General A18i

- CAN Communication

CAN Communication

A good example demonstrating the CAN communication between two URVEBoard A18i and A18PRO requires two URVEBoards. Connect them with a wire plugged into CAN1 on the first board and CAN2 on the second board.

Sending and receiving on first board

```
# ip link set can1 up type can bitrate 125000
# cansend can1  123#DEADBEEF
# candump can1
can1  123   [4]  DE AD BE EF
```

Sending and receiving on second board

```
# ip link set can0 up type can bitrate 125000
# candump can0
can0  123   [4]  DE AD BE EF
# cansend can0  123#DEADBEEF
```

CAN explanation

URVEBoards equipped with a CAN interface come with a pre-installed set of can-utils tools, which includes: candump, canplayer, cansend, cangen, and canbusload.

By using the command:

```
# candump can0
```

you can see all the incoming messages on the can0 port. By using the command:

```
# candump can0, 0x546:0x7FF
```

you will only see messages with the ID of 0x546. You can also add multiple filters at once. For example, if you only want to see IDs 0x546 and 0x291, use the command:

```
# candump can0, 0x546:0x7FF
```

You can also exclude a single ID using:

```
# candump can0, 0x546~0x7FF
```

Using CAN in URVEBoards in C++

When using BSP you need to find arm-linux-gnueabihf-cpp and run:

```
$ /opt/gcc/bin/arm-linux-gnueabihf-cpp cantransmit.c -o cantransmit
$ /opt/gcc/bin/arm-linux-gnueabihf-cpp canreceive.c -o canreceive
$ /opt/gcc/bin/arm-linux-gnueabihf-cpp canfilter.c -o canfilter
```

If you want to compile it directly on Debian powered URVEBoard:

```
gcc cantransmit.c -o cantransmit
gcc canreceive.c -o canreceive
gcc canfilter.c -o canfilter
```

canfilter.c - Filtering can messages

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>

#include <linux/can.h>
#include <linux/can/raw.h>

int main(int argc, char **argv)
{
	int s, i;
```

```
int nbytes;
struct sockaddr_can addr;
struct ifreq ifr;
struct can_frame frame;

printf("CAN Sockets Receive Filter Demo\r\n");

if ((s = socket(PF_CAN, SOCK_RAW, CAN_RAW)) < 0) {
    perror("Socket");
    return 1;
}

strcpy(ifr.ifr_name, "can0" );
ioctl(s, SIOCGIFINDEX, &ifr);

memset(&addr, 0, sizeof(addr));
addr.can_family = AF_CAN;
addr.can_ifindex = ifr.ifr_ifindex;

if (bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
    perror("Bind");
    return 1;
}

struct can_filter rfilter[1];

rfilter[0].can_id   = 0x550;
rfilter[0].can_mask = 0xFF0;
//rfilter[1].can_id   = 0x200;
//rfilter[1].can_mask = 0x700;

setsockopt(s, SOL_CAN_RAW, CAN_RAW_FILTER, &rfilter, sizeof(rfilter));

nbytes = read(s, &frame, sizeof(struct can_frame));

if (nbytes < 0) {
    perror("Read");
    return 1;
}
```

```
	printf("0x%03X [%d] ",frame.can_id, frame.can_dlc);

	for (i = 0; i < frame.can_dlc; i++)
		printf("%02X ",frame.data[i]);

	printf("\r\n");

	if (close(s) < 0) {
		perror("Close");
		return 1;
	}

	return 0;
}
```

canreceive.c - receiving from cantransmit.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>

#include <linux/can.h>
#include <linux/can/raw.h>

int main(int argc, char **argv)
{
	int s, i;
	int nbytes;
	struct sockaddr_can addr;
	struct ifreq ifr;
	struct can_frame frame;
```

```
    printf("CAN Sockets Receive Demo\r\n");

    if ((s = socket(PF_CAN, SOCK_RAW, CAN_RAW)) < 0) {
        perror("Socket");
        return 1;
    }

    strcpy(ifr.ifr_name, "can0" );
    ioctl(s, SIOCGIFINDEX, &ifr);

    memset(&addr, 0, sizeof(addr));
    addr.can_family = AF_CAN;
    addr.can_ifindex = ifr.ifr_ifindex;

    if (bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
        perror("Bind");
        return 1;
    }

    nbytes = read(s, &frame, sizeof(struct can_frame));

     if (nbytes < 0) {
        perror("Read");
        return 1;
    }

    printf("0x%03X [%d] ",frame.can_id, frame.can_dlc);

    for (i = 0; i < frame.can_dlc; i++)
        printf("%02X ",frame.data[i]);

    printf("\r\n");

    if (close(s) < 0) {
        perror("Close");
        return 1;
    }

    return 0;
```

```
}
```

cantransmit.c - transmiting to canreceive.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>

#include <net/if.h>
#include <sys/ioctl.h>
#include <sys/socket.h>

#include <linux/can.h>
#include <linux/can/raw.h>

int main(int argc, char **argv)
{
	int s;
	struct sockaddr_can addr;
	struct ifreq ifr;
	struct can_frame frame;

	printf("CAN Sockets Demo\r\n");

	if ((s = socket(PF_CAN, SOCK_RAW, CAN_RAW)) < 0) {
		perror("Socket");
		return 1;
	}

	strcpy(ifr.ifr_name, "can0" );
	ioctl(s, SIOCGIFINDEX, &ifr);

	memset(&addr, 0, sizeof(addr));
	addr.can_family = AF_CAN;
	addr.can_ifindex = ifr.ifr_ifindex;
```

```
    if (bind(s, (struct sockaddr *)&addr, sizeof(addr)) < 0) {
        perror("Bind");
        return 1;
    }

    frame.can_id = 0x555;
    frame.can_dlc = 5;
    sprintf(frame.data, "Hello");

    if (write(s, &frame, sizeof(struct can_frame)) != sizeof(struct can_frame)) {
        perror("Write");
        return 1;
    }

    if (close(s) < 0) {
        perror("Close");
        return 1;
    }

    return 0;
}
```