8.7 FlexCAN

8.7.1 FlexCAN User Manual

Description

The FlexCAN module is a communication controller implementing the CAN protocol according to the CAN 2.0B protocol specification. NXP's LS1021A can support 4 Flexcan IP module instances if there are 2 CAN ports brought out on DB9 male connectors on the board.

Dependencies

Hardware:

One LS1021A board, Serial cable (female-to-female) having the following connection schema, so that the DB9 CAN ports on the LS1021A board can be connected properly:

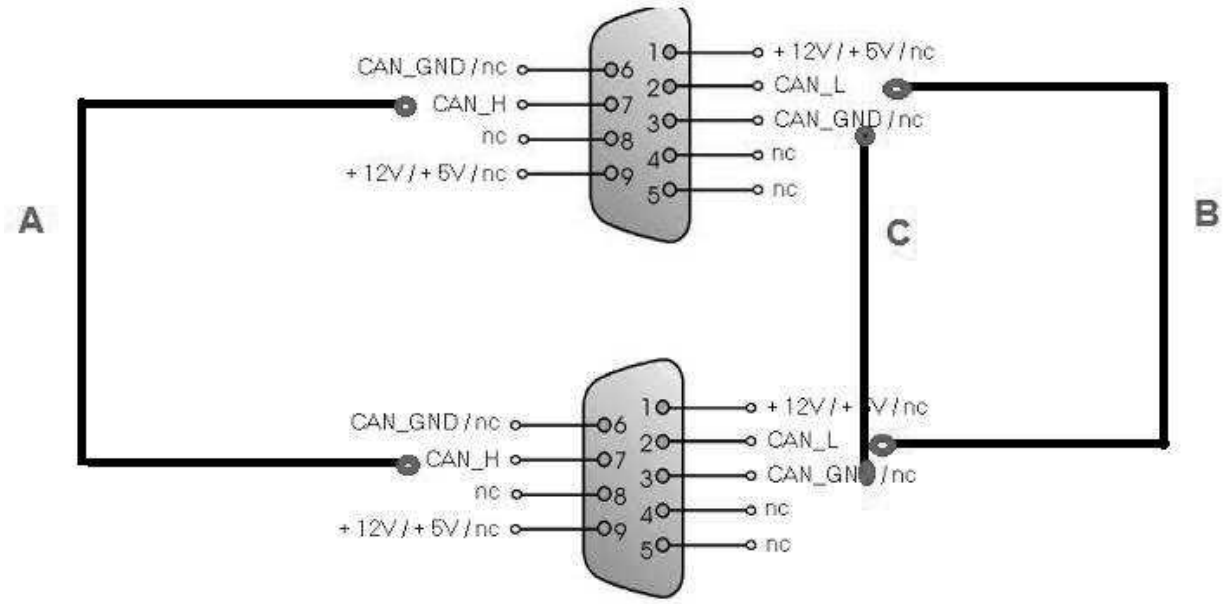

Figure 145. DB9 CAN Connections

Software:

RCW (LS1021A), U-Boot (LS1021A), Kernel (LS1021A), RFS (LS1021A Yocto RFS)

RCW Configuration

- Build the **rcw_1000_misc.bin** RCW file for FlexCAN using the following GIT tree: http://sw-stash.freescale.net/projects/SDK/repos/rcw/browse
- Flash the RCW on the LS1021A board.

U-Boot Configuration

Execute the following command at the u-boot prompt:

=> setenv hwconfig “can”

=> saveenv

Compile time options

N/A

Runtime options

Env Variable	Env Description	Option Description
hwconfig	setenv hwconfig "can"	This option is used to select CAN feature on the LS1021A board.

Kernel Configure Options

Tree View

Following is an example of how to enable FlexCAN driver support.

Kernel Configure Tree View Options	Description

`[*] Networking support -->`
`  <*> CAN bus subsystem support -->`
`  -- CAN bus subsystem support`
`  <*> Raw CAN Protocol (raw access with CAN-ID filtering)`
`  < > Broadcast Manager CAN Protocol (with content filtering)`
`  CAN Device Drivers -->`
`  <*> Platform CAN drivers with Netlink support`
`  [*] CAN bit-timing calculation`
`  <*> Support for NXP FlexCAN based chips`
	Enable FlexCAN driver and CAN protocol stack

NOTE

The FlexCAN driver can be statically/dynamically linked with the kernel Image.

Identifier

Option	Values	Default Value	Description
CONFIG_NET	y/n	y	Enable networking support
CONFIG_CAN	y/m/n	n	Enable CAN bus support
CONFIG_CAN_RAW	y/m/n	n	Enable CAN RAW Protocol support
CONFIG_CAN_DEV	y/m/n	n	Enable platform CAN driver support
CONFIG_CAN_CALC_BITTIMING	y/n	n	Enable baud rate setting using sysfs
CONFIG_CAN_FLEXCAN	y/m/n	n	Enable NXP FlexCAN protocol

Device Tree Binding

Below is the definition of the device tree node required by this feature

Property	Description
compatible = " fsl,ls1021a-flexcan"	Should be "fsl, ls1021a-flexcan"
reg = <0x0 0x2a70000 0x0 0x1000>	Offset and length of the register set for the device
interrupts = < GIC_SPI 127 IRQ_TYPE_LEVEL_HIGH>	IRQ line for FlexCAN controller CAN0
clocks = <&platform_clk 1>	CAN module Clock Source.
clock-names = <per>	CAN Engine Clock Source. This property selects the peripheral clock. Valid values are **ipg** and **per** **ipg**: CAN engine clock source is oscillator clock. Current release doesn't support this option. **per:** The CAN engine clock source is the peripheral clock (platform clock).

Below is an example device tree node required by FlexCAN.

```
can0@2a70000{
            compatible = "fsl, ls1021a-flexcan ";
            reg = <0x0 0x2a70000 0x0 0x1000>;
            interrupts = < GIC_SPI 127 IRQ_TYPE_LEVEL_HIGH >;
            clocks = <&platform_clk 1>;
            clock-names = <per>
};
```

Source Files

The following source file is related to Flexcan feature in u-boot.

Source File	Description
arch/arm/cpu/armv7/ls102xa/fdt.c	Providing the fix-up for clock_freq in the dts file.
board/freescale/ls1021aqds/ls1021aqds.c	Provides the support for board level FlexCAN muxing when u-boot environment variable **hwconfig** is set to can.

The following source files are related to Flexcan feature in Linux kernel.

Source File	Description
drivers/net/can/flexcan.c	Flexcan driver module

Board Connections

On the LS1021A board, mount Jumper 1 as shown in figure below to ensure that HI speed mode is working and also ensure that the 120ohm resistance b/w CAN_H and CAN_L is mounted properly on the board.

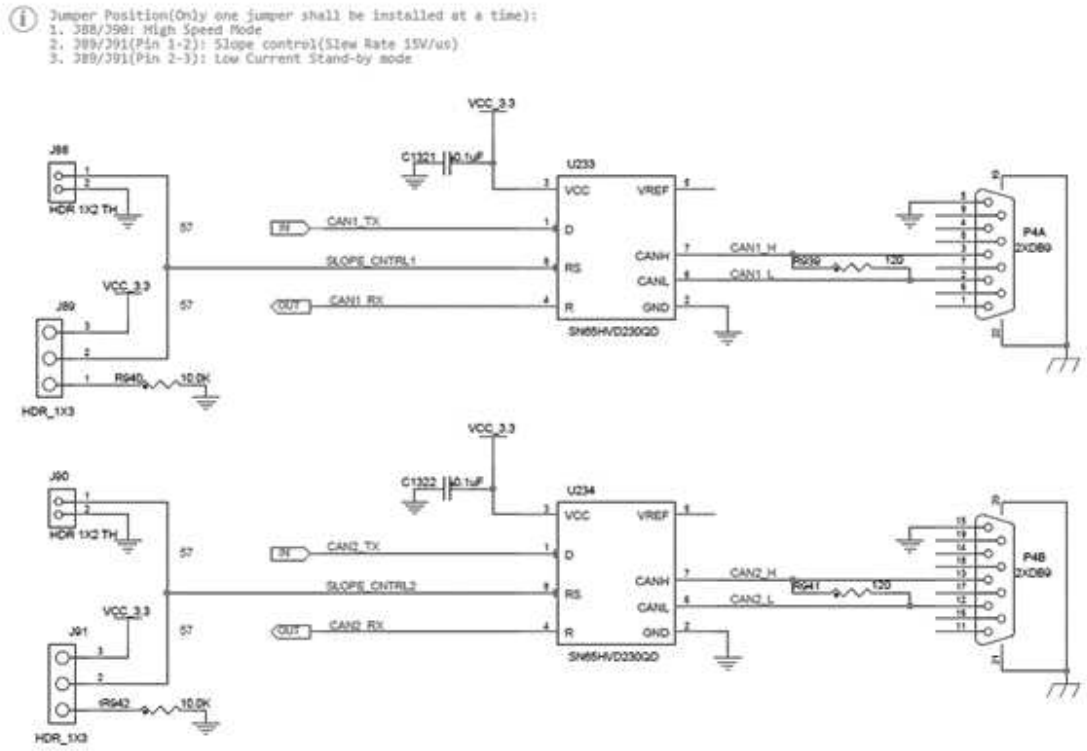

Figure 146. Board Connections

User Space Application

Please select the following package for testing Flexcan. Please refer to the LS1 Yocto WiKi page (http://wiki.freescale.net/display/layrarch/Yocto+framework+for+LS1+image+build) for the detailed build procedure.

Command Name	Description	Package Name
cansend	Userspace SocketCAN test program for sending CAN frames on a CAN interface	cansend
candump	Userspace SocketCAN test program for sniffing CAN frames received on a CAN interface	candump
ip	The iproute package contains networking utilities (ip and rtmon, for example) which are designed to use the advanced networking capabilities of the Linux kernel.	iproute(2)

Verification in U-Boot

N/A

Verification in Linux

To cross check whether FlexCAN has been configured in the kernel or not, run the following command at Linux prompt:

```
root@ls1021aqds:~# cat /proc/interrupts
           CPU0       CPU1
118:       3889          0       GIC 118  serial
120:        126          0       GIC 120  2180000.i2c
```

```
125:          0          0       GIC 125  xhci-hcd:usb1
126:       1521          0       GIC 126  mmc0
129:          0          0       GIC 129  2110000.dspi
135:          2          0       GIC 135  1710000.jr
136:          0          0       GIC 136  1720000.jr
137:          0          0       GIC 137  1730000.jr
138:          0          0       GIC 138  1740000.jr
150:       5371          0       GIC 150  Freescale ftm timer
158:          1          0       GIC 158  can0
167:          0          0       GIC 167  eDMA
189:         21          0       GIC 189  eth2_g0_tx
190:          4          0       GIC 190  eth2_g0_rx
191:          0          0       GIC 191  eth2_g0_er
201:          0          0       GIC 201  ds3232
IPI0:         0          0  CPU wakeup interrupts
IPI1:         0       5359  Timer broadcast interrupts
IPI2:      3158       3718  Rescheduling interrupts
IPI3:         0          0  Function call interrupts
IPI4:         3          3  Single function call interrupts
IPI5:         0          0  CPU stop interrupts
Err:          0
```

Test Procedure

Guide lines for testing FlexCAN on the LS1021A board.

Internal LoopBack test (SoC Level loopback)

- Enable CAN interface **can0** with root permissions on the Linux prompt

```
# ip link set can0 up type can bitrate 125000 loopback on
```

- Set candump to snif packets on can0 interface:

```
# candump can0 -n 2 &
```

- Set cansend to send a packet with standard identifier on can0 interface:

```
# cansend can0 5A1#123412341234
```

Expected behavior after internal loopback testing:

- CAN0 interface which is receiving the can frame should show, that it has read the frame ID and data correctly:

```
5A1 [6] 12 34 12 34 12 34
```

- After the successful Tx or RX from the CAN controller, the Tx and Rx Interrupt should increment for CAN0 interface.

```
   root@ls1021aqds:~# cat /proc/interrupts
           CPU0       CPU1
118:       3889          0       GIC 118  serial
120:        126          0       GIC 120  2180000.i2c
125:          0          0       GIC 125  xhci-hcd:usb1
126:       1521          0       GIC 126  mmc0
129:          0          0       GIC 129  2110000.dspi
135:          2          0       GIC 135  1710000.jr
136:          0          0       GIC 136  1720000.jr
137:          0          0       GIC 137  1730000.jr
138:          0          0       GIC 138  1740000.jr
```

```
150:       5371          0       GIC 150  Freescale ftm timer
158:          2          0       GIC 158  can0
167:          0          0       GIC 167  eDMA
189:         21          0       GIC 189  eth2_g0_tx
190:          4          0       GIC 190  eth2_g0_rx
191:          0          0       GIC 191  eth2_g0_er
201:          0          0       GIC 201  ds3232
IPI0:          0          0  CPU wakeup interrupts
IPI1:          0       5359  Timer broadcast interrupts
IPI2:       3158       3718  Rescheduling interrupts
IPI3:          0          0  Function call interrupts
IPI4:          3          3  Single function call interrupts
IPI5:          0          0  CPU stop interrupts
Err:           0
```

NOTE

The underlined number represents the interrupts on successful Tx or RX. So this number should increase on successful Tx and Rx.

External LoopBack test (Single Board).

- Attach the CAN0 and CAN1 interface on the LS1021A board with serial cable (prepared as mentioned in **Dependencies** section).
- Enable the CAN0 and CAN1 interface on the board:

```
# ip link set can0 up type can bitrate 125000
# ip link set can1 up type can bitrate 125000
```

- Set candump to snif packets on can0 interface:

```
# candump can0 -n 1 &
```

- Set cansend to send a packet with standard identifier on can1 interface:

```
# cansend can1 5A1#123412341234
```

Expected behavior after external loopback testing:

- CAN0 interface which is receiving the can frame should show, that it has read the frame ID and data correctly:

```
5A1 [6] 12 34 12 34 12 34
```

- After the successful Tx or RX from the CAN controllers, the Tx Interrupt should increase for CAN1 interface and Rx Interrupt should increment for CAN0 interface.

Benchmarking

TBD

Known Bugs, Limitations, or Technical Issues

Currently we are running Flexcan with Backwards Compatibility Configuration.

Supporting Documentation

SoC Reference Manual (for eg. LS1021A RM)