SCI2C on Host Platform

- DocRevision : 0.90
- Date : 2018-02-09

Host Platform specific information on the SCI2C protocol.

(1) Introduction

The Host SW contains an SCI2C protocol implementation (`hostLib/libCommon/sci2c.*` wrapped inside `hostLib/libCommon/smComSCI2C.*`). To ensure compatibility with SCI2C, the host platform's I2C driver needs to support:

- Block read (i.e. the response length is encoded in the first byte of the response; a feature typically associated with SMBUS)
- Repeated start

The SCI2C protocol used by the A71CH is described in full in:
SCI2C Protocol Specification Rev 1.5 or later (restricted to 1.x Revision, NOT compatible with Rev. 2.x)

(2) SCI2C on MCIMX6UL-EVK

We use the standard I2C driver that comes bundled with Yocto for the i.MX6 platform. In Linux the I2C driver implementation is layered. The top-level layer is contained in a file called 'i2c-dev.c' and is called the 'I2C device driver'. The bottom layer deals with the specific hardware of the I2C controller and is specific to the processor or board used, it is called the 'I2C bus driver'. In case of the MCIMX6UL-EVK board the bus driver is contained in a file called 'i2c-imx.c'.

The following code fragment highlights the correct invocation of the standard I2C driver to enable the Repeated Start and Block Read features.

By passing the two message structures via the packets structure as a parameter to the ioctl call one ensures a Repeated Start is triggered.

By setting the `'I2C_M_RECV_LEN'` bit in `'messages`[1].flags' one ensures the usage of the Block Read feature. Because the Block Read feature is requested the `'len'` and `'buf`[0]' field values of the `messages`[1] structure need special attention:

- `message`[1].len is first interpreted by the I2C device driver as the size of the buffer (pRx) provided by the caller. This value MUST be at least 33, a condition fulfilled by providing a buffer of 256 byte.
- The value contained in `messages`[1].buf[0] must be at least '1'. On the i.MX6 the value MUST be '1'.

If both conditions above are met, the I2C device driver will replace the value of `message`[1].len by the value contained in `messages`[1].buf0. Only then the platform specific I2C bus driver is invoked.

```
..
messages[0].addr = axSmDevice addr;
```

```
messages[0].flags = 0;
messages[0].len = txLen;
messages[0].buf = pTx;
messages[1].addr = axSmDevice_addr;
messages[1].flags = I2C_M_RD | I2C_M_RECV_LEN;
messages[1].len = 256;
messages[1].buf = pRx;
messages[1].buf[0] = 1;
/* Send the request to the kernel and get the result back */
packets.msgs = messages;
packets.nmsgs = 2;
r = ioctl(axSmDevice, I2C_RDWR, &packets);
..

(1) Code fragment from hostLib/platform/imx/i2c_a7.c illustrating correct I2C driver in
```

Furthermore one needs to ensure the packet size on the bus is set correctly during the initial SCI2C Parameter Exchange. By defining ‘PLATFORM_IMX’ at compilation time the correct packet size is set.
The following code fragment highlights the specific section of the source code.

```
..
#ifdef PLATFORM_IMX
#define SMCOM_MAX_BYTES (eMax31BytesS2M)
#else
#define SMCOM_MAX_BYTES (eMax254BytesS2M)
#endif
..

(2) Code fragment from hostLib/libCommon/smCom/sci2c_cfg.h Maximum packet size for SCI2
```