 5 Byte mit dem Muster des Zeichen auf Port \$FFFFFF7E

Das Muster des Zeichens ist vom Aufbau wie bei dem Befehl PROGZGE.

Dies erfolgt ohne erneutes Schreiben des X-Registers (autoincrement).

z.B. das Omega: `dc.b %10011110`

`dc.b %11100001`

`dc.b %00000001`

`dc.b %11100001`

`dc.b %10011110`

Transparent-Mode

Wie bei der original GDP werden beim Schreiben eines Buchstaben nur die Pixel mit der Vordergrundfarbe neu geschrieben. Daher bekommt man "Buchstabensalat", wenn zwei oder mehr Zeichen auf der selben Position geschrieben werden.

Anders wenn man das Transparent-Mode-Bit setzt, dann werden auch die Pixel mit Hintergrundfarbe neu geschrieben. Man kann dann also auf ein Löschen verzichten, bevor ein neues Zeichen geschrieben wird.

\$FFFFFF73 - CSIZE
\$FFFFFF74 - DeltaX MSB (1 Bit) neu ab Version vom 02.05.09
\$FFFFFF75 - DeltaX LSBs (8 Bit)
\$FFFFFF76 - DeltaY MSB (1 Bit) neu ab Version vom 02.05.09
\$FFFFFF77 - DeltaY LSBs (8 Bit)
\$FFFFFF78 - X MSBs (4 Bit)
\$FFFFFF79 - X LSBs (8 Bit)
\$FFFFFF7A - Y MSBs (4 Bit)
\$FFFFFF7B - Y LSBs (8 Bit)

\$FFFFFF7C - XLP / Reserve
\$FFFFFF7D - YLP / Reserve

\$FFFFFF7E - Schreibport für USER-Zeichensatz
siehe CTRL 2

\$FFFFFF7F - Reserve

GDP Farberweiterung

\$FFFFFFA0 - Vordergrundfarbe
Farbnummer der Vordergrundfarbe
siehe CLUT

\$FFFFFFA1 - Hintergrundfarbe
Farbnummer der Hintergrundfarbe
siehe CLUT

\$FFFFFFA4 - CLUT Farbnummer

\$FFFFFFA5 - CLUT Daten MSB (1 Bit)

\$FFFFFFA6 - CLUT Daten LSBs (8 Bit)

Die CLUT (ColorLookUpTable) ist eine Farbtabelle mit 16 9-Bit-Einträgen.

Dies bietet die Möglichkeit gleichzeitig 16 der 512 möglichen Farben

darzustellen. Die darzustellende Farbe wird durch ihre Nummer in der

CLUT festgelegt (\$FFFFFFA0 + A1). In der unten stehenden Tabelle ist die

Standardbelegung aufgeführt.

Zum ändern einer Farbe muß zunächst die gewünschte Farbnummer in das Register \$FFFFFFA4 geschrieben werden, danach das Highbyte und dann das Lowbyte des Farbwertes in die Register \$FFFFFFA5 und A6.

Es können weitere Farbwerte, ohne erneutes schreiben des Registers \$FFFFFFA4 übertragen werden (autoincrement).

Standardbelegung der CLUT

Farbnummer		Name	Wert		Entsprechender		
dez	hex	NKC	I-Net	hex	binär	HTML-Code	
					R G B		
0	\$0	Schwarz	black	\$0000	000 000 000	#000000	
1	\$1	Weiß	white	\$01FF	111 111 111	#FFFFFF	
2	\$2	Gelb	yellow	\$01F8	111 111 000	#FFFF00	
3	\$3	Grün	lime	\$0038	000 111 000	#00FF00	
4	\$4	Rot	red	\$01C0	111 000 000	#FF0000	
5	\$5	Blau	blue	\$0007	000 000 111	#0000FF	
6	\$6	Violett	fuchsia	\$01C7	111 000 111	#FF00FF	
7	\$7	Zyan	aqua	\$003F	000 111 111	#00FFFF	
8	\$8	Dunkelgrau	gray	\$0092	010 010 010	#404040	
9	\$9	Hellgrau	silver	\$0124	100 100 100	#808080	
10	\$A	Dunkelgelb	olive	\$00D8	011 011 000	#606000	
11	\$B	Dunkelgrün	green	\$0018	000 011 000	#006000	
12	\$C	Dunkelrot	maroon	\$00C0	011 000 000	#600000	
13	\$D	Dunkelblau	navy	\$0003	000 000 011	#000060	
14	\$E	Violett dunkel	purple	\$00C3	011 000 011	#600060	
15	\$F	Zyan dunkel	teal	\$001B	000 011 011	#006060	

5.3.5.14 KEY

\$FFFFFF68 - \$FFFFFF69

Funktion wie Key-Karte

5.3.5.15 MAUS

\$FFFFFF88 - \$FFFFFF8F

Funktion wie HARDCOPY/MAUS-Baugruppe

5.3.5.16 TIMER

\$FFFFFFF4 - Control Register

7 6 5 4 3 2 1 0

| | | | | | | |

| | | | | | | --- Run (Timer run)

| | | | | ----- WRM (Timer Register Write Mode)

| | | | | 0=Write only reload register

| | | | | 1=Write only timer register

| | | | | 2=Write both

| | ----- frei

| ----- TOVF Timer Overflow Flag.

| Muß per Software zurückgesetzt werden

----- IE (Interrupt enable)

\$FFFFFFF5 - Counter/Reload Register High

\$FFFFFFF6 - Counter/Reload Register Low

Das Counter und Reload Register ist je 16 bit breit.

Der Timer zählt mit 1 MHz abwärts.

Wenn das High-byte beschrieben wird dann wird der Wert in einem Zwischenregister gespeichert.

Wird dann das Low-byte geschrieben wird dieser Wert zusammen mit dem zwischengespeicherten High-Byte in das Counter oder Reload (oder beide) übernommen (somit werden immer alle 16 bit auf einmal geschrieben).

Gelesen wird immer das Counter-Register.

5.3.5.2 Funktionsanpassungen im GDPHS Core

Entnommen aus der Dokumentation von Andreas GDPHS64NEUv2:

Es kann im Sourcecode zwischen Farbe / Monochrom (kompatibel) umgeschaltet werden durch umsetzen einer Konstanten im Package gdp_global-p.vhd
GDP64HSFPGA\vhdl\rtl\gdp_global-p.vhd

Zeile 22:

```
constant color_support_c : boolean :=true;
```

Wird diese Konstante auf false gesetzt und danach eine neue Synthese mit isPLever gestartet so wird ein .jed File für eine monochrome GDP erzeugt (nur ein Speicherchip bestückt). Alle anderen Features (Timer, SPI, ...) sind identisch

In gdp_lattice_top.vhd können Konstanten gesetzt werden, um Module zu (de)aktivieren:

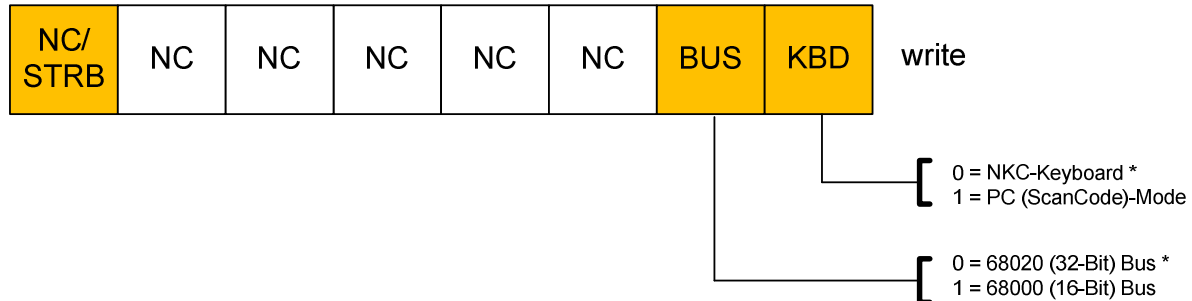
```
constant use_ser_key_c      : boolean := false;
constant use_ps2_key_c      : boolean := true;
constant use_ps2_mouse_c   : boolean := true;
constant use_ser1_c         : boolean := true;
constant use_sound_c        : boolean := true;
constant use_spi_c          : boolean := true;
constant use_timer_c        : boolean := true;
constant dipswitches_c     : std_logic_vector(7 downto 0) := X"49";
```

Mit folgenden Konstanten werden Speicherbereiche definiert:

```
constant GDP_BASE_ADDR_c      : std_ulogic_vector(7 downto 0) := X"70"; -- r/w
constant SFR_BASE_ADDR_c      : std_ulogic_vector(7 downto 0) := X"60"; -- w
constant COL_BASE_c           : std_ulogic_vector(7 downto 0) := X"A0"; -- r/w
constant KEY_BASE_ADDR_c      : std_ulogic_vector(7 downto 0) := X"68"; -- r
constant DIP_BASE_ADDR_c      : std_ulogic_vector(7 downto 0) := X"69"; -- r
constant MOUSE_BASE_ADDR_c    : std_ulogic_vector(7 downto 0) := X"88"; -- r/w
constant SER_BASE_ADDR_c      : std_ulogic_vector(7 downto 0) := X"F0"; -- r/w
constant SOUND_BASE_ADDR_c    : std_ulogic_vector(7 downto 0) := X"50"; -- r/w
constant SPI_BASE_ADDR_c      : std_ulogic_vector(7 downto 0) := X"00"; -- r/w
constant T1_BASE_ADDR_c       : std_ulogic_vector(7 downto 0) := X"F4"; -- r/w
```

5.3.5.3 Zusätzliche Register/Funktionalitäten

0xFFFFF67 = keyboard/cpu mode register



Das KBD/CPU-Mode-Register ist innerhalb des GDPHS Cores (PS2Keyboard.vhd) realisiert und hat zur Zeit 3 Funktionen:

Bit 0 bestimmt, in welchem Modus die Tastatur arbeite. Per default (=0) wird die Tastatur im NKC-Mode abgefragt. Wenn Bit 0 auf 1 gesetzt wird, liefert das KBD-Register 0x69 den Scancode der gedrückten Taste. Da der Scan-Code mit 8 Bit übertragen wird, wird das Strobe Signal (Zeichen im Buffer) in Bit 7 des Mode-Registers übertragen. Das Bit ist so lange gesetzt wie Zeichen im Buffer vorhanden sind und muß nicht von der Software zurückgesetzt werden.

Bit 1 bestimmt das interne Signal is68000, welches die Adressdekodierung des ECB-Bus Interface anpaßt. Thematisch gehört diese Funktionalität zwar in das Modul `nkc_bw_wrapper`, aber dann müßten mehr Register und Signale im FPGA realisiert werden.

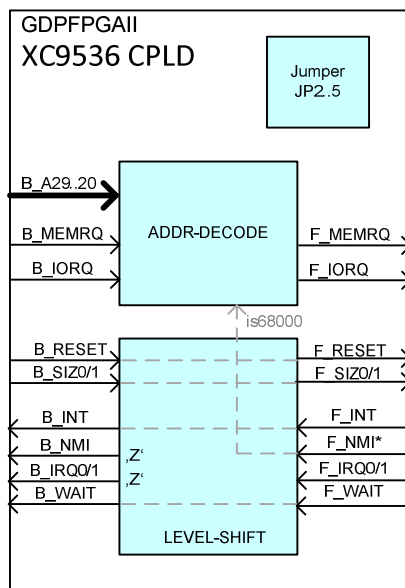
CTRL2.6:

Mit Bit6 im CTRL2 Register der GDP kann ein Hardware-Cursor eingeschaltet werden, der in der aktuellen Vordergrundfarbe blinkt.

5.4 Das CPLD

Im Folgenden wird im Wesentlichen das CPLD in der NKC Version beschrieben. Prinzipiell kann das Verhalten der Baugruppe auch an andere Anwendungsfälle angepaßt werden, z.B. können das Interrupt Handling verfeinert und die dekodierten Adressbereiche konfiguriert werden.

5.4.1 Blockschaltbild



5.4.2 Funktionsbeschreibung

Das CPLD auf der Platine dient der Adress(Vor)Dekodierung und Pegelanpassung zwischen 5V ECB Bus und 3.3V Logik. Es werden neben den Adressleitungen und dazugehörigen MEMRQ/IORQ Signalen die Signale für Interrupt- und WAIT- Steuerung generiert.

In der Default-Konfiguration werden die IRQ und NMI Leitungen nicht verwendet und hochohmig geschaltet. Das WAIT Signal wird vom FPGA durchgereicht bzw. bei F_WAIT=H wird der Pin am Bus hochohmig. Das Gleiche gilt für F_INT und B_INT.

Das Signal F_NMI wird z.Z. „fremdverwendet“ um das Signal is68000 vom FPGA an das CPLD zu routen. Damit kann im CPLD der zu dekodierende Adressbereich an den 68000 angepaßt werden. Soll das Signal NMI als solches verwendet werden, muß das CPLD entsprechend angepaßt und die Pins für A24..A29 auf der CPU68K00 auf ,0‘ gezogen werden.

5.4.3 Programmierung

Zum Programmieren des CPLDs benötigt man das ISE-Webpack von Xilinx, dieses kann man (nach kostenloser Registrierung) von der Webseite www.xilinx.com herunterladen. Weiterhin benötigt man einen Programmieradapter, hierfür kann z.B. die Schaltung von www.holger-klabunde.de verwendet. Der Programmierstecker auf der GIDE Platine hat die Belegung dieses Programmers.

Das eingesetzte CPLD wurde mit der Xilinx ISE Design Suite 13.4 erstellt.

Das zugehörige Projekt befindet sich im Verzeichnis FPGA\src\Xilinx\GDPFPGAII-XILINX (GDPFPGAII-XILINX.xise)

6 Aufbauanleitung

Schaltplan, Layout, Stückliste und Bestückungspläne sind im Anhang bzw. in den PCB-Daten zu finden.

6.1 Umgang mit IC's

CMOS-Bausteine sind hochempfindlich gegen elektrostatische Aufladung! Bewahren oder Transportieren Sie CMOS-Bausteine nur auf leitenden Schaumstoff! Alle Pins müssen kurzgeschlossen sein. Achten Sie darauf, daß Sie Verbindung mit einer Erdungsmöglichkeit haben, bevor Sie mit diesen Bausteinen arbeiten. Geeignete ESD-Artikel gibt es im Fachhandel.

6.2 Aufbau Schritt für Schritt

Beim Layout der Platine wurde Wert auf eine möglichst hohe Nachbausicherheit gelegt und z.B. Pads der SMD Bausteine für das Löten mit Lötkolben/Lötkelch angepaßt. Dennoch muß ausdrücklich darauf hingewiesen werden, daß das Einlöten gerade des FPGA mit seinem Pin-Abstand von 0.2mm nur mit viel Erfahrung gelingen wird. Es müssen auf jeden Fall spezielle Lötvorrichtungen (Lötkelch) und ein Mikroskop (zur Überprüfung der Lötstellen und evtl. Nachlöten mit feinsten Lötspitze) vorhanden sein.

7 Inbetriebnahme

7.1 Betrieb mit CPU68K20 (M68020)

Ist die Karte fehlerfrei aufgebaut und das CPLD sowie das FPGA mit den entsprechenden JEDEC Dateien programmiert, kann sie mit der CPU68K20 direkt in Betrieb genommen werden.

Um JADOS direkt von der SD-Card zu booten wird am besten das von Jens bereitgestellte Image auf die SD-Card geschrieben.

Als (G)IDE wird z.B. eine ScanDisk Ultrall 2.0GN eingesetzt. Man kann auch andere CF Karten einsetzen, es muß aber darauf geachtet werden, daß der ATA Modus unterstützt wird.

Der Zugriff (lesen/schreiben) auf die GIDE wurde mit den MTOOLS 08F bis 25MHz CPU-Takt fehlerfrei getestet. Auf der CPU68K20 wurde das GP in der Version 710R5 eingesetzt.

Da der 68020 ein dynamisches Bus-Sizing hat, muß das CPLD auf der CPU68K20 für den Betrieb mit dem 16Bit-Bus der GDPFPGAII angepaßt werden. Dazu muß das JEDEC File im Verzeichnis FPGA\jed\IC-68020 - 150202.vhd auf der CPU68K20 eingesetzt werden. Die geänderte Quelldatei für den 68020 ist in FPGA\jed\IC.vhd, das komplette Projekt ist in der Beschreibung des Projektes für den 68020 enthalten.

7.2 Betrieb mit CPU68K00 (M68000)

Im Prinzip gilt das Gleiche wie für die CPU68K20: Karte muß fehlerfrei aufgebaut und das CPLD und FPGA programmiert sein. Dann kann die Karte sofort in Betrieb genommen werden.

Im Register 0x67 (Bit 1 =1 =>ist 68000) kann die Adressdekodierung für den 68000 ausgewählt werden (aus gdphs Modul ins Wrapper Modul geroutet).

Alternativ können die Adressleitungen A24...29 auf der CPU Karte nach L gezogen werden (Pull-Down) damit die Adressen sauber dekodiert werden.

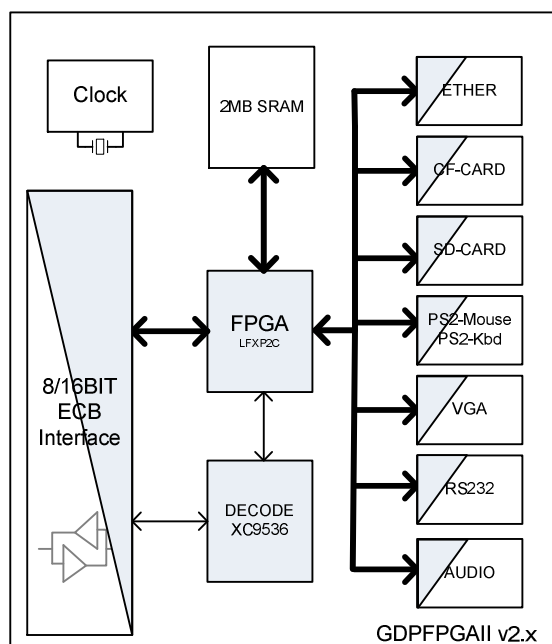
Im CPLD der CPU68K00 selbst muß nichts geändert werden, alle IO und ExtMEM Zugriffe sind bereits dekodiert. Bei direktem Zugriff auf dem Bildschirmspeicher (SRAM) muß das Businterface im FPGA die Steuerleitungen des 68000 korrekt interpretieren. Dazu muß im Keyboard/CPU-Mode Register 0x67 das Bit 1 auf 1 gesetzt werden.

8 Version mit LFXP2C

Diese Hardwareversion ist bisher nicht getestet. Sie unterscheidet sich von der LFXP6C durch folgende zusätzliche Features:

- FPGA: **LFXP2C**-8E-5QN208C
- **2MB** SRAM (8/10ns – z.B. CY7C1051DV33 oder IS61WV51216)
- **Ethernet** (ENC28J60)

8.1 Blockschaltbild



9 Fehlerkorrekturen

In den vorliegenden Versionen HW:1.3/FPGA: 1.0 (LFXP6C-3QN208C sind keine Fehler bekannt.

Ein reproduzierbarer Fehler tritt beim Kopieren von GIDE nach JADOS auf (GP:710R5 und MTOOLS 08F): ist die von der GIDE zu kopierende Date bereits auf dem JADOS Laufwerk vorhanden und etwas kleiner als die von der GIDE zu kopierende, wird die Datei zwar überschrieben, aber auf die Länge der JADOS Datei beschnitten. Workaround ist z.Z. die Datei auf den JADOS Laufwerk vorher zu löschen. Das Problem liegt wahrscheinlich in JADOS.

Das Laden der MTOOLS 09A wird mit „Division durch 0“ quittiert. Es handelt sich dabei um einen Software-Fehler im GP bis Version 7.10r5 im Modul ideio.asm, der bereits lokalisiert wurde und in der nächsten Revision beseitigt wird

Mit GP 1.10R5 muß mit den MTOOLS 08F und dem beschriebenen „Workaround“ gearbeitet werden.

In den MTOOLS sollte der Delay-Wert für das Warten auf BSY, DRQ und RDY auf einen höheren Wert eingestellt werden (0x100000), da beim Schreiben auf CF-CARD eine grössere Latenz-Zeit einkalkuliert werden muss.

10 Quellenhinweise

Projekt GDPHS64 von Andreas Voggenender

Handbuch Grundprogramm V 7.0 Jens Mewes 2007

Handbuch Jados 3.50 Klaus Janßen 1985-1990

<http://www.schuetz.thtec.org/index.html>

<http://www.drcrazy.de/nkc>

11 Anhänge

11.1 Steckerbelegungen

JP15 – Xilinx (IC6) programming connector

TMS	TDI	TDO	TCK	GND	VCC
-----	-----	-----	-----	-----	-----

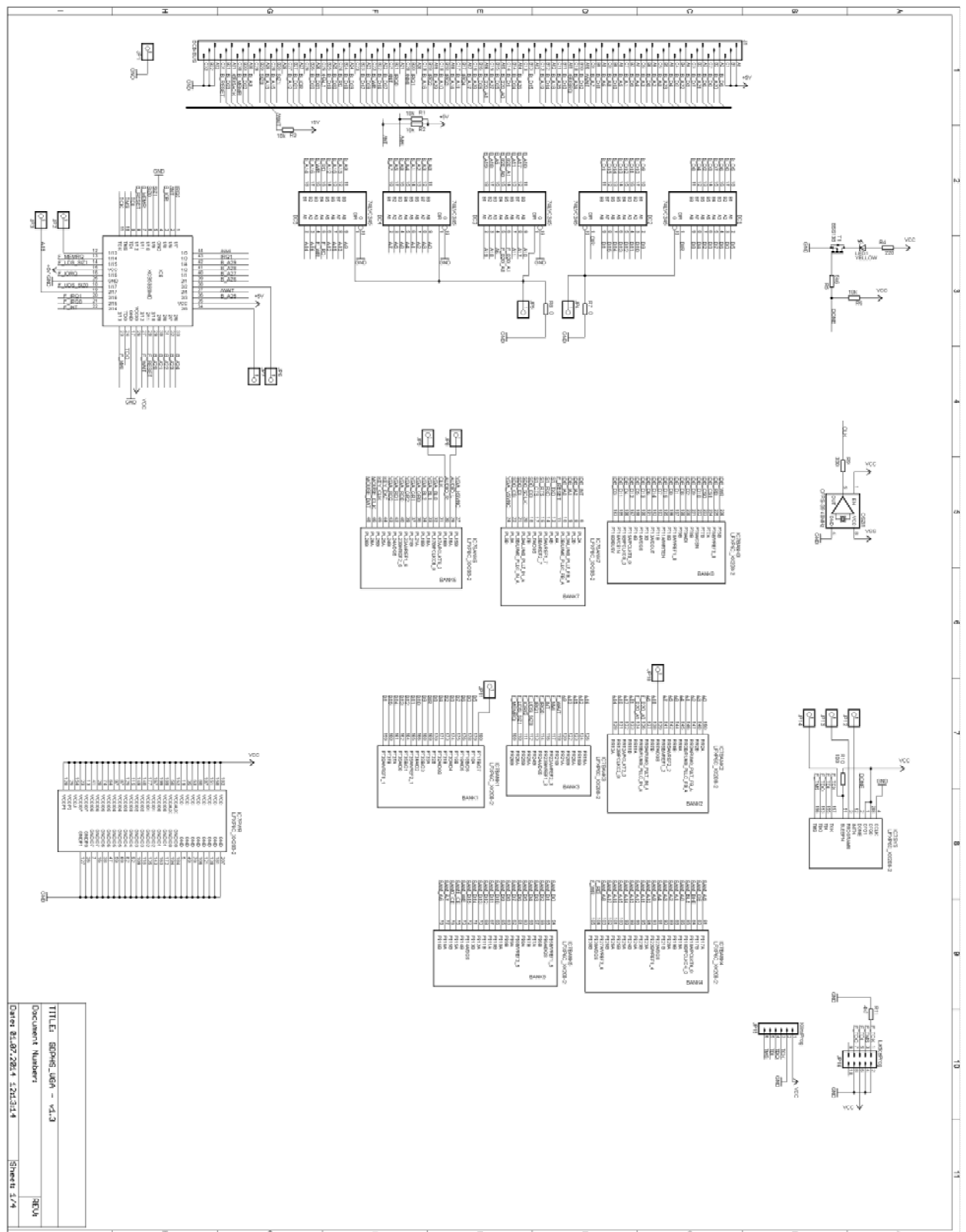
JP16 – Lattice (IC7) programming connector

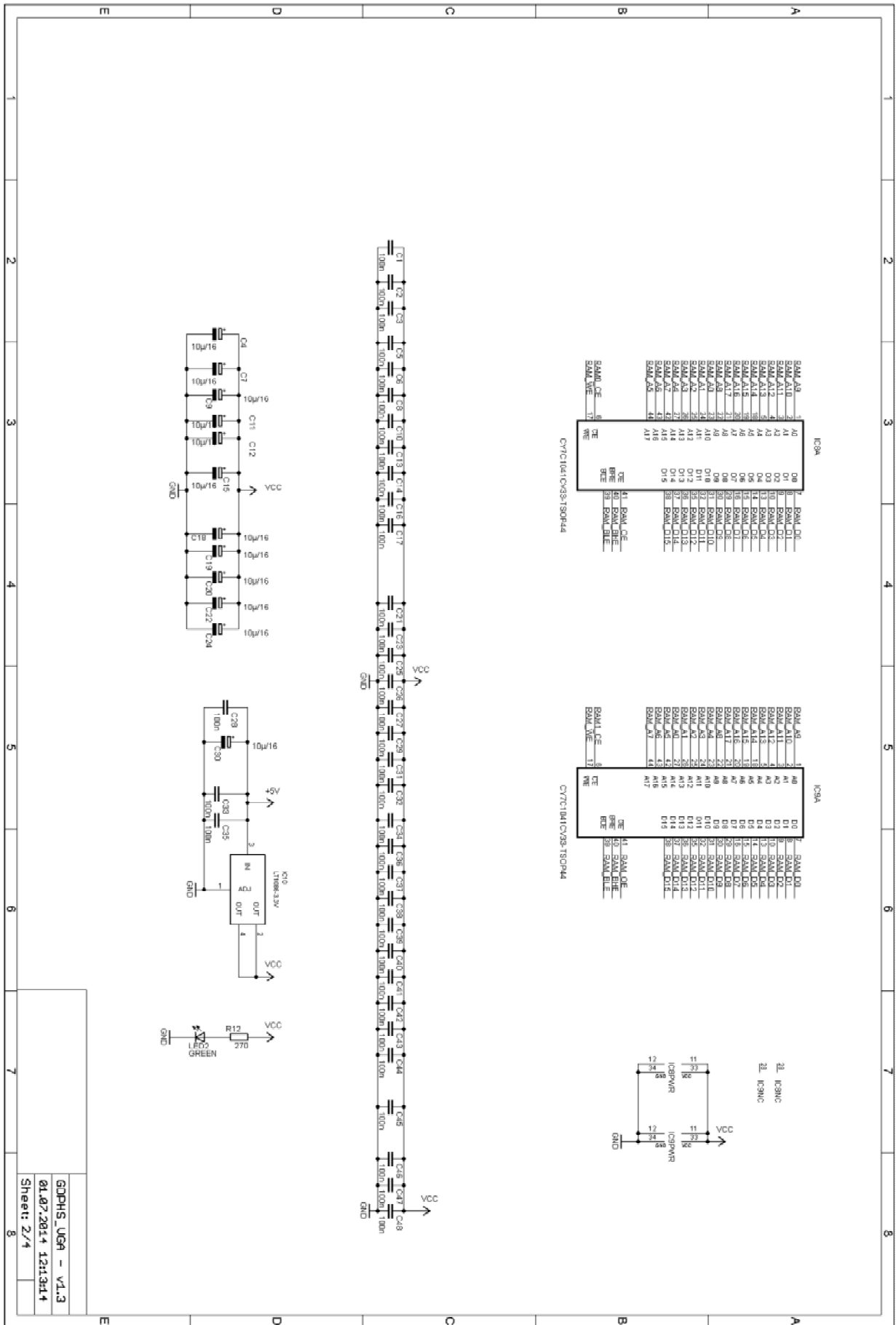
nc	TDO	TDI	TMS	TCK
nc	GND	VCC	GND	GND

JP20 – RS232 Serial Port connector

RXD			CTS	
	TXT	GND	RTS	

11.2 Schaltplan (LFXP6)





GOPHS_UGA - v1.3
01.07.2014 12:13:14
Sheet: 2/4

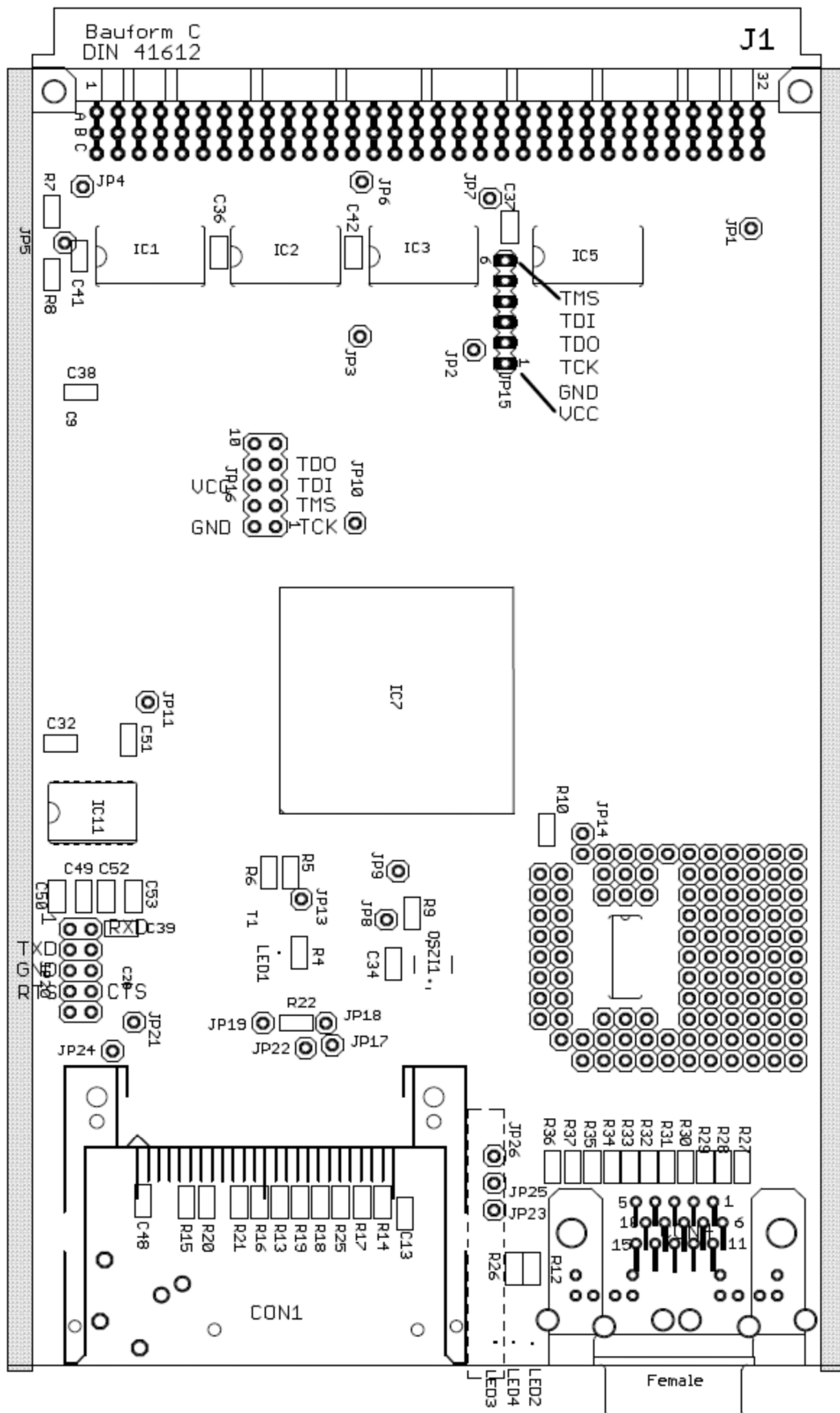
11.3 Stückliste (LFXP6)

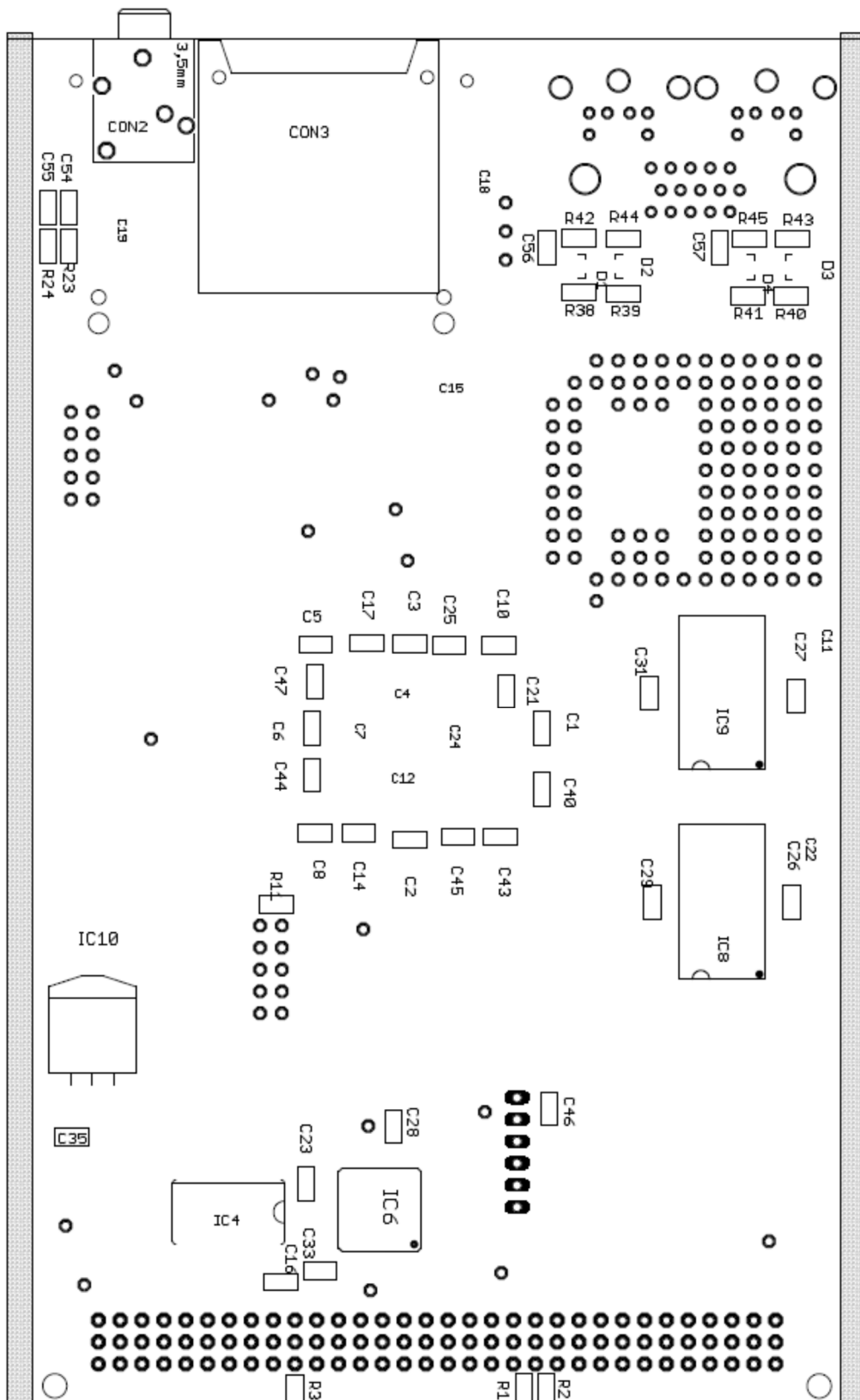
Qty	Value	Parts
4	0	R7, R8, R19, R22
1	100	R10
43	100n	C1, C2, C3, C5, C6, C8, C10, C13, C14, C16, C17, C21, C23, C25, C26, C27, C28, C29, C31, C32, C33, C34, C35, C36, C37, C38, C39, C40, C41, C42, C43, C44, C45, C46, C47, C48, C49, C50, C51, C52, C53, C56, C57
12	10k	R1, R2, R3, R6, R13, R14, R15, R16, R17, R18, R20, R21
12	10µ/16	C4, C7, C9, C11, C12, C15, C18, C19, C20, C22, C24, C30
4	120	R38, R39, R40, R41
1	1K	R28
1	1K8	R29
2	1k	R31, R34
2	1k8	R32, R35
1	220	R4
3	270	R12, R25, R26
4	2k	R42, R43, R44, R45
1	330	R9
2	3k3	R23, R24
2	47	R36, R37
3	470	R27, R30, R33
1	4k7	R11
2	4n7	C54, C55
1	5k6	R5
1	74123D	IC12
5	74LVC245	IC1, IC2, IC3, IC4, IC5
4	BAT54S	D1, D2, D3, D4
1	BSS138	T1
1	CFPS-39 40MHz	OSZI1
2	CY7C1041CV33-TSOP44	IC8, IC9
1	ECB-BUS	J1
1	FPS-009-2405-0	CON3
1	GREEN	LED2
1	IDE	CON1
1	LFXP6C_XX208-2	IC7
1	LT1086-3.3V	IC10
1	LatticeProg	JP16
1	MAX232ECWE	IC11
1	MDSSV-BD06-HF15B	CON4
1	PG203J	CON2

1	RED	LED3
1	RS232	JP20
1	XC9536SMD	IC6
1	XilinxProg	JP15
2	YELLOW	LED1, LED4

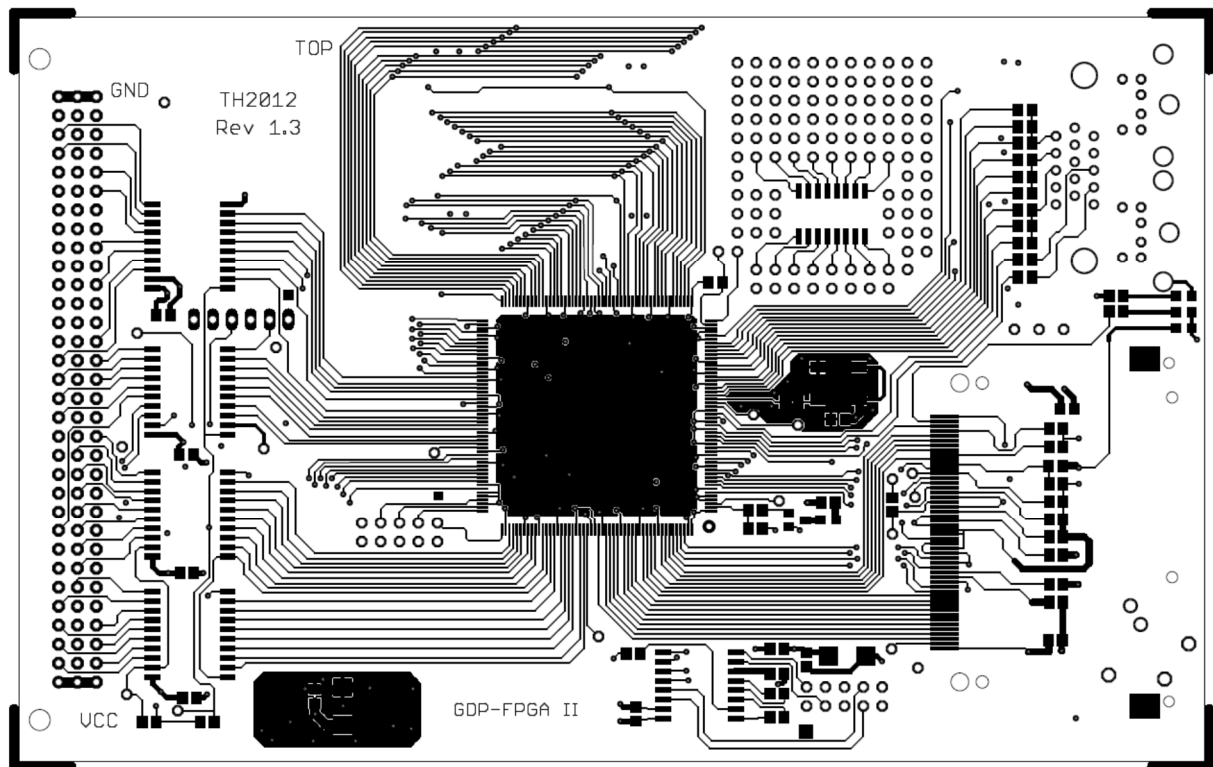
Eine aktuelle Stückliste mit Lieferanteninformationen ist in GDPHS_VGA - v2.2 - BOM.xlsx zu finden

11.4 Bestückungsplan (LFXP6)

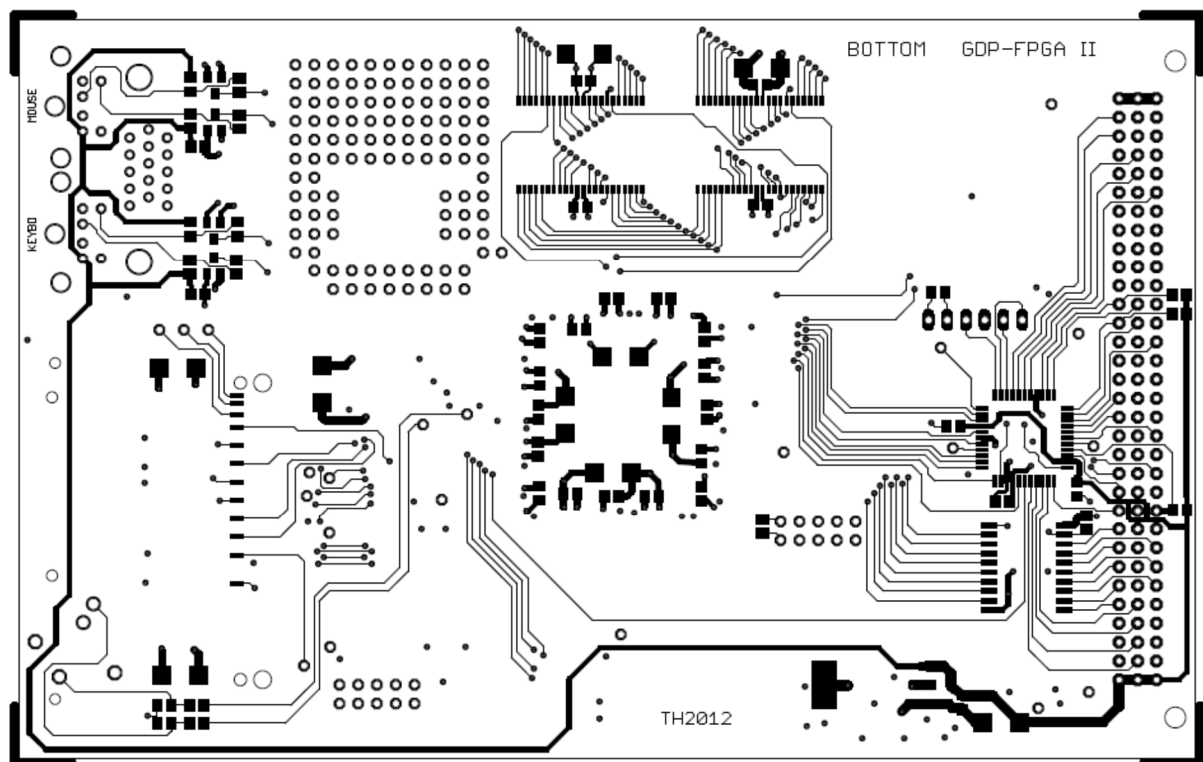




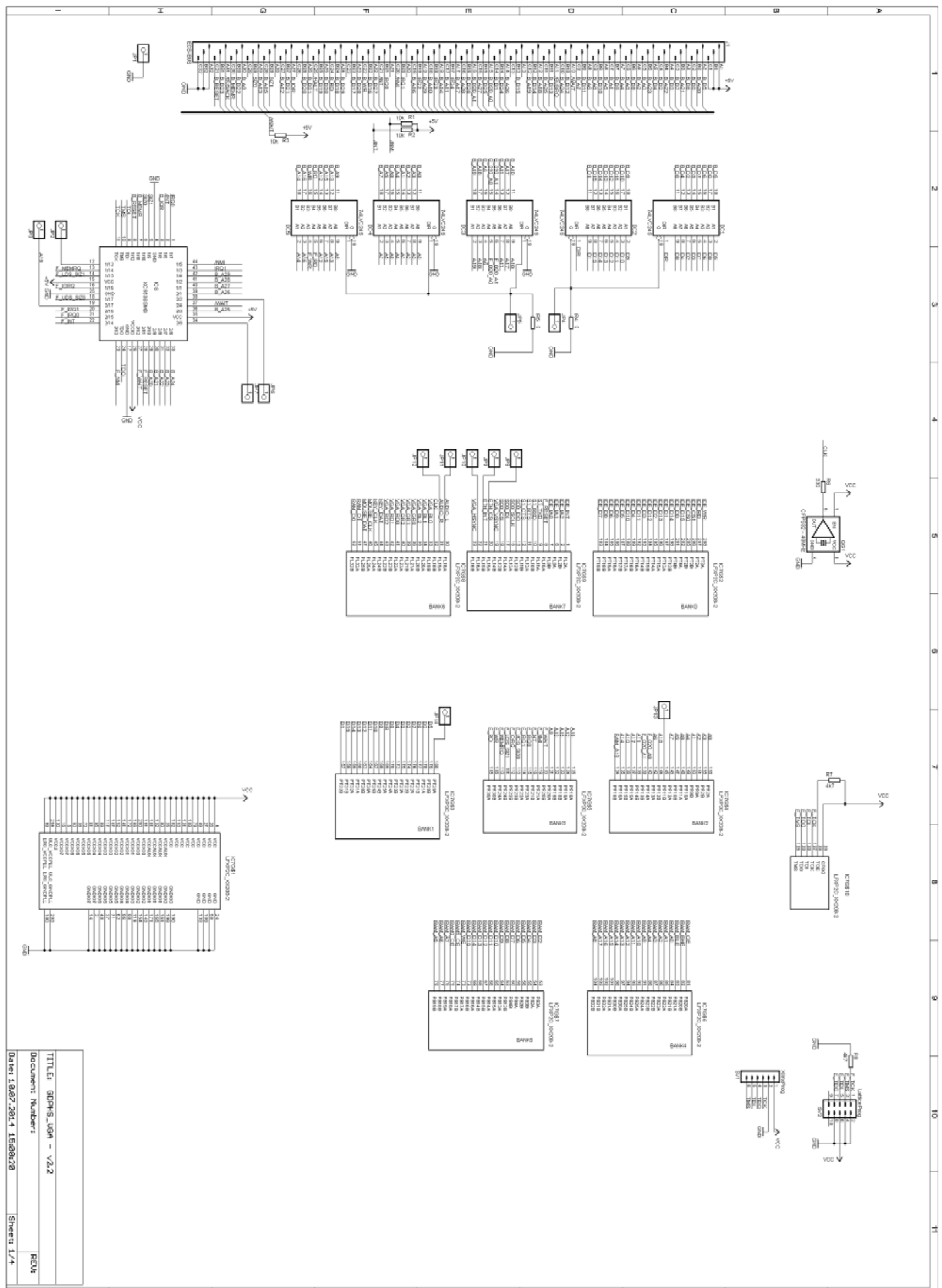
11.5 Layout Oben (LFXP6)

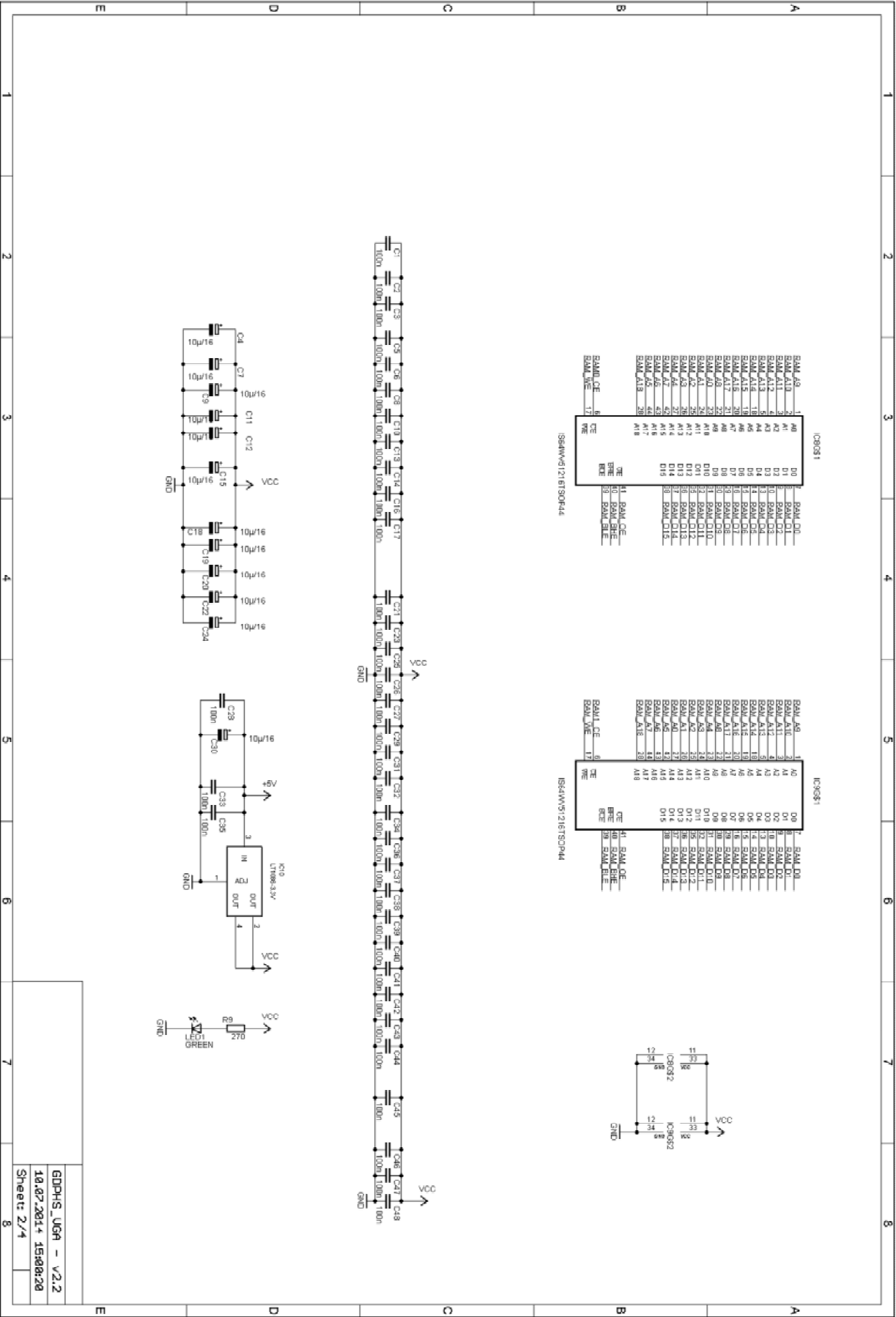


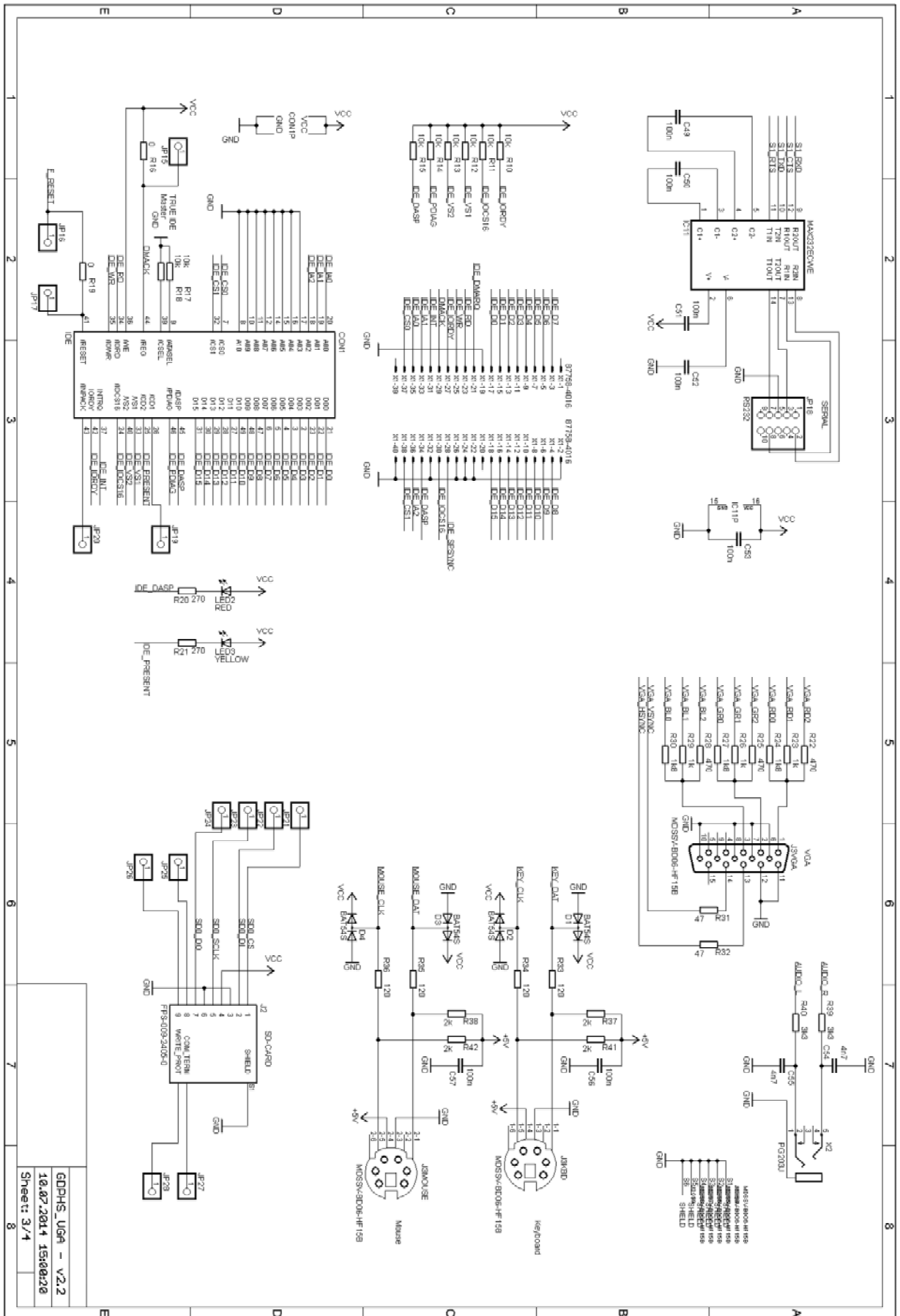
11.6 Layout Unten



11.7 Schaltplan (LFXP2C)





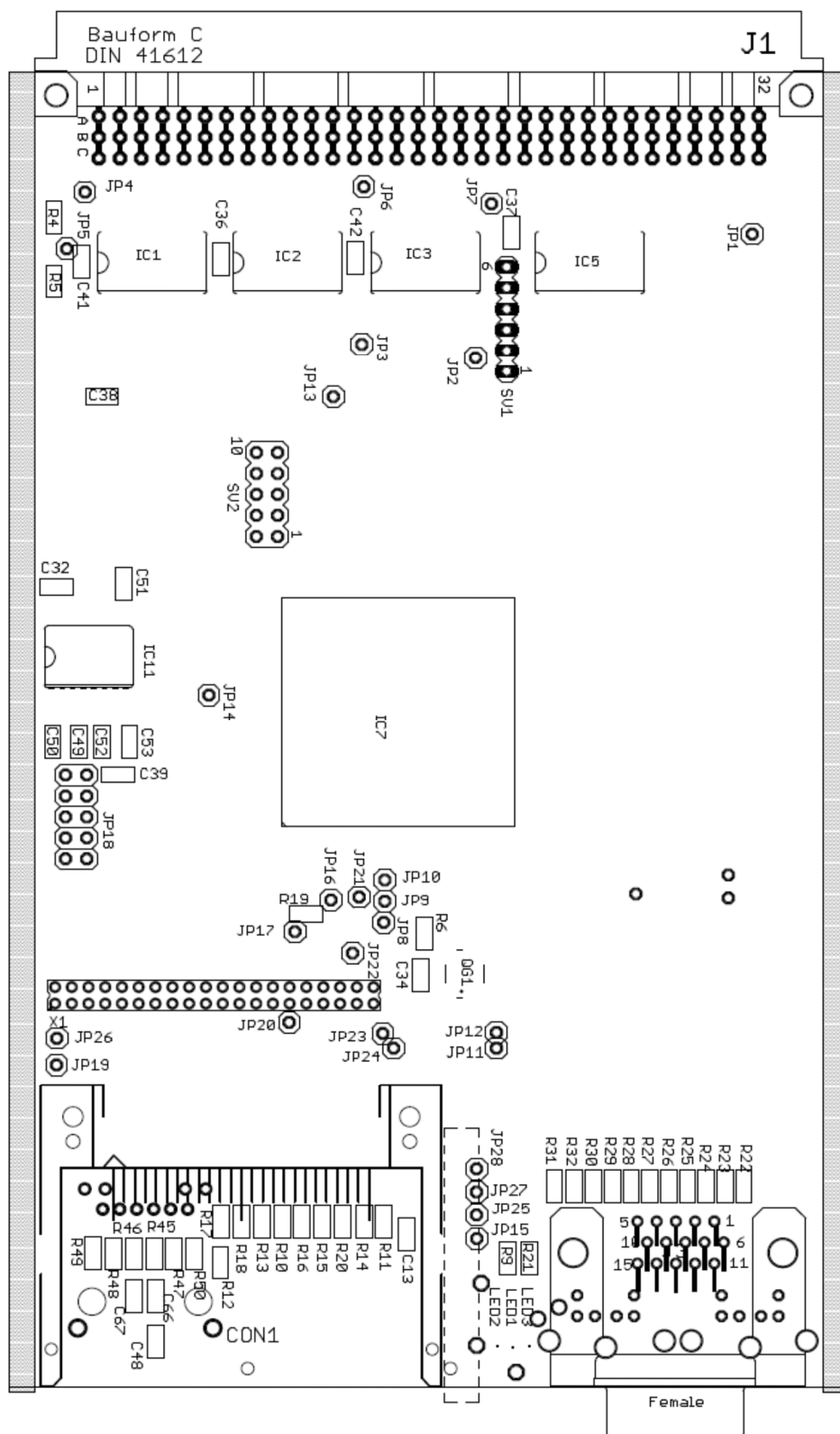


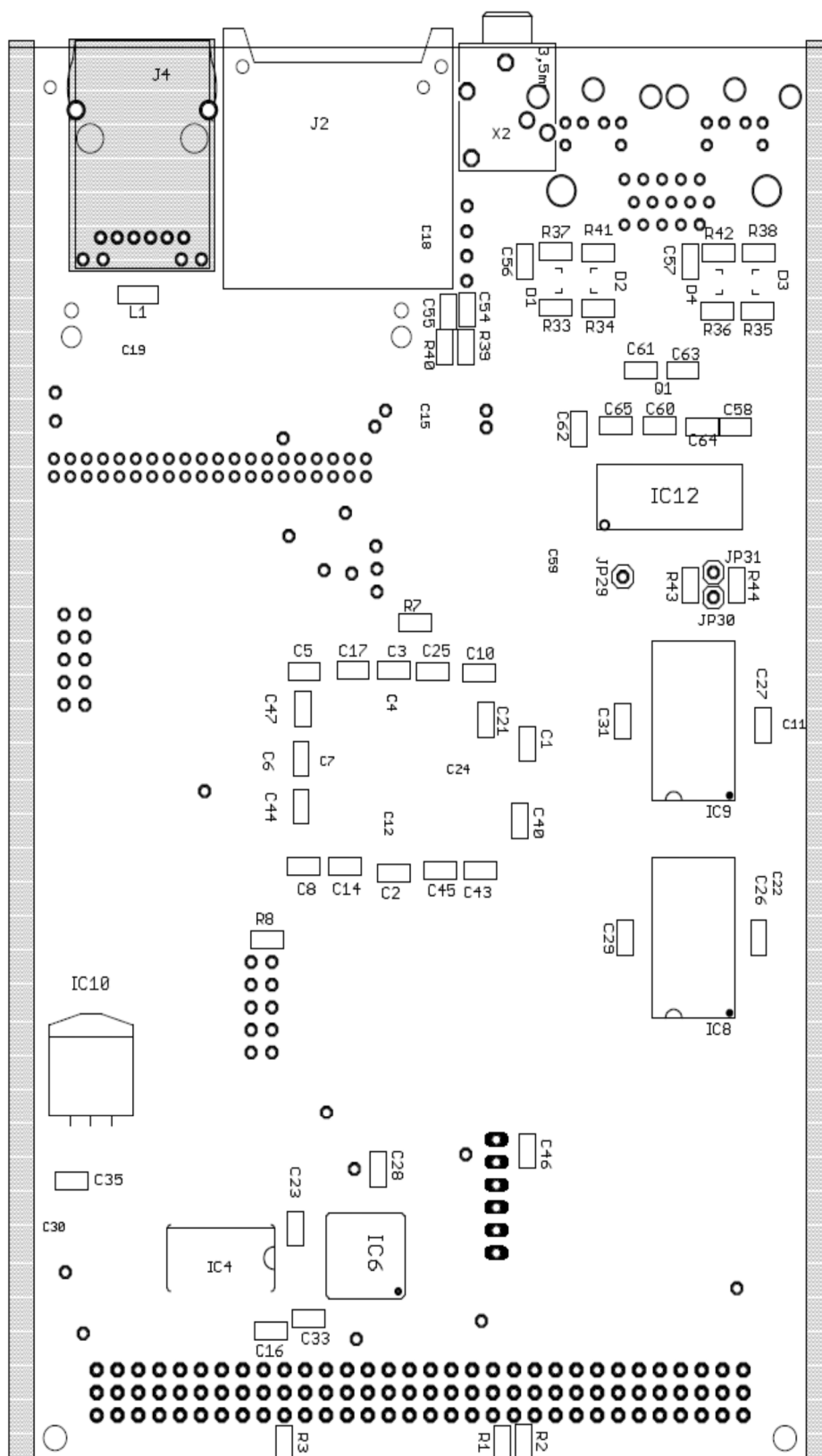
11.8 Stückliste (LFXP2C)

Qty	Value	Parts
1		J4
1		L1
30		JP1, JP2, JP3, JP4, JP5, JP6, JP7, JP8, JP9, JP10, JP11, JP12, JP13, JP14, JP15, JP16, JP17, JP19, JP20, JP21, JP22, JP23, JP24, JP25, JP26, JP27, JP28, JP29, JP30, JP31
5	0	R4, R5, R16, R19, R43
50	100n	C1, C2, C3, C5, C6, C8, C10, C13, C14, C16, C17, C21, C23, C25, C26, C27, C28, C29, C31, C32, C33, C34, C35, C36, C37, C38, C39, C40, C41, C42, C43, C44, C45, C46, C47, C48, C49, C50, C51, C52, C53, C56, C57, C58, C60, C62, C64, C65, C66, C67
11	10k	R1, R2, R3, R10, R11, R12, R13, R14, R15, R17, R18
13	10µ/16	C4, C7, C9, C11, C12, C15, C18, C19, C20, C22, C24, C30, C59
4	120	R33, R34, R35, R36
3	1k	R23, R26, R29
3	1k8	R24, R27, R30
2	22p	C61, C63
5	270	R9, R20, R21, R49, R50
4	2k	R37, R38, R41, R42
1	2k7	R44
1	330	R6
2	3k3	R39, R40
2	47	R31, R32
3	470	R22, R25, R28
2	4k7	R7, R8
2	4n7	C54, C55
4	50	R45, R46, R47, R48
5	74LVC245	IC1, IC2, IC3, IC4, IC5
1	87758-4016	X1
4	BAT54S	D1, D2, D3, D4
1	CFPS32 - 40MHz	QG1
1	CSM-7X-DU 25MHz	Q1
1	ECB-BUS	J1
1	ENC28J60SOIC28W	IC12
1	FPS-009-2405-0	J2
1	GREEN	LED1
1	IDE	CON1
2	IS64WV51216TSOP44	IC8, IC9

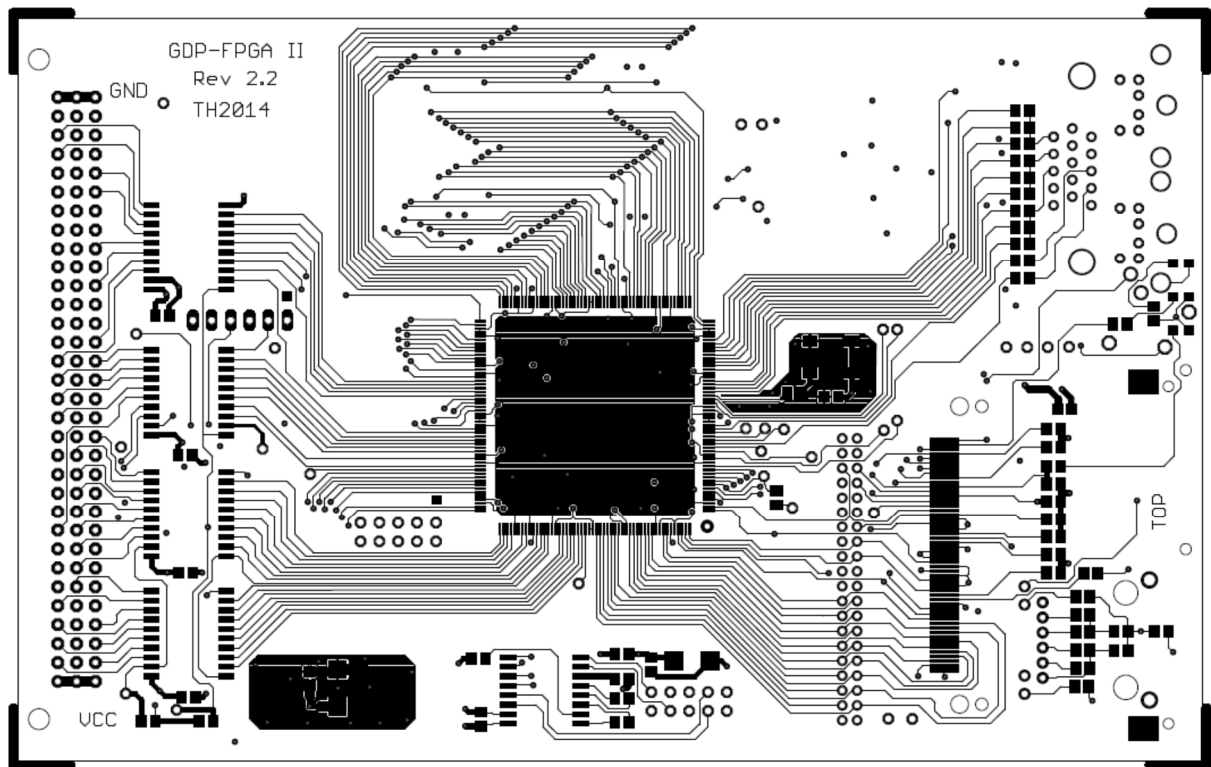
1	LFXP2C_XX208-2	IC7
1	LT1086-3.3V	IC10
1	LatticeProg	SV2
1	MAX232ECWE	IC11
1	MDSSV-BD06-HF15B	J3
1	PG203J	X2
1	RED	LED2
1	RS232	JP18
1	XC9536SMD	IC6
1	XilinxProg	SV1
1	YELLOW	LED3

11.9 Bestückungsplan (LFXP2C)





11.10 Layout Oben (LFXP2C)



11.11 Layout Unten (LFXP2C)

