Inhaltsverzeichnis

Buildroot

Für Buildroot werden folgende Pakte benötigt:

which sed make binutils build-essentials gcc g++ bash patch gzip bzip2 perl tar cpio python unzip rsync file bc wget ncurses5

Als erstes muss Buildroot heruntergeladen werden:

https://buildroot.org/downloads/

Wählt dazu den aktuellsten Tarball aus. Entpackt den Tarball in euer Workspace und betretet das neue Verzeichnis buildroot. Anschließend muss das Unterverzeichnis configs betreten werden.

In diesem Unterverzeichnis ladet ihr die vorgefertigte config Datei für den Blueberry PI herunter:

wget https://github.com/petit-miner/Blueberry-PI/raw/master/Software/blueberrypi_defconfig

Verlasst danach das Unterverzeichnis configs und wechselt ins Hauptverzeichnis von Buildroot, nun wird die config Datei angewendet:

make blueberrypi_defconfig

Möchte man nun Buildroot weiter konfigurieren, so muss dieser Befehl verwendet werden:

make menuconfig

Der Buildprozess lässt sich mit

make

starten. Dabei ist zu beachten, dass der Parameter -j nicht verwendet werden darf, da Buildroot kein paralleles Arbeiten unterstützt!

Nach erfolgreichem erstellen des Buildroots ist der Tarball unter: ***output/images/rootfs.tar***

zu finden.

Imagepacker

Im Imagepacker befindet sich bereits eine kompilierte Version von u-boot und des Linux Kernels (4.21.rc-1). Somit kann, wenn gewünscht, der Buildprozess von u-boot und dem Linux Kernel übersprungen werden. Das Buildroot muss aufgrund seiner Größe allerdings noch selbst erstellt werden.

Möchte man ein LCD verwenden müssen diese Dateien in das /boot Verzeichnis kopiert werden:

boot-lcd.scr

sun8i-v3s-blueberrypi.dtb

Möchte man den VGA Ausgang benutzen müssen folgende Dateien in das /boot Verzeichnis kopiert werden:

boot-vga.scr

sun8i-v3s-blueberrypi.dtb

Möchte man keine Videoausgabe oder möchte eine parallele Kamera anschließen sind folgende Dateien zu kopieren:

boot.scr

sun8i-v3s-blueberrypi-csi-ov7670.dtb oder ***sun8i-v3s-blueberrypi-csi-ov2640.dtb***

(Anschließend muss die Devicetree Datei noch in sun8i-v3s-blueberrypi.dtb umbenannt werden!)

Nachdem die korrekten Bootdateien kopiert worden sind, muss das zuvor erstellte tarball Buildroot in den Ordner ***rootfs*** entpackt werden. Wichtig ist dabei, dass dieses als Root geschieht, da sonst die Dateirechte verloren gehen.

mkdir rootfs

sudo tar xvf rootfs/rootfs.tar

rm rootfs/rootfs.tar

Als letzter Schritt wird nun das flashbare Image erstellt, dies geschieht so:

sudo chmod +x imagepacker.sh

sudo ./imagepacker.sh boot/ rootfs/

Nach erfolgreichem Erstellen des Images ist dieses als .dd Datei zu finden.

u-boot

Als erstes muss der Sourcecode für u-boot von Github geklont werden:

git clone https://github.com/petit-miner/u-boot.git -b v3s-current

Betrete anschließend das neue Verzeichnis u-boot und wende die Konfigurationsdatei an:

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- BlueberryPI-vga_defconfig

Möchte man nun u-boot weiter konfigurieren, so muss dieser Befehl verwendet werden:

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

Der Buildprozess lässt sich mit

make

starten.

Nach erfolgreichem Erstellen von u-boot ist dieses als ***u-boot-sunxi-with-spl.bin*** zu finden.

Linux Kernel

Als erstes muss der Sourcecode für den Linux Kernel von Github geklont werden:

git clone https://github.com/torvalds/linux.git

Betrete anschließend das neue Verzeichnis linux und wende folgende Konfigurationsdatei an:

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- sunxi_defconfig

Zu beachten ist, dass es sich hierbei um eine minimal Konfiguration handelt und somit viele Treiber für z.B. USB Geräte fehlen. Zudem fehlt der DVP Kamera Treiber, dieser ist unter device driver -> Multimedia -> Platform zu finden. Diese können durch ***menuconfig*** ausgewählt werden.

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf- menuconfig

Der Buildprozess lässt sich mit

make ARCH=arm CROSS_COMPILE=arm-linux-gnueabihf-

starten.

Nach erfolgreichem Erstellen vom Linux Kernel ist dieser als ***arch/arm/boot/zImage*** zu finden.

USB OTG (Universal Serial Bus, On the Go)

Der Allwinner V3s verfügt über einen USB OTG Port. Das USB Interface ist nicht multiplexbar.

Port	Pin
USB-DM	110
USB-DP	111

Im Linux Kernel ist der Treiber für das USB Interface als sun4i-usb-phy und sun4i-a10-musb zu finden.

Konfiguration im Devicetree

Bei der Verwendung der Standard Devicetree Datei ***sun8i-v3s.dtsi*** ist im folgenden Beispiel erklärt wie der ***USB Controller*** aktiviert wird: Die vollständige Devicetreedatei kann auf Github heruntergeladen werden. Auszug aus der ***sun8i-v3s.dtsi:***

```
usb_otg: usb@1c19000 {
                compatible = "allwinner,sun8i-h3-musb";
                reg = <0x01c19000 0x0400>;
                clocks = <&ccu CLK_BUS_OTG>;
                resets = <&ccu RST_BUS_OTG>;
                interrupts = <GIC_SPI 71 IRQ_TYPE_LEVEL_HIGH>;
                interrupt-names = "mc";
                phys = <&usbphy 0>;
                phy-names = "usb";
                extcon = <&usbphy 0>;
                status = "disabled";
        };

usbphy: phy@1c19400 {
                compatible = "allwinner,sun8i-v3s-usb-phy";
                reg = <0x01c19400 0x2c>,
                      <0x01c1a800 0x4>;
                reg-names = "phy_ctrl",
                            "pmu0";
                clocks = <&ccu CLK_USB_PHY0>;
                clock-names = "usb0_phy";
                resets = <&ccu RST_USB_PHY0>;
                reset-names = "usb0_reset";
                status = "disabled";
                #phy-cells = <1>;
        };
```

Dafür muss die Boardfile ***sun8i-v3s-blueberrypi.dts*** bearbeitet werden:

```
leds {
    compatible = "gpio-leds";

    act_led {
        label = "act_led:green:usr";
        gpios = <&pio 1 2 GPIO_ACTIVE_HIGH>; /* PB2 */
        linux,default-trigger = "mmc0";
        default-state = "off";
        };
    };
};

&usb_otg {
    dr_mode = "otg";
    status = "okay";
};

&usbphy {
    pinctrl-0 = <&usb0_id_detect_pin>;
    usb0_id_det-gpio = <&pio 5 6 GPIO_ACTIVE_LOW>;
    status = "okay";
};

&ehci0 {

    status = "okay";
};

&ohci0 {

    status = "okay";
};
```

Der EHCI und OHCI Controller werden benötigt um USB 1.0 und USB 2.0 Geräte zu unterstützen. Werden die beiden Controller nicht im Devicetree aktiviert, entsteht kein Host Verhalten des V3s SoC. Momentan ist es nicht möglich USB 3.0 Geräte an den V3s anzuschließen, da dies eine Kernel Panic hervorruft.

USB Device

Der V3s SoC verfügt über einen USB OTG Port, d.h. dass er als USB Device oder Host konfiguriert werden kann. Beim Blueberry PI befindet sich der USB_ID Pin an PF6. Dank der Treiberunterstützung im Linux Kernel kann der V3s als virtuelle USB zu Ethernet Karte und auch als USB zu UART Konverter fungieren. Dank des offenen USB Stacks lassen sich auch eigene USB Devices programmieren.

FEL-Mode

Über USB ist es möglich ein angeschlossenes SPI Flash zu beschreiben. Dazu muss die SD Karte aus dem Halter entfernt werden. Die weiteren Schritte zum Beschreiben sind auf Sunxi unter FEL/USBBoot zu finden.

Ethernet

Der Allwinner V3s verfügt über einen Ethernet Port. Das Ethernet Interface ist nicht multiplexbar.

Port	Pin
EPHY_RXN	89
EPHY_RXP	90
EPHY_TXN	91
EPHY_TXP	91
EPHY_LINK_LED	77
EPHY_SPD_LED	78

Im Linux Kernel ist der Treiber für das Ethernet Interface als dwmac-sun8i zu finden.

Konfiguration im Devicetree

Bei der Verwendung der Standard Devicetree Datei ***sun8i-v3s.dtsi*** ist im folgenden Beispiel erklärt wie das Ethernet Interface aktiviert wird. Dafür muss die Boardfile ***sun8i-v3s-blueberrypi.dts*** bearbeitet werden:

```
&emac {
    allwinner,leds-active-high;
    status = "okay";
};
```

Einstellen einer festen MAC Adresse

Bei jedem Start / Reboot erhält der Ethernet PHY eine neue MAC Adresse vom Treiber, um dies zu unterbinden muss folgendes in die /etc/network/interfaces eingetragen werden:

```
allow-hotplug eth0
iface eth0 inet dhcp
    hwaddress ether 00:11:22:33:44:55
```

Parallel RGB LCD interface

Der Allwinner V3s verfügt über eine parallele RGB LCD Schnittstelle, welche direkt LCDs ansteuern kann. Es können so gut wie alle 40 Pin parallelen RGB LCDs verwendet werden, zum Beispiel:

AT050TN33 (480x320)

HW800480F-3E-0B-10 (800x480)

Das parallele RGB interface ist nicht multiplexbar und besitzt folgendes Pinmapping:

Port	Pin
CSI_PCLK	PE0
CSI_MCLK	PE1
CSI_HSYNC	PE2
CSI_VSYNC	PE3
LCD_D2	PE4
LCD_D3	PE5
LCD_D4	PE6
LCD_D5	PE7
LCD_D6	PE8
LCD_D7	PE9
LCD_D10	PE10
LCD_D11	PE11
LCD_D12	PE12
LCD_D13	PE13
LCD_D14	PE14
LCD_D15	PE15
LCD_D18	PE16
LCD_D19	PE17
LCD_D20	PE18
LCD_D21	PE19
LCD_D22	PE23
LCD_D23	PE24

Konfiguration in u-boot:

Das Displaytiming wird bereits in u-boot konfiguriert um schon beim Booten eine Displayausgabe zu erhalten. Beim Starten des Linux Kernels übergibt u-boot jegliche Informationen über das Displaytiming sowie die Speicheradresse des zuvor initialisierten Framebuffers an den Linux Kernel. Somit ist es theoretisch nicht nötig die Displayhardware im Devicetree zu konfigurieren.

Es muss lediglich ein Framebuffer für u-boot angegeben werden, Auszug aus der

sun8i-v3s-blueberrypi.dts:

```
/ {
        #address-cells = <1>;
        #size-cells = <1>;
        interrupt-parent = <&gic>;


        chosen {
                #address-cells = <1>;
                #size-cells = <1>;
                ranges;


                fb0: framebuffer@0 {
                        compatible = "allwinner,simple-framebuffer", "simple-framebuffer";
                        allwinner,pipeline = "de0-lcd0";
                        clocks = <0x2 0x21 0x2 0x23 0x2 0x3f 0x2 0x40>;
                        status = "disabled";
                };
        };
```

Momentan wird der simple Framebuffer lediglich in meinem u-boot Repository unterstützt.

Um u-boot mit Unterstützung des parallelen RGB Interfaces zu bauen, siehe u-boot. Dort sind auch die benötigten Displaytimings für die beiden oben genannten Displays angegeben sowie für den RGB zu VGA Konverter.

Parallel CSI

Der Allwinner V3s verfügt über einen parallelen CSI Port. Das Interface ist nicht multiplexbar.

Port	PIN	OV7670	OV2640	ADV7611
CSI_PCLK	PE0	PLCK	PCLK	LLC
CSI_MCLK	PE1	XCLK	XCLK	X
CSI_HSYNC	PE2	HREF	HREF	X**
CSI_VSYNC	PE3	VSYNC	VSYNC	X**
CSI_D0	PE4	X	X	P8
CSI_D1	PE5	X	X	P9
CSI_D2	PE6	X	D0	P10
CSI_D3	PE7	X	D1	P11
CSI_D4	PE8	D0	D2	P12
CSI_D5	PE9	D1	D3	P13
CSI_D6	PE10	D2	D4	P14
CSI_D7	PE11	D3	D5	P15
CSI_D8	PE12	D4	D6	P0
CSI_D9	PE13	D5	D7	P1
CSI_D10	PE14	D6	D8	P2
CSI_D11	PE15	D7	D9	P3
CSI_D12	PE16	X	X	P4
CSI_D13	PE17	X	X	P5
CSI_D14	PE18	X	X	P6
CSI_D15	PE19	X	X	P7
CSI_SCK / I2C1*	PE21	SIOC	SIOC	SCL
CSI_SDA / I2C1*	PE22	SIOD	SIOD	SDA

*zu beachten ist, dass lediglich I2C1 bzw. CSI_I2C für die Ansteuerung der Kamera verwendet werden kann.

** HSync und VSync Signale werden nicht benötigt, da BT.656 mit embedded Sync verwendet wird.

Konfiguration im Devicetree

Bei der Verwendung der Standard Devicetree Datei ***sun8i-v3s.dtsi*** ist im folgenden Beispiel erklärt wie der ***parallele CSI Port*** aktiviert wird: Die vollständige Devicetreedatei kann auf Github heruntergeladen werden. Auszug aus der ***sun8i-v3s.dtsi:***

```
csi1: csi@1cb4000 {
        compatible = "allwinner,sun8i-v3s-csi";
        reg = <0x01cb4000 0x1000>;
        interrupts = <GIC_SPI 84 IRQ_TYPE_LEVEL_HIGH>;
        clocks = <&ccu CLK_BUS_CSI>,
                 <&ccu CLK_CSI1_SCLK>,
                 <&ccu CLK_DRAM_CSI>;
        clock-names = "bus", "mod", "ram";
        resets = <&ccu RST_BUS_CSI>;
        status = "disabled";
    };
```

OV7670

Konfiguration der OV7670 in ***sun8i-v3s-blueberrypi-csi-ov7670.dts***:

```
&csi1 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&csi1_clk &csi1_8bit>;

port {
    csi1_ep: endpoint {
        remote-endpoint = <&ov7670_0>;
         bus-width = <8>;
              hsync-active = <0>;
              vsync-active = <0>;
              pclk-sample = <1>;

        };
    };
 };

&i2c1 {
        pinctrl-0 = <&i2c1_pins>;
    pinctrl-names = "default";
    status = "okay";

    ov7670: camera@21 {
        compatible = "ovti,ov7670";
        reg = <0x21>;
        pinctrl-names = "default";
        pinctrl-0 = <&csi1_mclk>;
        clocks = <&ccu CLK_CSI1_MCLK>;
        clock-names = "xclk";
        assigned-clock-rates = <24000000>;

        port {
            ov7670_0: endpoint {
                remote-endpoint = <&csi1_ep>;
                 hsync-active = <1>;
                vsync-active = <0>;
                bus-width = <8>;
                data-active = <1>;
                pclk-sample = <1>;

            };
        };
    };
};
```

Unter ***&csi*** wird das Interface aktiviert und die dazu gehörigen Pins über ***&csi1_clk*** und ***&csi1_8bit*** dem Interface zugewiesen. Dies geschieht im ***&pio*** (Pincontroller):

```
&pio {
    csi1_8bit: csi1-8bit@0 {
            pins      = "PE8","PE9","PE10","PE11","PE12","PE13","PE14","PE15";
            bias-disable;
            function = "csi";
        };

    csi1_clk: csi1-clk@0 {
            pins = "PE0","PE2","PE3";
            bias-disable;
            function = "csi";
        };

    csi1_mclk: csi1-mclk@0 {
            pins = "PE1";
            bias-disable;
            function = "csi";
        };

    i2c1_pins: i2c1 {
        pins = "PE21", "PE22";
        function = "i2c1";
    };
```

Anschließend wird im ***endpoint*** die verwendete Kamera näher definiert, wie z.B. die Hsync, Vsync und die PCLK Eigenschaften.

Unter ***&i2c1*** werden die benutzen Pins zugewiesen und das I2C1 Interface aktiviert, innerhalb des ***&i2c1*** wird die Kamera konfiguriert. Mit reg = <0x21>; wird die I2C Adresse der Kamera angegeben.

Danach wird die MCLK, die den Arbeitstakt für die Kamera bereitstellt, konfiguriert. Hierzu wird der verwendete Pin angegeben, siehe ***&pio***, sowie die Frequenz in Hz.

Im ***endpoint*** von &i2c1 werden wieder die Eigenschaften von Hsync, Vsync und der PCLK definiert.

Zu beachten ist, dass die beiden Endpoints unter ***&csi*** und ***&i2c1*** aufeinander verweisen, siehe remote-endpoint = <>;

OV2640

Konfiguration der OV2640 in ***sun8i-v3s-blueberrypi-csi-ov2640.dts***:

```
&csi1 {
    status = "okay";
    pinctrl-names = "default";
    pinctrl-0 = <&csi1_clk &csi1_10bit>;

port {


    csi1_ep: endpoint {

        remote-endpoint = <&ov2640_0>;
        hsync-active = <0>;
        vsync-active = <0>;
        bus-width = <10>;
        pclk-sample = <1>;

        };
    };
 };

&i2c1 {
        pinctrl-0 = <&i2c1_pins>;
        pinctrl-names = "default";
        status = "okay";


    ov2640: camera@30 {
        compatible = "ovti,ov2640";
        reg = <0x30>;
        pinctrl-names = "default";
        pinctrl-0 = <&csi1_mclk>;
        clocks = <&ccu CLK_CSI1_MCLK>;
        clock-names = "xvclk";
        assigned-clocks = <&ccu CLK_CSI1_MCLK>;
        assigned-clock-rates = <24000000>;

        port {
            ov2640_0: endpoint {
                remote-endpoint = <&csi1_ep>;
                bus-width = <10>;
            };
        };
    };
};
```

Unter ***&csi*** wird das Interface aktiviert und die dazu gehörigen Pins über ***&csi1_clk*** und ***&csi1_10bit*** dem Interface zugewiesen. Dies geschieht im ***&pio*** (Pincontroller):

```
&pio {
    csi1_10bit: csi1-10bit@0 {
            pins      = "PE6","PE7","PE8","PE9","PE10","PE11","PE12","PE13","PE14","PE15";
            bias-disable;
            function = "csi";
        };

    csi1_clk: csi1-clk@0 {
            pins = "PE0","PE2","PE3";
            bias-disable;
            function = "csi";
        };

    csi1_mclk: csi1-mclk@0 {
            pins = "PE1";
            bias-disable;
            function = "csi";
        };

    i2c0_pins: i2c0 {
        pins = "PB6", "PB7";
        function = "i2c0";
    };

    i2c1_pins: i2c1 {
        pins = "PE21", "PE22";
        function = "i2c1";
    };
```

ADV7611

WIP

Verwendung der Kamera unter Linux

Mit dem Befehl: ***dmesg | grep ov*** kann überprüft werden ob die Kamera erkannt worden ist. Sollte die Kamera nicht erkannt worden sein, könnte dies an der fehlenden MCLK liegen.

Wurde die Kamera erkannt, kann nun mit fswebcam ein Bild aufgenommen werden:

```
fswebcam -S 20 -d /dev/video0 -p YUYV -r 640x480 YUYV.jpg
```

Mit ffmpeg ist es möglich Videos aufzuzeichnen:

```
ffmpeg -f v4l2 -framerate 25 -video_size 640x480 -i /dev/video0 output.mp4
```

SPI (Serial Peripheral Interface)

Der Allwinner V3s verfügt über ein Serial Peripheral Interface. Beim SPI ist es nicht möglich die Pins zu multiplexen.

Port	Pin
SPI0_MISO	PC0
SPI0_CLK	PC1
SPI0_CS	PC2
SPI0_MOSI	PC3

Im Linux Kernel ist der Treiber für das SPI als sun8i-h3-spi zu finden.

Konfiguration im Devicetree

Bei der Verwendung der Standard Devicetree Datei ***sun8i-v3s.dtsi*** ist im folgenden Beispiel erklärt wie das ***SPI*** aktiviert wird. Die vollständige Devicetreedatei kann auf Github heruntergeladen werden. Dafür muss die Boardfile ***sun8i-v3s-blueberrypi.dts*** bearbeitet werden:

```
&spi0 {

    pinctrl-0 = <&spi0_pins>;
    pinctrl-names = "default";
    status = "okay";
    #address-cells = <1>;
    #size-cells = <0>;

    };
```

Als erstes wird das SPI mit den ***spi0_pins*** initialisiert. Die zwei Zeilen: ***#adress-cells = <1>;*** und

#size-cells = <0>; sind Helfer welche den ***DMA*** Support auf dem ***SPI*** ermöglichen.

Auffällig ist, dass in der ***sun8i-v3s-blueberrypi.dts*** keine Pinzuweisung stattfindet. Die Pinzuweisung findet in der ***sun8i-v3s.dtsi*** statt, da das ***SPI*** nicht multiplexbar ist und somit nur eine korrekte Pinzuweisung existiert.

Möchte man nun weitere Geräte an das SPI anschließen, so kann auf das große Treiberspektrum des Linux Kernels zurückgegriffen werden. Es kann z.B: der MCP2515 (CAN-Controller), ENC28J60 (Ethernet Interface), SPI Displays oder auch zahllose Sensoren und Aktoren angeschlossen werden.

Weitere Beispiele mit Sensoren / Aktoren im Devicetree folgen noch!

I²C

Der Allwinner V3s verfügt über zwei I²C (TWI) Busse:

Port	Multiplex Option 1	Multiplex Option 2
TWI0-SDA	PB7	
TWI0-SCK	PB6	
TWI1-SDA	PB9	PE22
TWI1-SCK	PB8	PE21

I2C1 wird standardmäßig für die Ansteuerung einer seriellen oder parallelen Kamera verwendet.

Bei dem verwendetem I2C Controller handelt es sich um einen Marvell mv64xxx I2C.

Im Linux Kernel ist dieser als CONFIG_I2C_MV64XXX zu finden.

Konfiguration im Devicetree

Bei der Verwendung der Standard Devicetree Datei ***sun8i-v3s.dtsi*** ist im folgenden Beispiel erklärt wie der I1C0 Bus aktiviert wird. Die vollständige Devicetreedatei kann auf Github heruntergeladen werden. Dafür muss die Boardfile ***sun8i-v3s-blueberrypi.dts*** bearbeitet werden:

```
&i2c0 {
    pinctrl-0 = <&i2c0_pins>;
    pinctrl-names = "default";
    status = "okay";
};

&pio {

    i2c0_pins: i2c0 {
        pins = "PB6", "PB7";
        function = "i2c0";
    };
```

Als erstes wird der I2C0 Bus mit den ***i2c0_pins*** initialisiert. Die Pinzuweisung findet im ***PIO*** (Pincontroller) darunter statt. Zuerst wird die SCK des Busses und anschließend SDA als Pin angegeben.

Möchte man zusätzlich den zweiten I2C Bus aktivieren, so muss der Busname von ***i2c0*** auf ***i2c1*** geändert werden. Dies ist auch im Pincontroller zu tun.

Möchte man nun Sensoren oder Aktoren an den I2C Bus anschließen, so kann auf das große Treiberspektrum des Linux Kernels zurückgegriffen werden. Sollte ein Sensor noch nicht unterstützt sein, kann auch auf externe Python oder C Programme zurückgegriffen werden, welche auf die I2C Schnittstelle unter /dev/i2c-0 zugreifen.

Weitere Beispiele mit Sensoren / Aktoren im Devicetree folgen noch!

UART

Der Allwinner V3s verfügt über drei UART Busse:

Port	Multiplex Option 1	Multiplex Option 2
UART0-TXD	PB8*	PF2*
UART0-RXD	PB9*	PF4*
UART1-TXD	PE21	-
UART1-RXD	PE22	-
UART1-RTS	PE23	-
UART1-CTS	PE24	-
UART2-TXD	PB0	-
UART2-RXD	PB1	-
UART2-RTS	PB2	-
UART2-CTS	PB3	-

*(Pinkonflikt mit SDC0! Kann nur verwendet werden, wenn über das SPI-Flash gebootet wird.)

*(Serielle Standardausgabe von Linux und u-boot, Baud: 115200)

Konfiguration im Devicetree

Bei der Verwendung der Standard Devicetree Datei ***sun8i-v3s.dtsi*** ist im folgenden Beispiel erklärt wie der UART0 Bus aktiviert wird. Die vollständige Devicetreedatei kann auf Github heruntergeladen werden. Dafür muss die Boardfile ***sun8i-v3s-blueberrypi.dts*** bearbeitet werden:

```
&uart0 {
    pinctrl-0 = <&uart0_pins>;
    pinctrl-names = "default";
    status = "okay";
};

&pio {

    uart0_pins: uart0@0 {
        pins = "PB8", "PB9";
        function = "uart0";
    };
```

Als erstes wird der UART0 Bus mit den ***uart0_pins*** initialisiert. Die Pinzuweisung findet im ***PIO*** (Pincontroller) darunter statt. Zuerst wird der TXD Pin des Busses und anschließend RXD als Pin angegeben.

Möchte man zusätzlich den zweiten oder dritten UART Bus aktivieren, so muss der Busname von ***UART0*** auf ***UART1*** geändert werden. Dies ist auch im Pincontroller zu tun.

Audio Codec

Der Allwinner V3s verfügt über einen Audio Codec, dieser besteht aus einem Stereo DAC für die Wiedergabe und einem Stereo ADC zum Aufzeichnen von Audio.

Port	Pin
MICIN1P	113
MICIN1N	114
AVCC	115
AGND	116
VRA1	117
VRA2	118
HBIAS	119
HPOUTR	120
HPOUTL	121
HPVCCIN	122
HPVCCBP	123
HPCOMFB	124
HPCOM	125

Im Linux Kernel ist der Treiber für den Audiocodec als sun8i-codec zu finden.

Konfiguration im Devicetree

Bei der Verwendung der Standard Devicetree Datei ***sun8i-v3s.dtsi*** ist im folgenden Beispiel erklärt wie der ***Audio Codec*** aktiviert wird. Die vollständige Devicetreedatei kann auf Github heruntergeladen werden. Dafür muss die Boardfile ***sun8i-v3s-blueberrypi.dts*** bearbeitet werden:

```
&codec {
    allwinner,audio-routing = "Headphone", "HP", "Headphone", "HPCOM", "MIC1", "Mic", "Mic", "HBIAS";
    status = "okay";
};
```

Als erstes wird beim Audio Routing die Audio Senke und danach die Audio Quelle angegeben.

Senke <- Quelle

Headphone <- HP

Headphone <- HPCOM

MIC1 <- Mic

Mic <- HBIAS

Verwendung des Audiocodecs unter Linux

Wiedergabe:

Zuerst über amixer / alsamixer die Laustärke der Headphones erhöhen:

amixer -c 0 sset 'Headphone',0 100% unmute

Oder wahlweise über die GUI mit:

amixer

aplay test.mp3

Aufnahme mit dem Mikrofon:

amixer -c 0 cset numid=12 2 // Mikrofon aktivieren

arecord -D hw:0,0 -d 3 -f S16_LE -r 16000 tmp.wav

LRADC0

Der Allwinner V3s verfügt über einen Low resolution ADC. Dieser wird dazu verwendet um 4 Taster auszulesen. Hinter jedem Taster ist ein Widerstand verschaltet, welcher wenn er gedrückt wird einen Spannungsteiler formt. Der ADC liest die abfallende Spannung am Spannungsteiler und bestimmt somit welcher Taster gedrückt worden ist.

Port	Pin
LRADC0	112

Im Linux Kernel ist dieser als sun4i-a10-lradc-keys zu finden.

Konfiguration im Devicetree

Bei der Verwendung der Standard Devicetree Datei ***sun8i-v3s.dtsi*** ist im folgenden Beispiel erklärt wie der LRADC0 aktiviert wird. Die vollständige Devicetreedatei kann auf Github heruntergeladen werden. Dafür muss die Boardfile ***sun8i-v3s-blueberrypi.dts*** bearbeitet werden:

```
&lradc {
    vref-supply = <&reg_vcc3v0>;
    status = "okay";

    button@200 {
        label = "Volume Up";
        linux,code = <KEY_VOLUMEUP>;
        channel = <0>;
        voltage = <200000>;
    };

    button@400 {
        label = "Volume Down";
        linux,code = <KEY_VOLUMEDOWN>;
        channel = <0>;
        voltage = <400000>;
    };

    button@600 {
        label = "Select";
        linux,code = <KEY_SELECT>;
        channel = <0>;
        voltage = <600000>;
    };

    button@800 {
        label = "Start";
        linux,code = <KEY_OK>;
        channel = <0>;
        voltage = <800000>;
    };
};
```

Zuerst wird die Speisespannung des LRADC0 mit ***vref-supply = <>;*** festgelegt. Im Fall des Blueberry PIs sind dies 3V. Anschließend werden die vier Taster wie folgt definiert:

label = " "; Beschreibung des Tasters

linux,code = <>; Angabe des numerischen Keycodes

voltage = <>; Spannung des Spannungsteiler wenn der Taster gedrückt ist

Onewire Bus

Da der Allwinner V3s über kein natives Onewire Interface verfügt, muss dies über einen GPIO Port gebitbangt werden.

Konfiguration im Devicetree

```
onewire_device {
   compatible = "w1-gpio";
   gpios = <&pio 1 4 GPIO_ACTIVE_HIGH>; /* PB4 */
   pinctrl-names = "default";
   pinctrl-0 = <&my_w1_pin>;
};

&pio {

    my_w1_pin: my_w1_pin@0 {
        pins = "PB4";
        allwinner,function = "gpio_in";
        bias-pull-up;
    }
```

Zuerst wird unter der Angabe des zu verwendenden Treibers und dem benutzten GPIO Pin der Onewirebus konfiguriert. Im Pincontroller ***&pio*** muss der verwendete Pin für den Onewirebus noch zusätzlich definiert werden. Dieser ist als GPIO-In und mit einem Pull-Up zu konfigurieren.

Nun sollte unter /sys/bus/w1/devices/ alle angeschlossenen Onewiredevices sichtbar sein.

Um Sensoren auszulesen, können die einschlägigen Tutorials für andere Single Board Computer verwendet werden.

Troubleshooting Guide

Dieses Kapitel hilft dir dabei, wenn du den Blueberry Pi selbst in Betrieb nehmen willst.

Sollte dieser nicht booten, folge dieser Schritt für Schritt Anleitung um den Fehler zu finden.

- Im angeschalteten Zustand, ohne Boot Medium, verbraucht der SoC ungefähr 90mA @ 5V. (Versorgung durch DC Step down Module)
  Sollte dies der Fall sein liegt höchstwahrscheinlich ein Softwareproblem vor. Versuche den V3s mit dem fertigen Image von Github zu booten. Sollte dies nicht funktionieren teste auch andere SD Karten.

  Sollte der SoC wesentlich weniger verbrauchen folge diesen Schritten:

- Ist der V3s SoC korrekt auf der Platine platziert? Der kleine Kreis auf dem SoC markiert Pin 1.
- Alle Lötstellen überprüfen, sind alle Pins des SoC korrekt ausgerichtet und es existieren keine Lötbrücken?
- Ist das Masse Pad auf der Unterseite des SoC verlötet worden?
- Sind alle passiven Bauteile an der richtigen Stelle?
- Ist VCC – RTC und VCC – PE mit 3,3V verbunden? Beim Blueberry Pi geschieht dies mit zwei Jumpern die auf die jeweiligen beschrifteten Pinheader gesetzt werden.
- Sind alle nötigen Spannungen für den SoC vorhanden? (3,3V 3,0V 1,8V 1,2V)
- Oszilliert das 24Mhz Quarz?