Using the maXTouch Linux Driver

TABLE OF CONTENTS

1.0 INTRODUCTION

This application note introduces the maXTouch Linux driver, its features and available debug options to help developers with driver integration. It also provides details on how to configure and build the kernel on a test platform.

The maXTouch Linux driver is designed to support the Linux input subsystem. This driver resides in the **drivers/input/touchscreen** directory of the kernel and interfaces with the hardware to generate 2D touch events to the Linux input subsystem.

The Linux driver can be compiled as an in-built driver or as a module to be loaded after the initial kernel boot. The application note provides details regarding function of the driver and the debug options that are available through the touch driver along with information regarding the testing of the touch driver.

This application note assumes that the user has a Linux based PC (either actual or a virtual machine) and has basic knowledge of the Linux OS. It also assumes basic use of Github. All testing has been done using the Ubuntu OS version 16.04. Any critical libraries or tools that need to be installed will be mentioned in this application note.

The details within this document will mainly reference the standard Linux kernel. Where possible, differences between the Linux “distributions” (for example, Ubuntu, Android) in terms of tools, directory locations or behavior will be noted.

1.1 SAMA5D3 Xplained Board

The platform used for testing and integration is the SAMA5D3 Xplained board. This SOC is a Cortex A5 ARM MPU. The pre-built demonstration Yocto project provides a basis for touch driver testing and integration.

In this application note, the demo project **linux4sam-poky-sam5d3_xplained_pda-5. 7** will be used. This can be found at the following link:

ftp://www. at91.com/pub/demo/linux4sam_5.7/linux4sam-poky-sama5d3_xplained_pda4-5.7.zip

NOTE At time of writing, the latest version of the demo project is version 6.0. However, a hardware or u-boot change is required to enable video to work. See the following locations for more details:

https://www.at91.com/linux4sam/bin/view/Linux4SAM/U-Boot#PDA_detection_at_boot

or

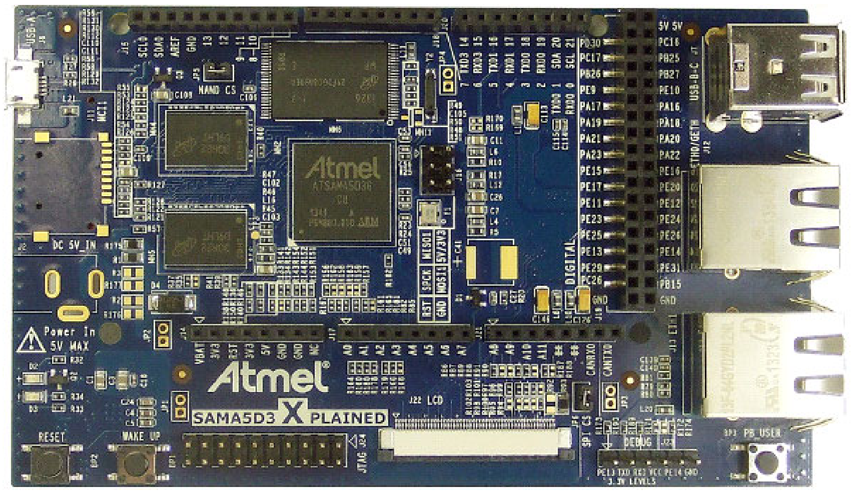
https://www.at91.com/linux4sam/bin/view/Linux4SAM/SelectingPDAatBoot

This project should be extracted to a local directory on the user’s PC. The content will be updated and used on the SAMA5D3 Xplained board to develop and test the touch driver. Note that only the compressed kernel image (**zImage**) and device tree binaries (**.dtb**) will be updated for testing; the **u-boot**, **at91bootstrap** and the Yocto project layer files will not be modified.

The following link contains additional information regarding the demo projects for the SAMA5D3 Xplained board and other supporting tools for the board:

https://www.at91.com/linux4sam/bin/view/Linux4SAM/Sama5d3XplainedMainPage

FIGURE 1: SAMA5D3 XPLAINED BOARD



1.2 SAM Boot Assistance (SAM-BA)

The SAM Boot Assistance (SAM-BA) in-system programmer is Microchip’s software for programming Microchip ARM Thumb-based MPUs. It is run on the Linux OS platform on the user’s PC to re-program the MPU on the SAMA5D3 Xplained board.

Follow the link below to download and extract the SAM-BA program to a directory.

https://github.com/atmelcorp/sam-ba/releases

1.3 Putty Terminal Program

To monitor activity during flashing and accessing the SAMA5D3 board, the Linux `putty` terminal application is used for serial communication. The `putty` terminal program is used throughout this application note to communicate to the driver and exercise the various features that are supported by the maXTouch devices.

The baud rate settings for serial communication are listed in Table 1.

TABLE 1: BAUD RATE SETTINGS

Parameter	Setting
Baud rate	115200
Data	8 bits
Parity	None
Stop	1 bit
Flow control	None

To install putty enter the following commands in the terminal window.

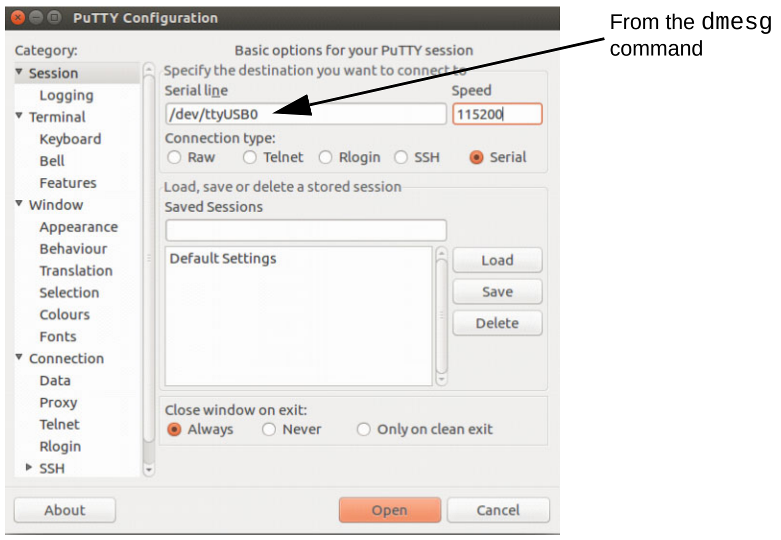
```
sudo apt-get update
sudo apt install putty
```

A USB to TTL serial cable is required to connect from the USB port of the PC to the J23 DBGU port of the SAMA5D3 evaluation board.

To find the USB port that is used for the USB to TTL serial cable, identify the USB connection by monitoring the last lines of the `dmesg` command when the cable is plugged into the local PC. This will provide the device name assigned by the Linux operating system. For example, in the `dmesg` listing below, the device `/dev/ttyUSB0` will be used to configure the terminal emulator (as in the green text below).

```
[167305. 231492] usb 1-5: new full-speed USB device number 16 using xhci_hcd
[167305. 386051] usb 1-5: New USB device found, idVendor=0403, idProduct=6001
[167305. 386057] usb 1-5: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[167305. 386061] usb 1-5: Product: FT232R USB UART
[167305. 386065] usb 1-5: Manufacturer: FTDI
[167305. 386069] usb 1-5: SerialNumber: A505MLX6
[167306. 762669] usbcore: registered new interface driver usbserial_generic
[167306. 762696] usbserial: USB Serial support registered for generic
[167306. 768859] usbcore: registered new interface driver ftdi_sio
[167306. 768867] usbserial: USB Serial support registered for FTDI USB Serial Device
[167306. 768915] ftdi_sio 1-5:1. 0: FTDI USB Serial Device converter detected
[167306. 768946] usb 1-5: Detected FT232RL
[167306. 769108] usb 1-5: FTDI USB Serial Device converter now attached to ttyUSB0
```

FIGURE 2: PUTTY CONFIGURATION



Specify the destination you want to connect to
Serial line: /dev/ttyUSB0
Speed: 115200
Connection type: ( ) Raw ( ) Telnet ( ) Rlogin ( ) SSH [✓ Serial]
Close window on exit: [✓ Always] ( ) Never ( ) Only on clean exit

1.4 Github Repository – maXTouch_linux

The information in this document references the **maXTouch_linux** Github repository. This is found at the following location:

https://github.com/atmel-maxtouch/maXTouch_linux

Within this repository, the user will find the releases for the maXTouch Linux driver. The latest released versions of the driver can be found at the following location:

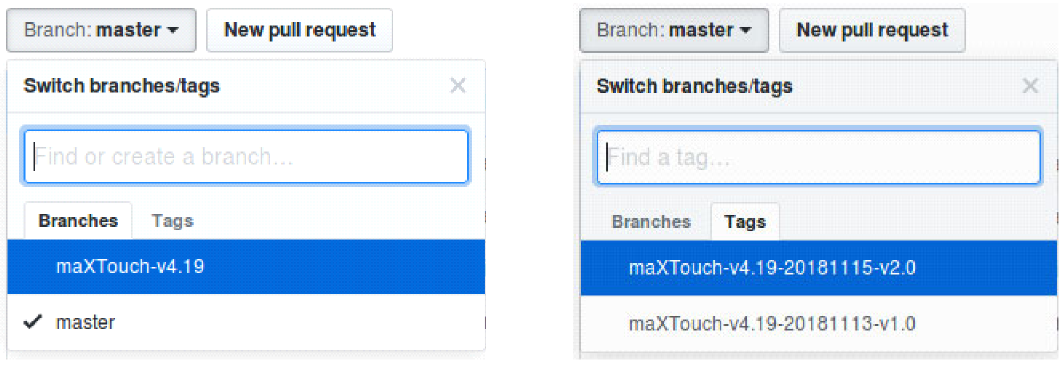
https://github.com/atmel-maxtouch/maXTouch_linux/releases

Under the Branch tab, the user will find the latest branches and tags listed for the repository:

- Branches are specifically for software development and may not have full features and functions that are mentioned in this document.

  The **master** branch contains the latest updated code changes to the repository. This branch should be used for any new driver development.
- Tags have regular releases of the touch driver with full change notes. All commits are available for all tags that are created in the repository.

FIGURE 3: BRANCHES AND TAGS – MASTER BRANCH



To download a working copy of the repository for testing, use the following command:

```
sudo git clone git@github.com:atmel-maxtouch/maXTouch_linux.git
```

2.0 BUILDING THE SOURCE AND SETTING UP THE TOOLS

The SAMA5D3 Xplained board was used to test the Linux driver. To enable the driver to work on the SAMA5D3 Xplained board, it is therefore necessary to modify various support files within the kernel. Modified versions of the kernel support files are located in the **maxTouch_Linux** Github repository and this section describes the modifications to these files.

2.1 Device Tree Files

The device tree source and include files describe the hardware platform being used and have been modified to provide support for the touch device. These files are included in the **maXTouch_linux** kernel repository (see Section 1.4 "Github Repository – maXTouch_linux"), located in the **arch/arm/boot/dts** directory:

- **at91-sama5d3_xplained_dm_pda4.dtsi –** Main device tree include file
- **at91-sama5d3_xplained_pda4.dts –** Main device tree source file

The following sections describe the modifications that have been made to these files.

2.1.1 AT91-SAMA5D3_XPLAINED_DM_PDA4.DTSI

The **at91-sama5d3_xplained_dm_pda4.dtsi** file has been modified to contain the following lines:

```
atmel_mxt_ts@4a {
    compatible = “atmel,atmel_mxt_ts”;
    reg = <0x4a>;
    reset-gpios = <&pioE 6 GPIO_ACTIVE_LOW>;
    interrupt-parent = <&pioE>;
    interrupts = <7 0x2>; /* Falling edge only */
    pinctrl-names = “default”;
    pinctrl-0 = <&pinctrl_mxt_ts>;
};
```

The device default address is 0x4A. Two of the SAMA5D3 GPIO lines are used for the maXTouch controller's $\overline{\text{RESET}}$ and $\overline{\text{CHG}}$ lines (GPIO 6 and GPIO 7 respectively). The $\overline{\text{RESET}}$ line is assigned to be active low and the $\overline{\text{CHG}}$ interrupt line is a falling edge-triggered GPIO.

2.1.2 AT91-SAMA5D3_XPLAINED_PDA4.DTS

The only modification to the **at91-sama5d3_xplained_pda4.dts** file is the addition of the device tree include file with the main modifications done above:

```
#include "at91-sama5d3_xplained_dm_pda4.dtsi"
```

2.2 Configuring the Kernel Options

Before the kernel can be built, the build options must be set up. These options are used during build time to enable the software features needed for the development platform and are defined in a **.config** file located in the root directory of the kernel.

The **.config** file is generated using one of the following methods:

- A `make` command applied to a **defconfig** file
- The `menuconfig` application, provides a menu-driven user interface to enable or disable options

The **defconfig** file is the Linux kernel definition file for specific architectures and systems. Different flavors of these files are located in the **arch/arm/configs/** directory. For the SAMA5D3 platform, the **sama5_defconfig** file is used.

The **.config** file is generated using the **defconfig** file by the following command:

```
make ARCH=arm sama5_defconfig
```

The **.config** file is automatically generated from the `make` command. The resulting **.config** file will contain the following two modified entries that allow touchscreen support to be enabled:

```
CONFIG_INPUT_TOUCHSCREEN=y
CONFIG_TOUCHSCREEN_ATMEL_MXT=y
```

Alternatively, the `menuconfig` application can be used to generate the **.config** file. To do this, the **Kconfig** file (located in the **drivers/input/touchscreen/** directory) must contain the `menuconfig` option that will be seen when the user runs the `menuconfig` application. In this case, the `config TOUCHSCREEN_ATMEL_MXT` entry in the **Kconfig** file defines a method that allows the user to chose whether the **atmel_mxt_ts** driver is included in the compiled image as built-in code or as a module:

```
config TOUCHSCREEN_ATMEL_MXT
    tristate "Atmel mXT I2C Touchscreen"
    depends on I2C
    select FW_LOADER
    help
      Say Y here if you have Atmel mXT series I2C touchscreen,
      such as AT42QT602240/ATMXT224, connected to your system.

      If unsure, say N.

      To compile this driver as a module, choose M here: the
      module will be called atmel_mxt_ts.
```

To run the `menuconfig` option to configure the kernel options, use the following command:

```
make ARCH=arm menuconfig
```

Lastly, to use the options in the **.config** file at build time to compile the maXTouch driver, the following entry is placed in the **Makefile** located in the **drivers/input/touchscreen/** directory:

```
obj-$(CONFIG_TOUCHSCREEN_ATMEL_MXT) += atmel_mxt_ts.o
```

2.3 Loading the Touch Driver as a Module

The touch driver can be configured to be built in with the kernel image or as a module to be loaded later. Note that if the maXTouch device configuration is to be loaded on a driver probe, the driver must be compiled as a module (see Section 3.4.2 "Automatically downloading the configuration on driver probe").

Before the kernel is built, the **defconfig** file should be modified to create a driver module (**atmel_mst_ts.ko**):

```
CONFIG_TOUCHSCREEN_ATMEL_MXT=m
```

The following commands can be used to ensure the correct permissions are set up for the driver module once it is installed:

```
chmod 0777 /sys/bus/i2c/drivers/atmel_mxt_ts/1-004a/debug_enable
chmod 0777 /sys/bus/i2c/drivers/atmel_mxt_ts/1-004a/mem_access
write /proc/sys/kernel/dmesg_restrict 0
```

The `chmod` and `dmesg_restrict` lines are used for debug access to the chip during integration and debugging. Permissions should be restored for production unless debug access is required at all times.

After the driver module is built, the user can install the module using the following command:

```
insmod /sys/lib/modules/atmel_mxt_ts.ko
```

2.4 Building the Kernel

Once the build options have been added to the **.config** file, the kernel can be built.

To build the kernel from source, an ARM cross compiler is needed. If a compiler has not been installed, the following command will install the `gcc` compiler:

```
sudo apt-get install gcc-arm-linux-gnueabi
```

To add a system environment variable for the `gcc` compiler to the PC, use the following command:

```
export CROSS_COMPILE=arm-linux-gnueabi-
```

To build the kernel, use the following command:

```
make ARCH=arm
```

This will build the kernel and place the resulting compiled kernel image (**zImage**) in the **arch/arm/boot** directory. A new device tree binary file will also be compiled and the resulting file (**at91-sama5d3_xplained_pda4.dtb**) placed in the **arch/arm/boot/dts** directory.

Replace the old **zImage** and the **.dtb** file in the demo project **linux4sam-poky-sam5d3_xplained_pda-5.7/** directory with the newly compiled files. To replace the files, use the following commands:

```
sudo cp arch/arm/boot/zImage ~/linux4sam-poky-sam5d3_xplained_pda-5.7/zImage-sama5d3-xplained.bin
sudo cp arch/arm/boot/dts/at91-sama5d3_xplained_pda4.dtb ~/linux4sam-poky-sama5d3_xplained_pda-5.7/
```

2.5 Modifying the U-boot-env.txt File

The U-boot environment binary file (**u-boot-env.bin**) contained in the **linux4sam-poky-sam5d3_xplained_pda-5.7/** directory holds the address ranges and the sizes of the various install files from the demo project. To adjust the starting addresses in the **u-boot-env.bin** file, you will need to create a shell script (called **create-env.sh** below).

Start by downloading the u-boot source code. A tool called `mkenvimage` will also be needed to create a new **u-boot-env.bin** file with the correct size for the new compiled **zImage**. The `mkenvimage` file is located in the **u-boot-at91/tools/** directory. Use the following commands to get the u-boot source and make the `mkenvimage` tool:

```
git clone git://github.com/linux4sam/u-boot-at91.git
make sama5d3_xplained_nandflash_defconfig
make
```

Now create the shell script file (**create-env.sh**) with the following content (where *user-dir* should be replaced by the appropriate user directory):

```
#!/bin/bash

MKENV=/home/user-dir/u-boot-at91/tools/mkenvimage
SAMBA=/home/user-dir/sam-ba_3.2.1/sam-ba

UBENV=/home/user-dir/linux4sam-poky-sama5d3_xplained_pda4-5.7/u-boot-env.bin

dtbName=/home/user-dir/linux4sam-poky-sama5d3_xplained_pda4-5.7/at91-sama5d3_xplained_pda4.dtb

dtbLoadAddr=0x21000000
dtbAddr=0x180000
dtbSize=$(wc -c $dtbName | cut -d ' ' -f 1)

kernelName=/home/user-dir/linux4sam-poky-sama5d3_xplained_pda4-5.7/zImage

kernelLoadAddr=0x22000000
kernelAddr=0x200000
kernelSize=$(wc -c $kernelName | cut -d ' ' -f 1)

# nand 0x20000, serial flash 0x2000, mmc raw 0x1000 (at91-sama5_common.h)

envSize=0x20000

rm -f $UBENV

$MKENV -p 0 -s $envSize -o $UBENV << EOF

bootdelay=1

baudrate=115200

bootargs=console=ttyS0,115200
mtdparts=atmel_nand:256k(bootstrap)ro,768k(uboot)ro,256K(env_redundant),256k(env),512k(dtb),
    6M(kernel)ro,-(rootfs) rootfstype=ubifs ubi.mtd=6 root=ubi0:rootfs

bootcmd=nand read $dtbLoadAddr $dtbAddr $dtbSize; nand read $kernelLoadAddr $kernelAddr
    $kernelSize; bootz $kernelLoadAddr - $dtbLoadAddr

ethact=gmac0

stdin=serial

stdout=serial

stderr=serial

EOF
```

Set the permissions for the shell script:

```
chmod 777 create-env.sh
```

Finally run the shell script to create a new **u-boot-env.bin** file for the newly compiled **zImage** in the demo project folder:

```
./create-env.sh
```

2.6 Programming the SAMA5D3 Xplained Board

Programming the SAMA5D3 Xplained board requires the use of the SAM-BA application (see Section 1.2 “SAM Boot Assistance (SAM-BA)”). You will also need a USB cable to connect the PC to the SAMA5D3 Xplained board.

To program the SAMA5D3 Xplained board:

1. Open the “Boot Dis” jumper, JP9 (NAND Flash Chip Select).
2. Power on the board while holding down the RESET button.
3. Release the RESET button.
4. Finally close the “Boot Dis” jumper.

Issue the following command from the **linux4sam-poky-sama5d3_xplained_pda4-5.7** directory:

```
sudo ~/samba_3.2.2/samba –x demo_linux_nandflash-usb.qml
```

The serial terminal will show the status of the NAND Flash while in programming mode.

When the flashing process has completed, power cycle the board.

3.0 MAXTOUCH LINUX DRIVER

3.1 I^{2}C Driver – atmel_mxt_ts.c

The Linux mainline kernel (www.kernel.org) contains a driver for maXTouch chips that use I^{2}C-based communication. The driver was originally added in kernel version 2.6.36 and was called `qt602240`**.** It was later renamed to `atmel_mxt_ts` in version 2.6.39 onwards.

The `atmel_mxt_ts` driver is still present in the current v4.19 mainline Linux kernel and is located at:

https://github.com/torvalds/linux/blob/master/drivers/input/touchscreen/atmel_mxt_ts.c

The main purpose of the maXTouch Linux driver is to read touch events from the input device and transfer the data to the Linux input subsystem. Event handlers distribute the events from the device, typically to userspace applications that use the information to determine the location of the touch.

Events are generated when a user touches the touchscreen and interrupts are sent to the MPU. These interrupts are received by the touch driver, which then initiates a read from the input device. Events can come from a finger, stylus, glove or even a hovering object.

For debugging purposes, the input touch driver uses the `sysfs` filesystem to provide an interface to the kernel by creating `sysfs` files within userspace. These files can be used to support firmware or configuration updates, and also to provide information about the input device.

There have been many improvements and changes to the driver to fix bugs and improve support for the various maXTouch devices. Over time, the mainline kernel touch driver has been simplified and support for the `mxt-app` userspace tool has been removed from the mainline kernel. The primary focus of development for the maXTouch Linux repository is therefore to restore support for `mxt-app` and continue to enhance the touch driver for new products.

3.2 Object-based Protocol

The control interface to all maXTouch chips conforms to a standard system known as the Object-based Protocol (OBP), which defines how to interact with the various objects that are implemented on any particular chip. It defines a mapping between registers and configuration values, and a way for objects to send status update messages to the host. For a full description of the OBP for a particular device and firmware version, refer to the *Protocol Guide* issued for that device and version.

3.3 Power Up and Reset

The driver controls touch acquisition by altering parameters in the Power Configuration T7 object.

When the Linux OS starts the driver probe process to pair hardware components with their supporting drivers, the running configuration in the maXTouch device is cached.

When the system is stopped, a zero configuration is written to the active and idle times parameters. This puts the maXTouch device into a deep sleep mode.

During driver probe, a reset of approximately 100 ms is performed. Note that the maXTouch device's $\overline{\text{RESET}}$ line is active low, so a `gpiod_set_value` system call of `1` to the `reset_gpio` variable will set the $\overline{\text{RESET}}$ line low. The driver code is as follows:

```
#define MXT_RESET_INVALID_CHG 100 /* msec */

if(!(IS_ERR(data->reset_gpio))) {
        dev_info(&client->dev, "Resetting chip\n");
        msleep(MXT_RESET_GPIO_TIME);
        gpiod_set_value(data->reset_gpio, 1);
        msleep(MXT_RESET_INVALID_CHG);
        gpiod_set_value(data->reset_gpio, 0);
    }
```

3.4 Downloading the Configuration

The maXTouch device contains a configuration which is stored in the device's non-volatile memory (NVM). Using the maXTouch Studio application, users can create a configuration optimized for their target sensor.

The `mxt-app` application and maXTouch Studio tools can save configuration files in two formats: extended configuration format and RAW format. Configurations should be saved in RAW format when using the Linux driver.

An example of the header from a **.raw** file is as follows:

```
OBP_RAW V1
A4 03 23 AA 00 00 00
FCDAB2
C05DF0
```

Note that the first line of the **.raw** file contains the string “`OBP_RAW V1`” to indicate that the configuration was generated in the RAW format.

There are several methods to update the configuration of the device:

- Through the `mxt-app` userspace application
- Triggered from the `sysfs` attribute “`update_cfg`”
- Automatically, upon driver probe

3.4.1 TRIGGERING CONFIGURATION DOWNLOAD WITH SYSFS

The following command downloads the configuration to the device and backs it up to the NVM. Note that the configuration file is named **maxtouch.cfg.**

```
echo maxtouch.cfg > /sys/bus/i2c/drivers/atmel_mxt_ts/1-004a/update_cfg
```

On Linux systems, the file should be placed in the **/lib/firmware** directory.

On Android systems, the file should be placed in the **/system/vendor/firmware** directory.

The following is an example of the configuration download messages:

```
echo maxtouch.cfg > /sys/bus/i2c/drivers/atmel_mxt_ts/1-004a/update_cfg
atmel_mxt_ts 1-004a: Found configuration file: maxtouch.cfg
atmel_mxt_ts 1-004a: Config CRC 0x4FD6D1: does not match file 0x5FD6D1
atmel_mxt_ts 1-004a: Config CRC in file inconsistent, calculated=4FD6D1, file=5FD6D1
atmel_mxt_ts 1-004a: Resetting device
atmel_mxt_ts 1-004a: Config successfully updated
```

3.4.2 AUTOMATICALLY DOWNLOADING THE CONFIGURATION ON DRIVER PROBE

The configuration can be loaded on driver probe. For production devices it is desirable to have the maXTouch driver check the configuration on every boot and upload it if necessary. This guards against problems caused by the configuration being changed accidentally and allows the initial configuration to be set when the firmware is updated.

To ensure that the firmware and configuration can be loaded during driver probe, The driver must be compiled as a module. The reason for this is that the system call for loading firmware and configuration is not available until the root file system is booted up. If the driver is built-in, therefore, it is not possible to load configuration and firmware during boot-up as the root file system is not ready on driver probe. If the driver is built as a module, however, it will be installed after the SAMA5D3 board is powered up and the root file system is loaded.

The following error message is the one that is typically seen on boot up in the `dmesg` dump:

```
Direct firmware load for maxtouch.cfg failed with error -2
```

This message can be ignored if the touch driver is built-in.

3.5 Firmware Upgrade

A maXTouch device may support different firmware versions. Changes between firmware versions may result in removal or addition of individual configuration settings and entire objects, which alters the configuration CRC. This means that when the firmware is upgraded, a new configuration is normally needed to be loaded into the chip. The kernel driver will attempt to reload the configuration automatically.

The default “enc” firmware files are in ASCII HEX and must be converted into raw format:

```
xxd -r -p firmware.enc > /lib/firmware/maxtouch.fw
```

The file must be placed in the firmware loader path similar to the **maxtouch.cfg** file (see Section 3.4 “Downloading the Configuration”).

Next issue the following command to trigger the firmware update through the `sysfs` attribute:

```
echo maxtouch.fw > /sys/bus/i2c/drivers/atmel_mxt_ts/1-004a/update_fw
```

The following is an example of the firmware update messages:

```
echo maxtouch.fw > /sys/bin/bus/i2c/drivers/atmel_mxt_ts/1-004a/update_fw
atmel_mxt_ts 1-004a: Opened firmware file: maxtouch.fw
atmel_mxt_ts 1-004a: File format is okay
atmel_mxt_ts 1-004a: Sent bootloader command.
atmel_mxt_ts 1-004a: Bootloader address: 26
atmel_mxt_ts 1-004a: Found bootloader I2C address
atmel_mxt_ts 1-004a: __mxt_write_reg: i2c send failed (-121)
atmel_mxt_ts 1-004a: Unlocking bootloader
atmel_mxt_ts 1-004a: Sent 50 frames, 8024/150868 bytes
atmel_mxt_ts 1-004a: Sent 100 frames, 15824/150868 bytes
atmel_mxt_ts 1-004a: Sent 150 frames, 23624/150868 bytes
atmel_mxt_ts 1-004a: Sent 200 frames, 31424/150868 bytes
atmel_mxt_ts 1-004a: Sent 250 frames, 39224/150868 bytes
atmel_mxt_ts 1-004a: Sent 300 frames, 47024/150868 bytes
atmel_mxt_ts 1-004a: Sent 350 frames, 54824/150868 bytes
atmel_mxt_ts 1-004a: Sent 400 frames, 62624/150868 bytes
atmel_mxt_ts 1-004a: Sent 450 frames, 70424/150868 bytes
atmel_mxt_ts 1-004a: Sent 500 frames, 78224/150868 bytes
atmel_mxt_ts 1-004a: Sent 550 frames, 86024/150868 bytes
atmel_mxt_ts 1-004a: Sent 600 frames, 93824/150868 bytes
atmel_mxt_ts 1-004a: Sent 650 frames, 101624/150868 bytes
atmel_mxt_ts 1-004a: Sent 700 frames, 109424/150868 bytes
atmel_mxt_ts 1-004a: Sent 750 frames, 117224/150868 bytes
atmel_mxt_ts 1-004a: Sent 800 frames, 125024/150868 bytes
atmel_mxt_ts 1-004a: Sent 850 frames, 132824/150868 bytes
atmel_mxt_ts 1-004a: Sent 900 frames, 140608/150868 bytes
atmel_mxt_ts 1-004a: Sent 950 frames, 148408/150868 bytes
atmel_mxt_ts 1-004a: Sent 965 frames, 150868 bytes
atmel_mxt_ts 1-004a: The firmware update succeeded
atmel_mxt_ts 1-004a: Family: 164 Variant: 21 Firmware V2.3.AA Objects: 39
atmel_mxt_ts 1-004a: Config CRC 0x4FD6D1: does not match file 0x09B259
atmel_mxt_ts 1-004a: Resetting device
atmel_mxt_ts 1-004a: Config successfully updated
atmel_mxt_ts 1-004a: Touchscreen size X1023Y1023
```

The `__mxt_write_reg: i2c send failed (-121)` message can be ignored. This message occurs because an interrupt is generated from the device that causes an additional read at the 0x4A device address after the chip has switched over to the bootloader address.

After the flash, the configuration is written if the **maxtouch.cfg** file is found in the same firmware loader path.

4.0 DEBUGGING THE DRIVER

There are various system level ways of debugging the touch driver. These are discussed below.

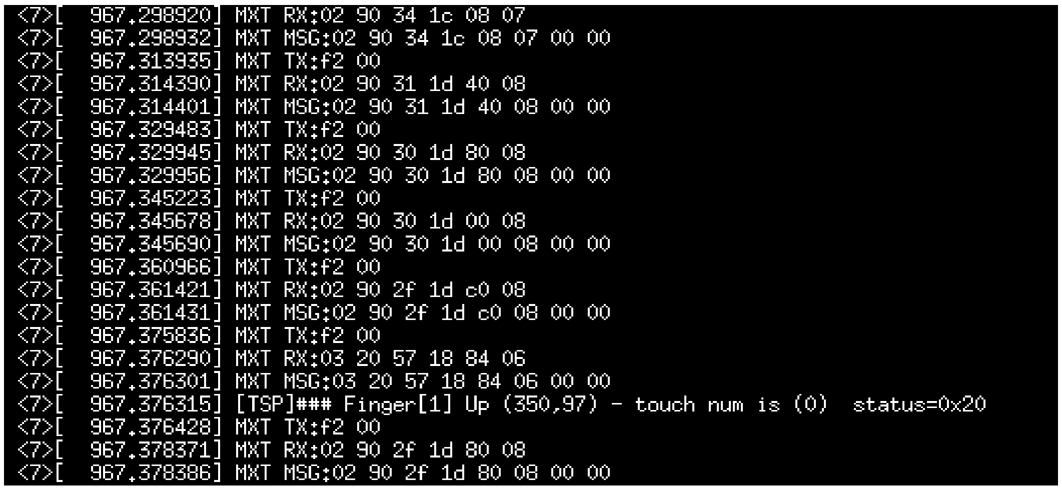
4.1 Enabling Driver Debug Messages – dev_dbg

One mechanism is to enable the `dev_dbg` messages that are coded into the driver to help understand the logic flow. To enable debug message support, the following line of code needs to be added to the first line of the maXTouch driver.

```
#define DEBUG /*  Can be added to the driver to get dev_dbg messages, first line  */
```

NOTE	This debug statement should be removed for production.

FIGURE 4: DRIVER DEBUG MESSAGES



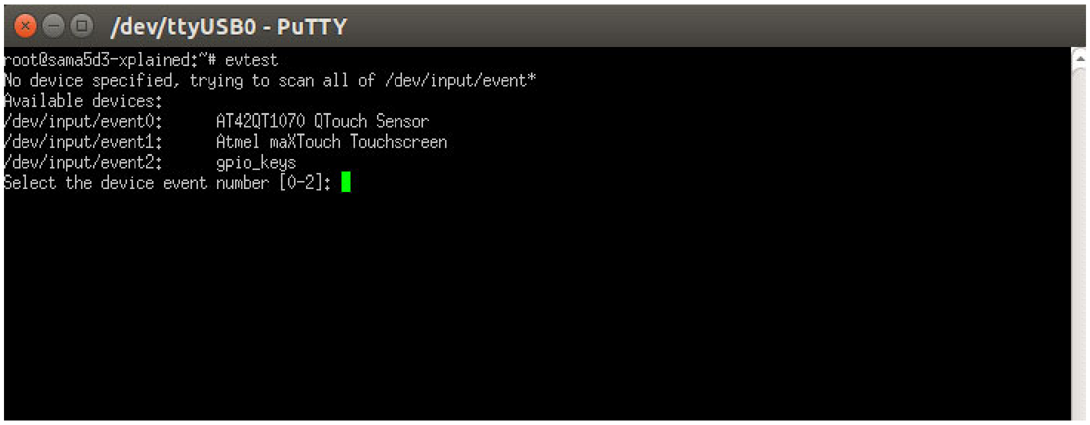
```
<7>[  967.298920] MXT RX:02 90 34 1c 08 07
<7>[  967.298932] MXT MSG:02 90 34 1c 08 07 00 00
<7>[  967.313935] MXT TX:f2 00
<7>[  967.314390] MXT RX:02 90 31 1d 40 08
<7>[  967.314401] MXT MSG:02 90 31 1d 40 08 00 00
<7>[  967.329483] MXT TX:f2 00
<7>[  967.329945] MXT RX:02 90 30 1d 80 08
<7>[  967.329956] MXT MSG:02 90 30 1d 80 08 00 00
<7>[  967.345223] MXT TX:f2 00
<7>[  967.345678] MXT RX:02 90 30 1d 00 08
<7>[  967.345690] MXT MSG:02 90 30 1d 00 08 00 00
<7>[  967.360966] MXT TX:f2 00
<7>[  967.361421] MXT RX:02 90 2f 1d c0 08
<7>[  967.361431] MXT MSG:02 90 2f 1d c0 08 00 00
<7>[  967.375836] MXT TX:f2 00
<7>[  967.376290] MXT RX:03 20 57 18 84 06
<7>[  967.376301] MXT MSG:03 20 57 18 84 06 00 00
<7>[  967.376315] [TSP]### Finger[1] Up (350,97) - touch num is (0)  status=0x20
<7>[  967.376428] MXT TX:f2 00
<7>[  967.378371] MXT RX:02 90 2f 1d 80 08
<7>[  967.378386] MXT MSG:02 90 2f 1d 80 08 00 00
```

4.2 Viewing Touch Events – evtest

The `evtest` tool can display input event information that is reported from the maXTouch driver. The tool can also display all of the touch events that are supported by the device.

By typing `evtest`, the user will see the available input devices, along with a listing of the event numbers assigned to them.

FIGURE 5:

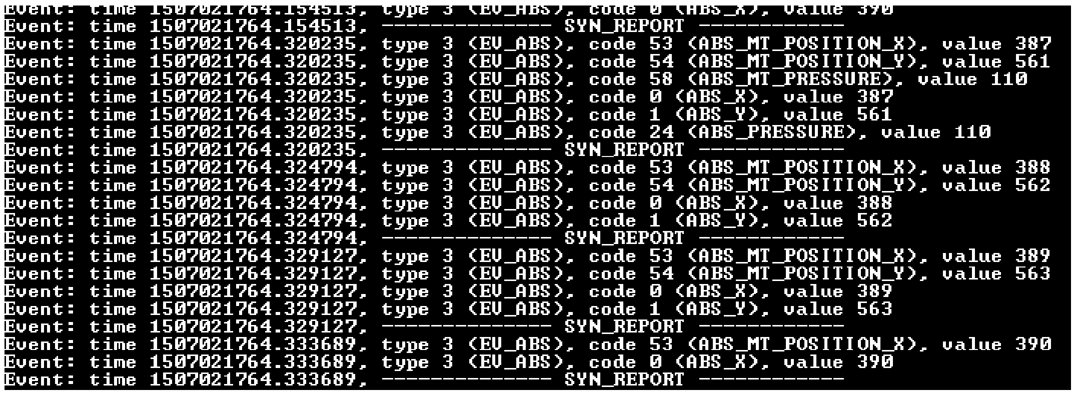


```
root@sama5d3-xplained:~# evtest
No device specified, trying to scan all of /dev/input/event*
Available devices:
/dev/input/event0:      AT42QT1070 QTouch Sensor
/dev/input/event1:      Atmel maXTouch Touchscreen
/dev/input/event2:      gpio_keys
Select the device event number [0-2]:
```

When touching the screen, the monitor will list out all the events that occur, including events for press, move, and release (see Figure 6). For a list of event codes, see the following location:

https://www.kernel.org/doc/html/latest/input//event-codes.html

FIGURE 6: TOUCH EVENT CODES WHEN FINGER IS PRESENT ON THE SENSOR



```
Event: time 1507021764.154513, type 3 (EV_ABS), code 0 (ABS_X), value 390
Event: time 1507021764.154513, -------------- SYN_REPORT ------------
Event: time 1507021764.320235, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 387
Event: time 1507021764.320235, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 561
Event: time 1507021764.320235, type 3 (EV_ABS), code 58 (ABS_MT_PRESSURE), value 110
Event: time 1507021764.320235, type 3 (EV_ABS), code 0 (ABS_X), value 387
Event: time 1507021764.320235, type 3 (EV_ABS), code 1 (ABS_Y), value 561
Event: time 1507021764.320235, type 3 (EV_ABS), code 24 (ABS_PRESSURE), value 110
Event: time 1507021764.320235, -------------- SYN_REPORT ------------
Event: time 1507021764.324794, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 388
Event: time 1507021764.324794, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 562
Event: time 1507021764.324794, type 3 (EV_ABS), code 0 (ABS_X), value 388
Event: time 1507021764.324794, type 3 (EV_ABS), code 1 (ABS_Y), value 562
Event: time 1507021764.324794, -------------- SYN_REPORT ------------
Event: time 1507021764.329127, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 389
Event: time 1507021764.329127, type 3 (EV_ABS), code 54 (ABS_MT_POSITION_Y), value 563
Event: time 1507021764.329127, type 3 (EV_ABS), code 0 (ABS_X), value 389
Event: time 1507021764.329127, type 3 (EV_ABS), code 1 (ABS_Y), value 563
Event: time 1507021764.329127, -------------- SYN_REPORT ------------
Event: time 1507021764.333689, type 3 (EV_ABS), code 53 (ABS_MT_POSITION_X), value 390
Event: time 1507021764.333689, type 3 (EV_ABS), code 0 (ABS_X), value 390
Event: time 1507021764.333689, -------------- SYN_REPORT ------------
```

4.3 Viewing Touch Events – Android getevent

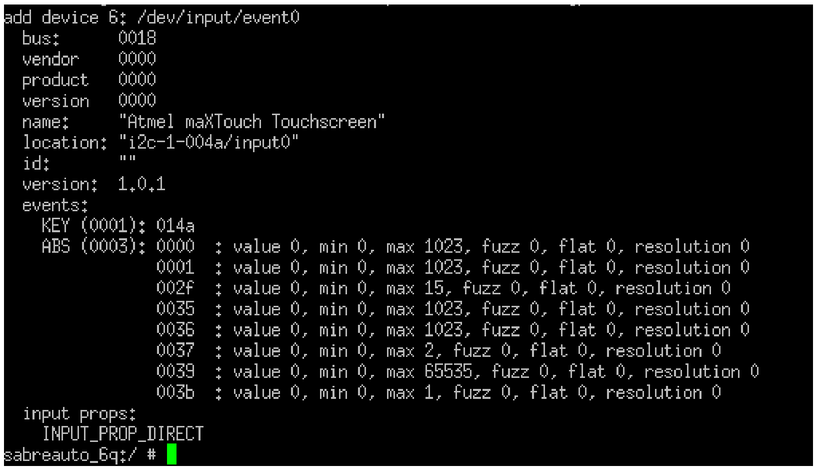
The Android `getevent` tool runs on the device and provides input event information that is reported from the kernel. You can find more information regarding `getevent` at the following location:

https://source.android.com/devices/input/getevent

Issuing the following command will display the device information in the terminal window (see Figure 7).

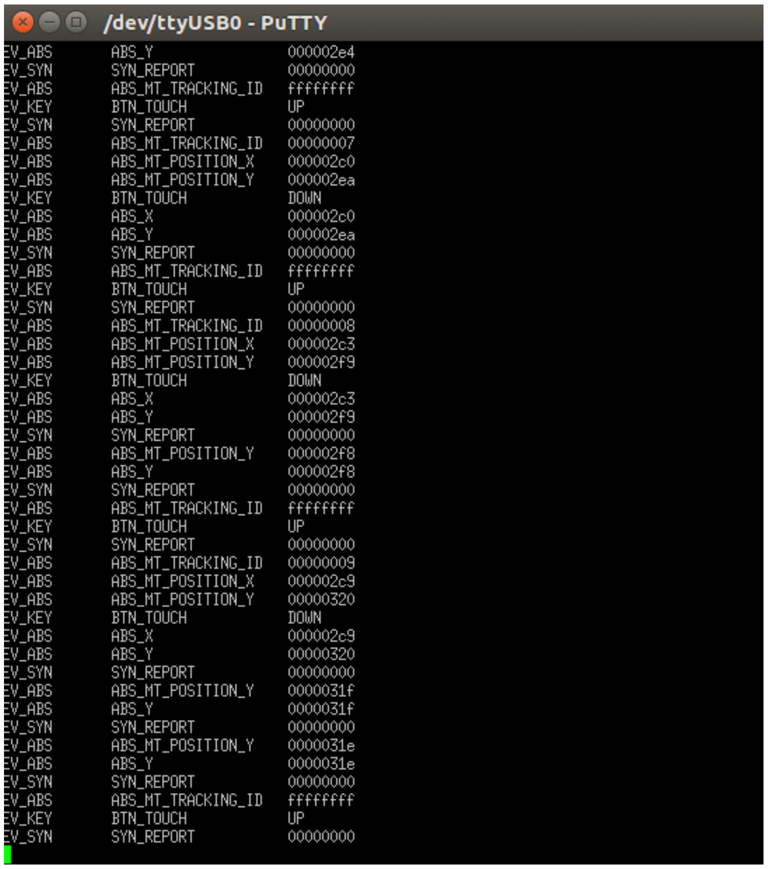
```
getevent -i
```

FIGURE 7: GETEVENT DEVICE INFORMATION



```
add device 6: /dev/input/event0
  bus:      0018
  vendor    0000
  product   0000
  version   0000
  name:     "Atmel maXTouch Touchscreen"
  location: "i2c-1-004a/input0"
  id:       ""
  version:  1.0.1
  events:
    KEY (0001): 014a
    ABS (0003): 0000  : value 0, min 0, max 1023, fuzz 0, flat 0, resolution 0
                0001  : value 0, min 0, max 1023, fuzz 0, flat 0, resolution 0
                002f  : value 0, min 0, max 15, fuzz 0, flat 0, resolution 0
                0035  : value 0, min 0, max 1023, fuzz 0, flat 0, resolution 0
                0036  : value 0, min 0, max 1023, fuzz 0, flat 0, resolution 0
                0037  : value 0, min 0, max 2, fuzz 0, flat 0, resolution 0
                0039  : value 0, min 0, max 65535, fuzz 0, flat 0, resolution 0
                003b  : value 0, min 0, max 1, fuzz 0, flat 0, resolution 0
  input props:
    INPUT_PROP_DIRECT
sabreauto_6q:/ #
```

FIGURE 8: GETEVENT MESSAGES



```
/dev/ttyUSB0 - PuTTY
EV_ABS       ABS_Y                000002e4
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_TRACKING_ID   ffffffff
EV_KEY       BTN_TOUCH            UP
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_TRACKING_ID   00000007
EV_ABS       ABS_MT_POSITION_X    000002c0
EV_ABS       ABS_MT_POSITION_Y    000002ea
EV_KEY       BTN_TOUCH            DOWN
EV_ABS       ABS_X                000002c0
EV_ABS       ABS_Y                000002ea
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_TRACKING_ID   ffffffff
EV_KEY       BTN_TOUCH            UP
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_TRACKING_ID   00000008
EV_ABS       ABS_MT_POSITION_X    000002c3
EV_ABS       ABS_MT_POSITION_Y    000002f9
EV_KEY       BTN_TOUCH            DOWN
EV_ABS       ABS_X                000002c3
EV_ABS       ABS_Y                000002f9
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_POSITION_Y    000002f8
EV_ABS       ABS_Y                000002f8
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_TRACKING_ID   ffffffff
EV_KEY       BTN_TOUCH            UP
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_TRACKING_ID   00000009
EV_ABS       ABS_MT_POSITION_X    000002c9
EV_ABS       ABS_MT_POSITION_Y    00000320
EV_KEY       BTN_TOUCH            DOWN
EV_ABS       ABS_X                000002c9
EV_ABS       ABS_Y                00000320
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_POSITION_Y    0000031f
EV_ABS       ABS_Y                0000031f
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_POSITION_Y    0000031e
EV_ABS       ABS_Y                0000031e
EV_SYN       SYN_REPORT           00000000
EV_ABS       ABS_MT_TRACKING_ID   ffffffff
EV_KEY       BTN_TOUCH            UP
EV_SYN       SYN_REPORT           00000000
```

4.4 Debug File System Interface – sysfs

The `sysfs` interface is a virtual file system for exporting kernel objects and is a ram-based file system based on `ramfs`. It provides a means to export kernel data structures, their attributes, and the linkages between them to userspace.

Attributes are created within the driver for direct access to a firmware flash or the configuration (for example, `update_fw`, `update_cfg`). Other attributes are created to access information obtained by the driver (for example, `fw_version`, `hw_version`, `config_crc`). Store and show functions are created to allow write and read access to the attributes.

When the `sysfs` attributes are created, they appear as files in the **/sys** directory. The path is as follows:

```
cd /sys/bus/i2c/drivers/ddddd/b-00xx/
```

where:

ddddd = Name of the driver. For instance, "atmel_mxt_ts", "sec_touch", "maXTouch", etc.
b = I^{2}C Adapter (value “1”)
00xx = I^{2}C Address

For example:

```
cd /sys/bus/i2c/drivers/atmel_mxt_ts/1-004a/
```

The attributes created by the driver are as follows:

- `fw_version` – Gets current firmware version of the device
- `hw_version` – Gets the maXTouch device’s family ID and the variant ID
- `update_cfg` – Used to update the configuration
- `config_crc` – Used to get current configuration CRC from the device
- `debug_enable`, `debug_v2_enable`, `debug_notify` – Attributes used to enable debug messages to be polled from the `dmesg` (driver message) buffer or the maXTouch device Message Processor T5 object.

- `mem_access` – Gives access to the I^{2}C address space for **`mxt-app`** operation and config loading
- `mxt_debug_msg` – Enables access to the Message Processor T5 messages from the device

4.4.1 MEM_ACCESS

The `mem_access` binary attribute provides read and write access to the touch device. It is used for loading the firmware and configuration, as well as, provides access to the device when using the `mxt-app` application. The permissions to this attribute need to be set to `777` to allow data to be buffered between the device and the `mxt-app` application.

4.4.2 DEBUG_ENABLE

The `debug_enable` attribute allows the user to turn on messages that would directly come from the Message Processor T5 object. These messages can be enabled by echoing a `"1"` to the `debug_enable` attribute.

```
echo “1” > /sys/bus/i2c/drivers/atmel_mxt_ts/1-004a/debug_enable

atmel_mxt_ts 1-004a: debug enabled
atmel_mxt_ts 1-004a: message: 31 94 90 02 b2 01 00 00 00 00
atmel_mxt_ts 1-004a: message: 31 91 91 02 b2 01 00 00 00 00
atmel_mxt_ts 1-004a: message: 31 91 92 02 b2 01 00 00 00 00
atmel_mxt_ts 1-004a: message: 31 91 93 02 b2 01 00 00 00 00
```

4.5 Improved Debug Interface

An enhanced debug interface is supported in the driver for message debug output. This consists of the following `sysfs` attributes:

```
debug_v2_enable
debug_notify
mxt_debug_msg (binary attribute)
```

The interface is enabled using `debug_v2_enable`.

The interface does not use `dmesg`. Instead, messages can be read from the `mxt_debug_msg` binary attribute. The `debug_notify` attribute can be polled synchronously to determine if there are more messages.

4.6 The mxt-app Utility

The `mxt-app` utility allows the user to perform certain operations from the command line and through a menu of select options. The source for the `mxt-app` and information on how to build the application is available at the following location:

https://github.com/atmel-maxtouch/mxt-app

When run without options `mxt-app` provides a menu from which the user can choose from the following options:

- Load a configuration file (both **.raw** and **.xcfg** format)
- Save configuration file
- Display the info block
- Read object configuration settings
- Write object configuration settings
- Run self tests using the Self Test T25 object
- Flash firmware to the chip (in **.enc** format)
- Backup configuration data to NVRAM
- Reset the device
- Calibrate the device
- Display raw messages
- Dump diagnostic data

Additional commands are available when running the utility on the command line. These are listed below.

The `mxt-app` command line features also allow scripts to be created for automated testing.

```
Command line tool for Atmel maXTouch chips version: 1.28-5-gd8202e1
Usage: ./mxt-app [command] [options]
When run with no options, access menu interface.

General commands:
  -h [--help]                 : display this help and exit
  -i [--info]                 : print device information
  -M [--messages] [TIMEOUT]   : print the messages (for TIMEOUT seconds)
  -F [--msg-filter] TYPE      : message filtering by object TYPE
  --reset                     : reset device
  --reset-bootloader          : reset device in bootloader mode
  --calibrate                 : send calibrate command
  --backup[=COMMAND]          : backup configuration to NVRAM
  -g                          : store golden references
  --self-cap-tune-config      : tune self capacitance settings to config
  --self-cap-tune-nvram       : tune self capacitance settings to NVRAM
  --version                   : print version
  --block-size BLOCKSIZE      : set the maximum block size used for i2c transfers (default 255)

Configuration file commands:
  --load FILE                 : upload cfg from FILE in .xcfg or OBP_RAW format
  --save FILE                 : save cfg to FILE in .xcfg or OBP_RAW format
  --checksum FILE             : verify .xcfg or OBP_RAW file config checksum

Register read/write commands:
  -R [--read]                 : read from object
  -W [--write]                : write to object
  -n [--count] COUNT          : read/write COUNT registers
  -f [--format]               : format register output
  -I [--instance] INSTANCE    : select object INSTANCE
  -r [--register] REGISTER    : start at REGISTER (offset when used with TYPE)
  -T [--type] TYPE            : select object TYPE
  --zero                      : zero all configuration settings

TCP socket commands:
  -C [--bridge-client] HOST   : connect over TCP to HOST
  -S [--bridge-server]        : start TCP socket server
  -p [--port] PORT            : TCP port (default 4000)

Bootloader commands:
  --bootloader-version        : query bootloader version
  --flash FIRMWARE            : send FIRMWARE to bootloader
  --firmware-version VERSION  : check firmware VERSION before and after flash

T68 Serial Data commands:
  --t68-file FILE             : upload FILE
  --t68-datatype DATATYPE     : select DATATYPE

T25 Self Test commands:
  -t [--test]                 : run all self tests
  -tXX [--test=XX]            : run individual test, write XX to CMD register

T37 Diagnostic Data commands:
  --debug-dump FILE           : capture diagnostic data to FILE
  --frames N                  : capture N frames of data
  --references                : capture references data
  --self-cap-signals          : capture self cap signals
  --self-cap-deltas           : capture self cap deltas
  --self-cap-refs             : capture self cap references
  --active-stylus-deltas      : capture active stylus deltas
  --active-stylus-refs        : capture active stylus references
```

```
Broken line detection commands:
  --broken-line                : run broken line detection
  --dualx                      : X lines are double connected
  --x-center-threshold N       : set X line center threshold to N percent
  --x-border-threshold N       : set X line border threshold to N percent
  --y-center-threshold N       : set Y line center threshold to N percent
  --y-border-threshold N       : set Y line border threshold to N percent
  --pattern PATTERN            : sensor PATTERN (ITO or XSense)

Sensor Variant algorithm commands:
  --sensor-variant             : Perform the Sensor Variant algorithm
  --dualx                      : X lines are double connected
  --fail-if-any                : Fail the Sensor Variant test on any defects
  --max-defects N              : Maximum No. of continuious defects
  --upper-limit N              : Upper limit for regression, in %
  --lower-limit N              : Lower limit for regression, in %
  --matrix-size N              : The allowed matrix size

Device connection options:
  -q [--query]                 : scan for devices
  -d [--device] DEVICESTRING   : DEVICESTRING as output by --query

Examples:
    -d i2c-dev:ADAPTER:ADDRESS : raw i2c device, eg "i2c-dev:2-004a"
    -d sysfs:PATH              : sysfs interface
    -d hidraw:PATH             : HIDRAW device, eg "hidraw:/dev/hidraw0"

Debug options:
  -v [--verbose] LEVEL         : set debug level
```

APPENDIX A: REVISION HISTORY

Revision A (May 2019)

Initial version

ISBN: 978-1-5224-4496-1

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://www.microchip.com/support
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453
Tel: 317-536-2380

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608
Tel: 951-273-7800

Raleigh, NC
Tel: 919-844-7510

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110
Tel: 408-436-4270

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Australia - Sydney
Tel: 61-2-9868-6733

China - Beijing
Tel: 86-10-8569-7000

China - Chengdu
Tel: 86-28-8665-5511

China - Chongqing
Tel: 86-23-8980-9588

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115

China - Hong Kong SAR
Tel: 852-2943-5100

China - Nanjing
Tel: 86-25-8473-2460

China - Qingdao
Tel: 86-532-8502-7355

China - Shanghai
Tel: 86-21-3326-8000

China - Shenyang
Tel: 86-24-2334-2829

China - Shenzhen
Tel: 86-755-8864-2200

China - Suzhou
Tel: 86-186-6233-1526

China - Wuhan
Tel: 86-27-5980-5300

China - Xian
Tel: 86-29-8833-7252

China - Xiamen
Tel: 86-592-2388138

China - Zhuhai
Tel: 86-756-3210040

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444

India - New Delhi
Tel: 91-11-4160-8631

India - Pune
Tel: 91-20-4121-0141

Japan - Osaka
Tel: 81-6-6152-7160

Japan - Tokyo
Tel: 81-3-6880- 3770

Korea - Daegu
Tel: 82-53-744-4301

Korea - Seoul
Tel: 82-2-554-7200

Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

Malaysia - Penang
Tel: 60-4-227-8870

Philippines - Manila
Tel: 63-2-634-9065

Singapore
Tel: 65-6334-8870

Taiwan - Hsin Chu
Tel: 886-3-577-8366

Taiwan - Kaohsiung
Tel: 886-7-213-7830

Taiwan - Taipei
Tel: 886-2-2508-8600

Thailand - Bangkok
Tel: 66-2-694-1351

Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

Finland - Espoo
Tel: 358-9-4520-820

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Garching
Tel: 49-8931-9700

Germany - Haan
Tel: 49-2129-3766400

Germany - Heilbronn
Tel: 49-7131-67-3636

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Germany - Rosenheim
Tel: 49-8031-354-560

Israel - Ra'anana
Tel: 972-9-744-7705

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Padova
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Norway - Trondheim
Tel: 47-7289-4388

Poland - Warsaw
Tel: 48-22-3325737

Romania - Bucharest
Tel: 40-21-407-87-50

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Gothenberg
Tel: 46-31-704-60-40

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820