REDS
Reconfigurable & embedded Digital Systems

heig-vd
Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Architecture and Drivers for Smartphones
Input Devices

Cours SAS

Salvatore Valenza

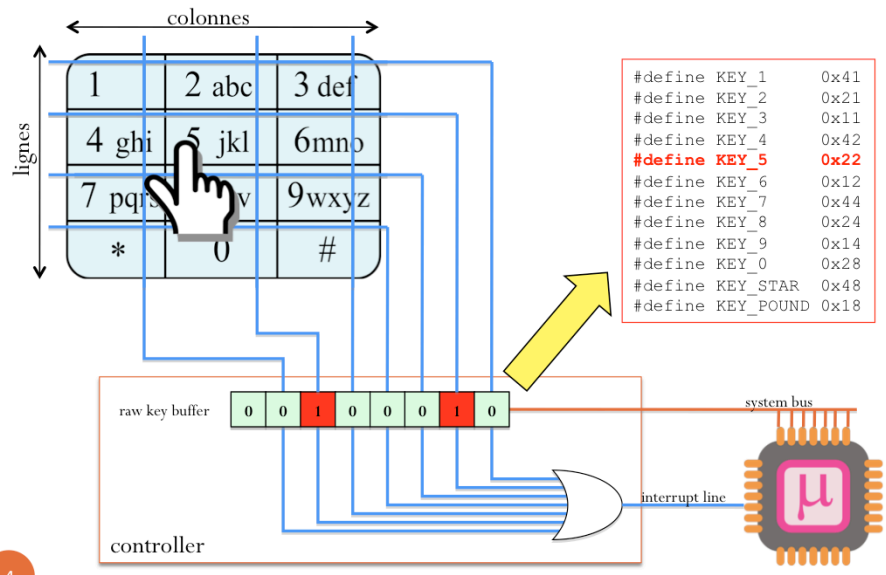
Version 1.0 (2012-2013)

Plan

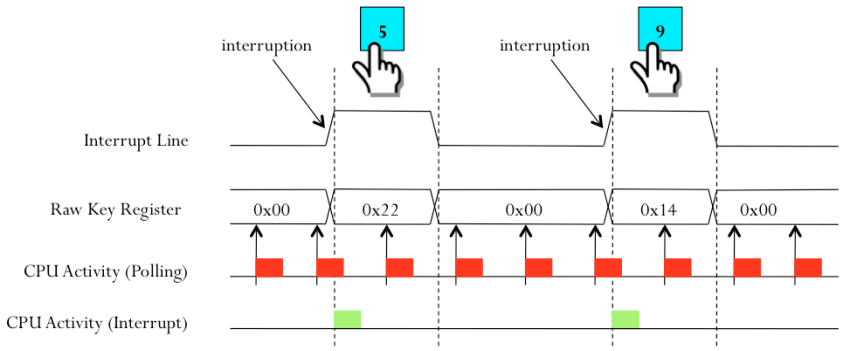
- Keypads
  - Polling vs interruptions
  - Problèmes principaux
  - Pilotes
- Touch Screens
  - Typologies
  - Contrôleurs
  - Pilotes

Keypad

Exemple: Phone Keyboard



Polling vs Interruptions



	Avantages	Inconvénients
Polling	Gestion simple	• Beaucoup de consommation d'énergie • Beaucoup de trafic sur le bus
Interrupts	• Optimisation de la consommation d'énergie • Optimisation du trafic sur le bus • Temps réel	• Gestion plus compliquée • Stack management

Exemple: Polling

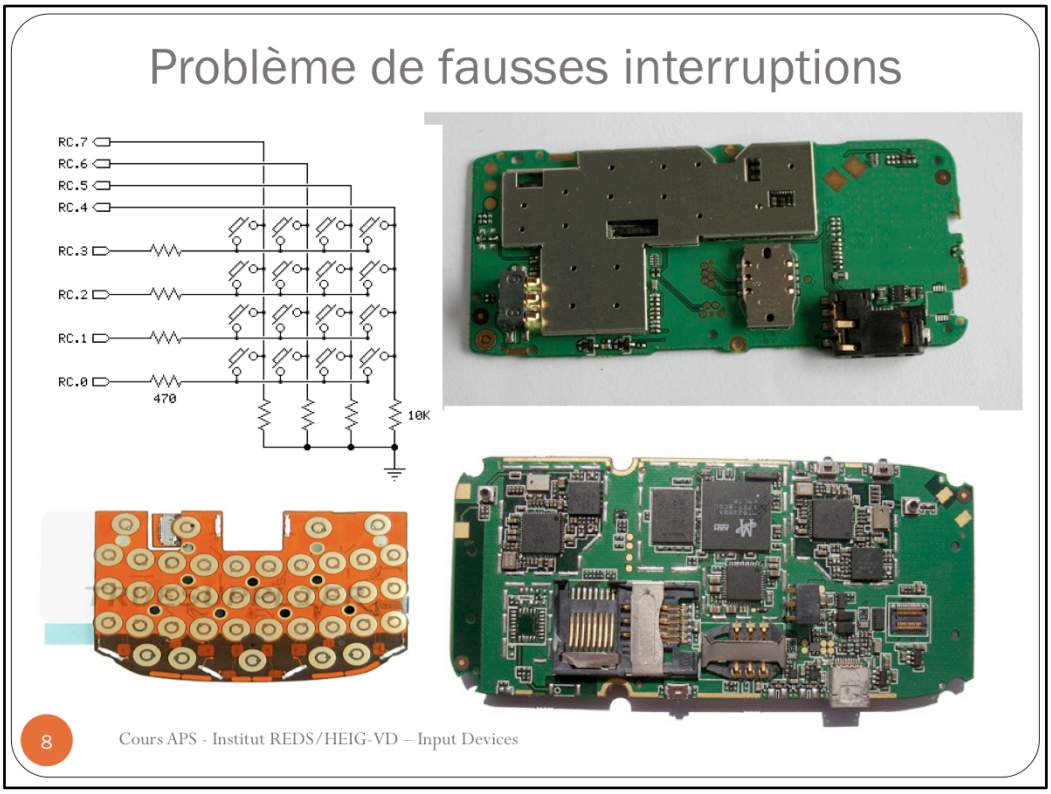
```
void timer key_poll()
{
  uint raw_key;
  uint *key_buff = KEY_BUFFER;
  uint *key_ctrl = KEY_CONTROL;

  timer_stop(key_timer);

  if (*key_buff != 0) {
     raw_key = *key_buff;

     switch(raw_key)  {
       case KEY_1: os_send_key('1');
         break;
       case KEY_2: os_send_key('2');
         break;
       ...
       case KEY_POUND: os_send_key('#');
         break;
       default:
         break;
     }
     *key_ctrl = 0;
  }

  timer_start(key_timer, POLL_TIME);
}
```

```
#define KEY_1        0x41
#define KEY_2        0x21
#define KEY_3        0x11
#define KEY_4        0x42
#define KEY_5        0x22
#define KEY_6        0x12
#define KEY_7        0x44
#define KEY_8        0x24
#define KEY_9        0x14
#define KEY_0        0x28
#define KEY_STAR     0x48
#define KEY_POUND    0x18

#define KEY_BUFFER   0x60
#define KEY_CONTROL  0x61
#define KEY_INT      0x20

#define POLL_TIME    100
```

Exemple: Interrupt

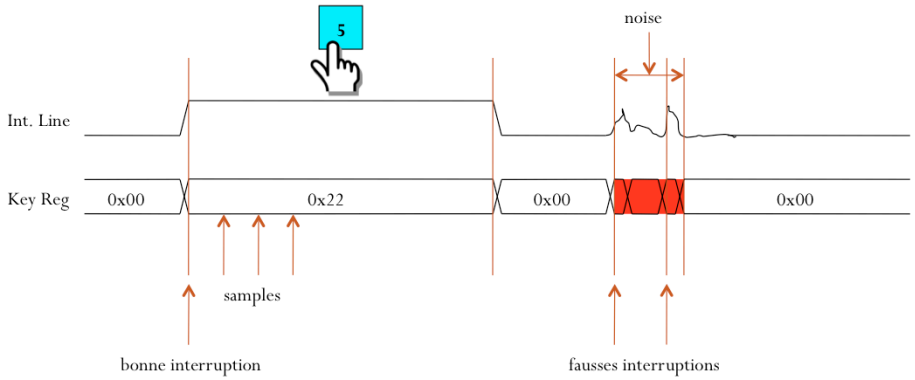
```
void _interrupt key_int()
{
  uint raw_key;

  _asm {
    dii                     ; disable interrupts
    push a1                 ; save context
    or   icr,KEY_INT        ; clear key interrupt
    in   a1,KEY_BUFFER      ; get the key that was pressed
    mov  raw_key,a1         ; store the key in variable raw_key
    and  a1, 0h             ; clean a1
    out  KEY_CONTROL,a1     ; clear the key buffer
    pop  a1                 ; restore context
    eni                     ; re-enable interrupts
  }

  switch(raw_key)  {
    case KEY_1: os_send_key('1');
      break;
    case KEY_2: os_send_key('2');
      break;
    ...
    case KEY_POUND: os_send_key('#');
      break;
    default:
      break;
  }
}
```

```
#define KEY_1        0x41
#define KEY_2        0x21
#define KEY_3        0x11
#define KEY_4        0x42
#define KEY_5        0x22
#define KEY_6        0x12
#define KEY_7        0x44
#define KEY_8        0x24
#define KEY_9        0x14
#define KEY_0        0x28
#define KEY_STAR     0x48
#define KEY_POUND    0x18

#define KEY_BUFFER   0x60
#define KEY_CONTROL  0x61
#define KEY_INT      0x20
```



EM Shilding est pas très simple pour les circuits liés au kipad

Problème de fausses interruptions

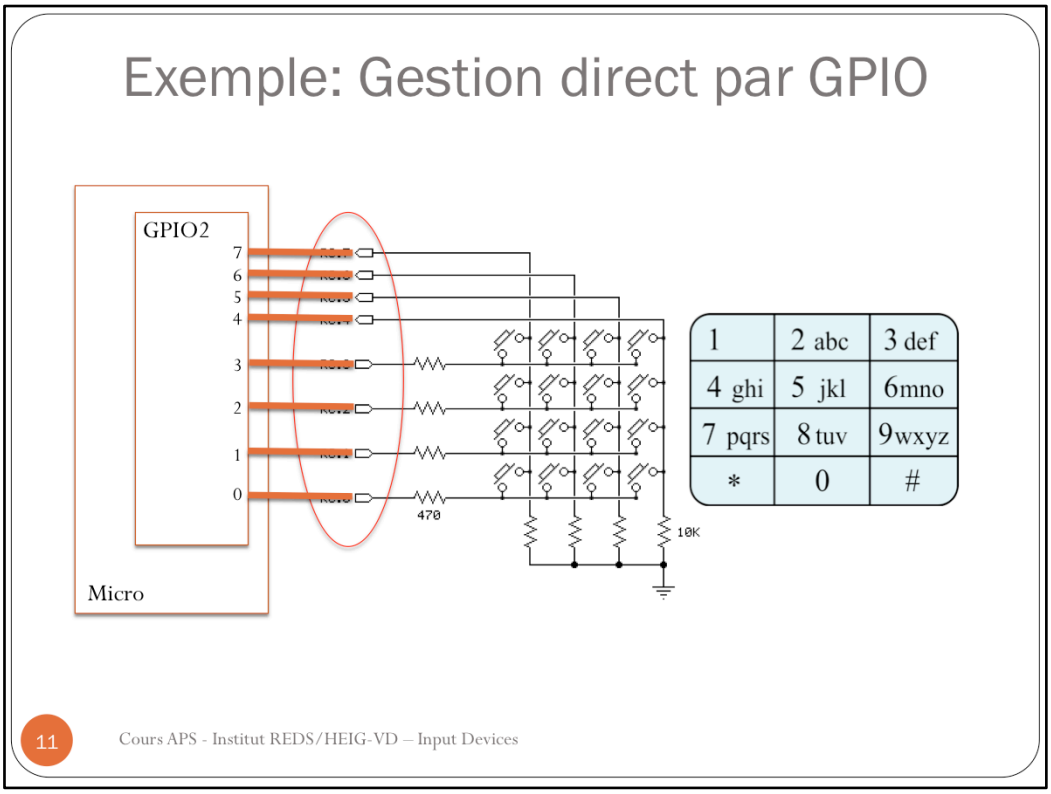


Exemple: Hybrid

```
void _interrupt key_int()
{
  timer_start(key_timer, POLL_TIME);
}
```

```
void timer key_timer()
{
  static uint old_raw_key = 0;
  static int i = 0;
  uint raw_key;
  uint *key_buff = KEY_BUFFER;
  uint *key_ctrl = KEY_CONTROL;

  timer_stop(key_timer);
  raw_key = *key_buff;

  if (i == 0) {
    old_raw_key = raw_key;
    i++;
    timer_start(key_timer, POLL_TIME);
    return;
  }

  if (raw_key == old_raw_key) {
    if (i++ < MAX_POLL) {
      timer_start(key_timer, POLL_TIME);
    } else {
      i = 0;
      switch(raw_key)  {
       case KEY_1: os_send_key('1');
         break;
       ...
       case KEY_POUND: os_send_key('#');
         break;
       default:
         break;
      }
      *key_ctrl = 0;
    }
  } else {
    *key_ctrl = 0;
  }
}
```

```
#define KEY_1         0x41
#define KEY_2         0x21
#define KEY_3         0x11
#define KEY_4         0x42
#define KEY_5         0x22
#define KEY_6         0x12
#define KEY_7         0x44
#define KEY_8         0x24
#define KEY_9         0x14
#define KEY_0         0x28
#define KEY_STAR      0x48
#define KEY_POUND     0x18

#define KEY_BUFFER    0x60
#define KEY_CONTROL   0x61
#define KEY_INT       0x20

#define POLL_TIME     100
#define MAX_POLL      5
```

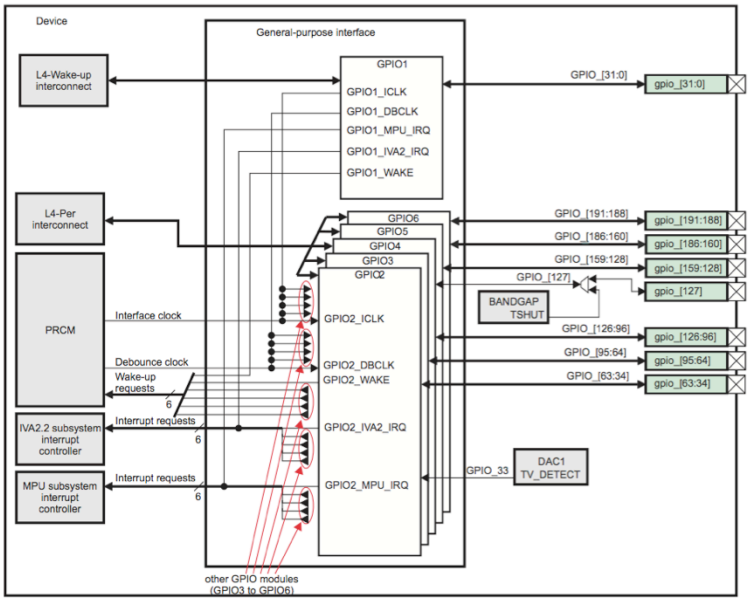


Pour gérer ce keypad on doit

1. Configurer GPIO2.0, GPIO2.1, GPIO2.2 et GPIO2.3 comme output
2. Configurer GPIO2.4, GPIO2.5, GPIO2.6 et GPIO2.7 comme input
3. Périodiquement « scanner » GPIO2.0, GPIO2.1, GPIO2.2 et GPIO2.3 en enviant un signal HIGH et lire GPIO2.4, GPIO2.5, GPIO2.6 et GPIO2.7 pour voir si il y a un signal HIGH

Le point 3 peut être géré avec interruptions ou avec un gestion hybride.

General Purpose Input Output (1/4)



General Purpose Input Output (2/4)

Module Name	Base Address
GPIO1	0x4831 0000
GPIO2	0x4905 0000
GPIO3	0x4905 2000
GPIO4	0x4905 4000
GPIO5	0x4905 6000
GPIO6	0x4905 8000

Register Name	Type	Register Width (Bits)	Address Offset
GPIO_REVISION	R	32	0x000
GPIO_SYSCONFIG	RW	32	0x010
GPIO_SYSSTATUS	R	32	0x014
GPIO_IRQSTATUS1	RW	32	0x018
GPIO_IRQENABLE1	RW	32	0x01C
GPIO_WAKEUPENABLE	RW	32	0x020
GPIO_IRQSTATUS2	RW	32	0x028
GPIO_IRQENABLE2	RW	32	0x02C
GPIO_CTRL	RW	32	0x030
GPIO_OE	RW	32	0x034
GPIO_DATAIN	R	32	0x038
GPIO_DATAOUT	RW	32	0x03C
GPIO_LEVELDETECT0	RW	32	0x040
GPIO_LEVELDETECT1	RW	32	0x044
GPIO_RISINGDETECT	RW	32	0x048
GPIO_FALLINGDETECT	RW	32	0x04C
GPIO_DEBOUNCENABLE	RW	32	0x050
GPIO_DEBOUNCINGTIME	RW	32	0x054
GPIO_CLEARIRQENABLE1	RW	32	0x060
GPIO_SETIRQENABLE1	RW	32	0x064
GPIO_CLEARIRQENABLE2	RW	32	0x070
GPIO_SETIRQENABLE2	RW	32	0x074
GPIO_CLEARWKUENA	RW	32	0x080
GPIO_SETWKUENA	RW	32	0x084
GPIO_CLEARDATAOUT	RW	32	0x090
GPIO_SETDATAOUT	RW	32	0x094

General Purpose Input Output (3/4)

Description This register provides IRQ 1 enable information.
Type RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQENABLE1																															

Bits	Field Name	Description	Type	Reset
31:0	IRQENABLE1	Interrupt 1 Enable Register 0x0: Disable IRQ generation for channel N 0x1: Enable IRQ generation for channel N	RW	0x00000000

Description This register provides IRQ 1 status information.
Type RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IRQSTATUS1																															

Bits	Field Name	Description	Type	Reset
31:0	IRQSTATUS1	Interrupt 1 Status Register. Write a 1 in the corresponding bit to clear it to 0. Write 0 in the corresponding bit does not affect its value. 0x0: IRQ channel N not triggered 0x1: IRQ channel N triggered	RW	0x00000000

General Purpose Input Output (4/4)

Description This register provides wakeup-enable information.
Type RW

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WAKEUPEN																															

Bits	Field Name	Description	Type	Reset
31:0	WAKEUPEN	Wake Up Enable Register 0x0: Disable wake-up generation for channel N 0x1: Enable wake-up generation for channel N	RW	0x00000000

Touch Screen

Touch Screens Overview (1/2)

Beaucoup de différentes technologies:

- à ondes de surface
- résistive analogique
- capacitive
- à infrarouge
- optique
- FTIR (*Frustrated Total Internal Reflection*)
- NFI (*Near Field Imaging*)

Sur les smartphones 2 technologies principaux:

Single-touch

- Résistif
- Capacitif

Multi-touch

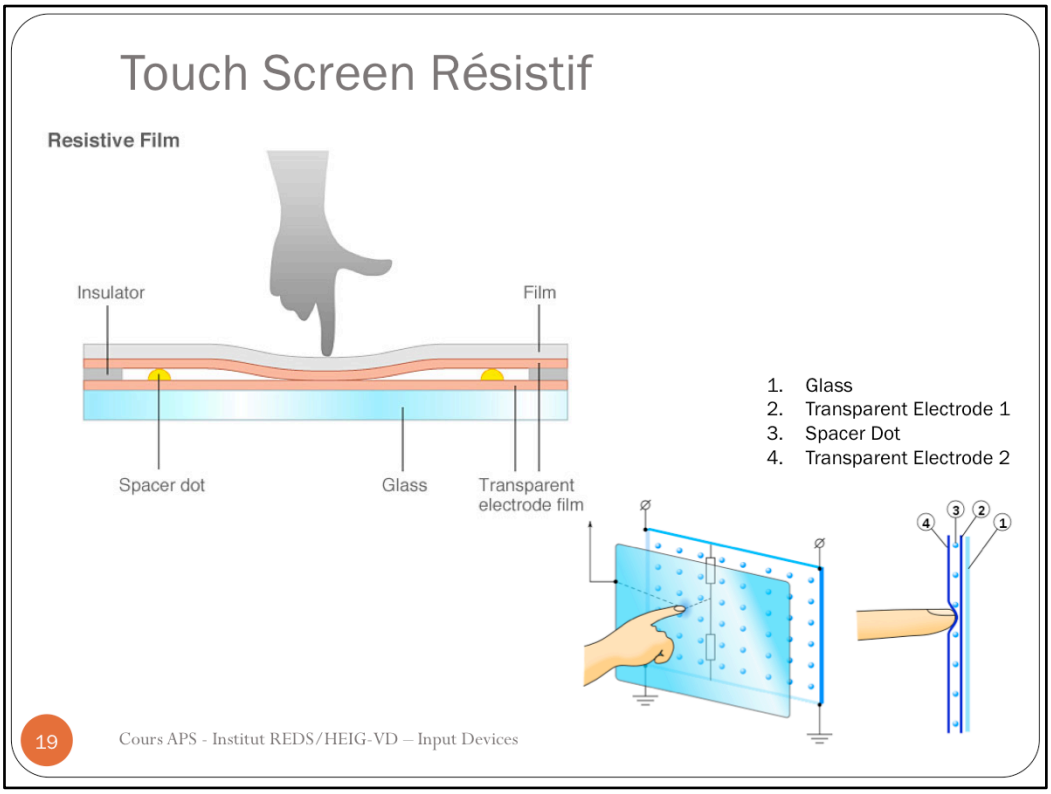
- Capacitif

Plus d'info sur les différentes technologies touch screens:

http://fr.wikipedia.org/wiki/%C3%89cran_tactile

Touch Screen Overview (2/2)

Sensing method	Resistive film	Capacitive
Light transmittance	Not so good	Good
Finger touch	Excellent	Excellent
Gloved touch	Excellent	No
Stylus touch	Excellent	Not so good (special-purpose stylus)
Durability	Not so good	Excellent
Resistance to water drops	Excellent	Excellent
Cost	Reasonable	Not so reasonable



Les systèmes résistifs sont constitués d'une plaque de verre dont la surface est conductrice (résistive). Celle-ci est recouverte par un film plastique dont la sous face est conductrice (résistive).

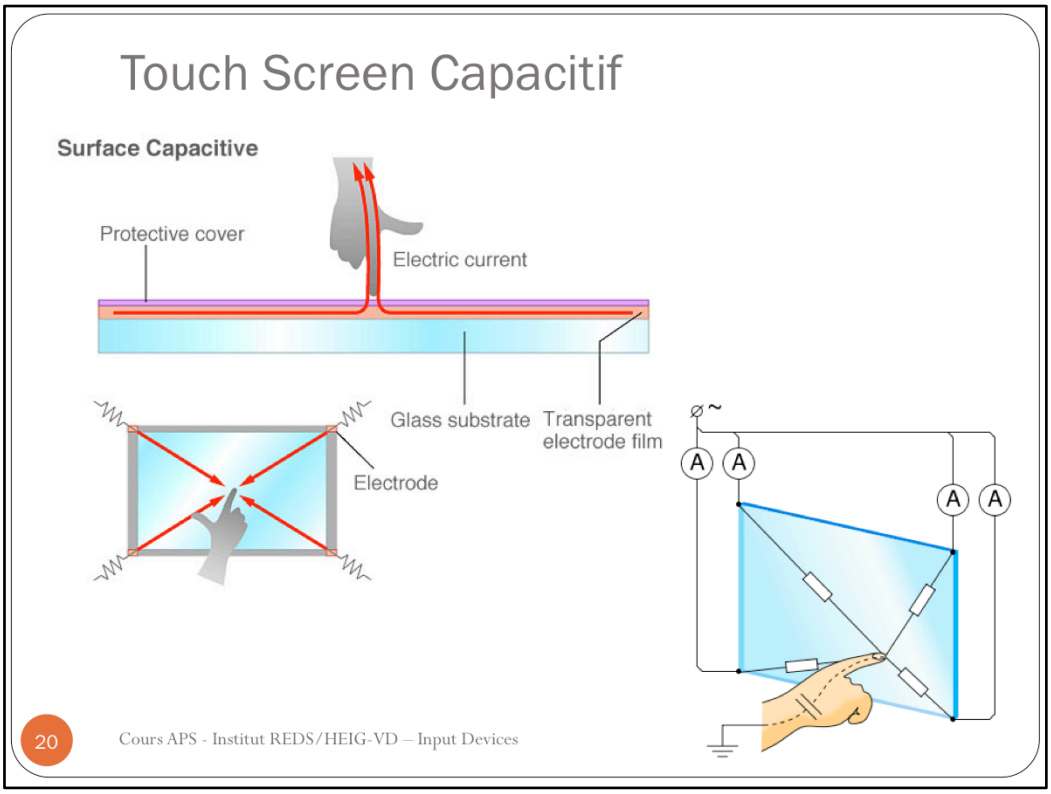
Ces deux couches sont tenues distantes par de microscopiques cales d'espacement ; de plus, une couche additionnelle est ajoutée en surface pour éviter les égratignures (par exemple, par les pointes de stylets).

Un courant électrique est induit dans les deux faces conductrices pendant l'opération. Lorsque l'utilisateur touche avec la pointe d'un stylet (ou d'un doigt), la pression exercée amorce un contact entre les deux faces électrifiées. La variation dans les champs électriques de ces deux faces conductrices permet de déterminer les coordonnées du point de contact. Une fois les coordonnées déterminées, le traitement logiciel par le système s'établit.

La conductivité électrique de ces deux faces s'use un peu lors de chaque contact entre elles (à cause des décharges électriques : micro étincelles). C'est pourquoi la précision de la détection des coordonnées du point touché se réduit avec l'usage. Cette technologie oblige l'utilisateur à *recalibrer* le pavé tactile. Ce recalibrage consiste à masquer l'usure du tactile en répartissant, sur toute sa surface, les erreurs des régions tactiles les plus usagées.

de périphériques utilisant ce système : ordinateur portable à dalle tactile sous Windows 7, les anciens PDA de l'entreprise PALM, certains smartphones (HTC Tattoo, HTC Tytn II, LG Viewty…)

Source: Wikipedia



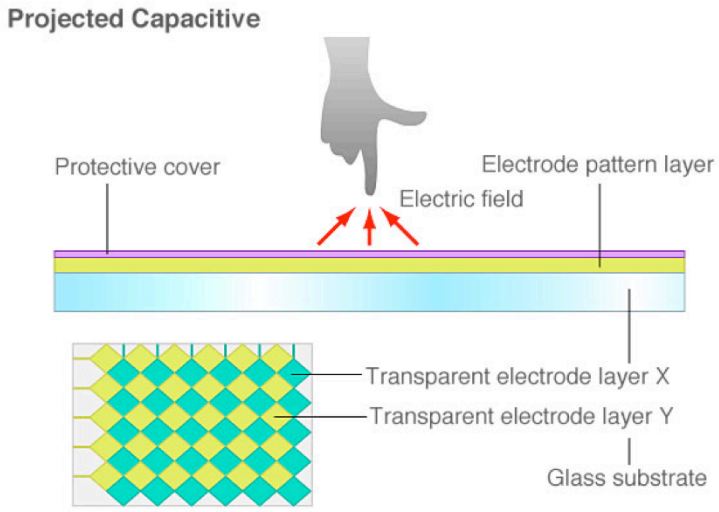
Dans les systèmes capacitifs, une couche qui accumule les charges, à base d'indium, métal de plus en plus rare, est placée sur la plaque de verre du moniteur. Lorsque l'utilisateur touche la plaque avec son doigt, certaines de ces charges lui sont transférées. Les charges qui quittent la plaque capacitive créent un déficit quantifiable. Avec un capteur dans chacun des coins de la plaque, il est possible en tout temps de mesurer et de déterminer les coordonnées du point de contact. Le traitement de cette information demeure le même que pour les circuits résistifs.

Un avantage majeur des systèmes capacitifs, par rapport aux résistifs, est leur capacité à laisser passer la lumière avec un meilleur rendement. En effet, jusqu'à 90 % de la lumière traversera une surface capacitive par rapport à un maximum de 75 % pour les systèmes résistifs, ce qui donne une clarté d'image supérieure pour les systèmes capacitifs.

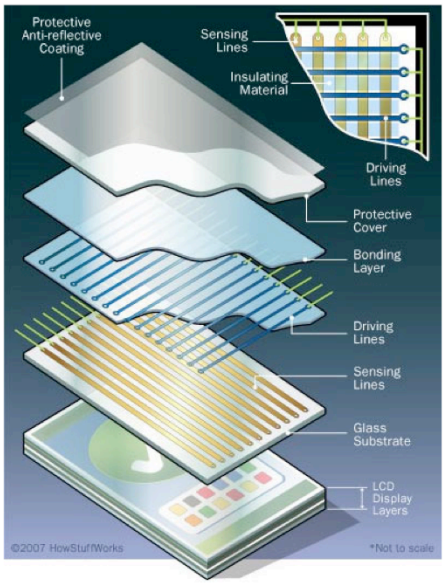
Malheureusement ces systèmes ne sont pas facilement extensibles aux écrans plus grands qu'une vingtaine de pouces. Ils sont par contre très compétitifs aux petites tailles et on les retrouve ainsi dans de nombreux smartphone et tablettes, par exemple la gamme des Nokia Lumia, la gamme des Galaxy S2, Galaxy Tab, Apple iPhone, iPad et iPod Touch, Motorola Milestone, HTC Desire, LG Arena, Nexus One, Nexus 7…

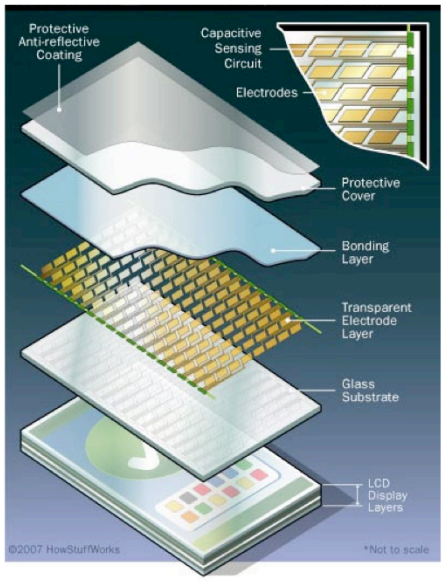
Source: Wikipedia

Touch Screen Multi-Touch Capacitif (1/2)

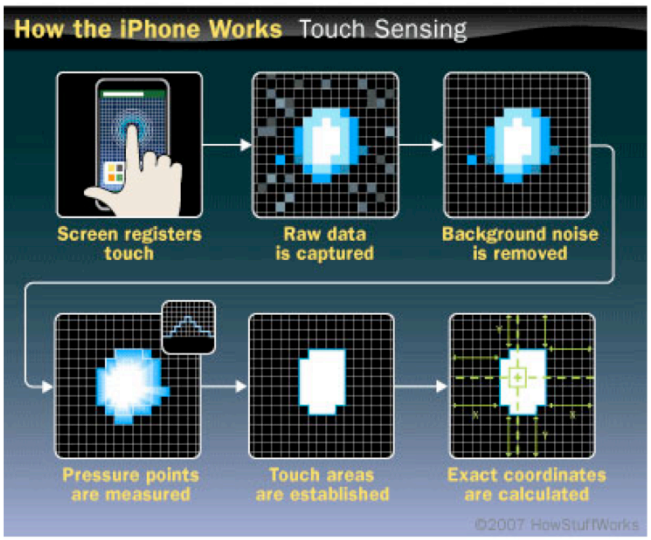


Touch Screen Multi-Touch Capacitif (2/2)





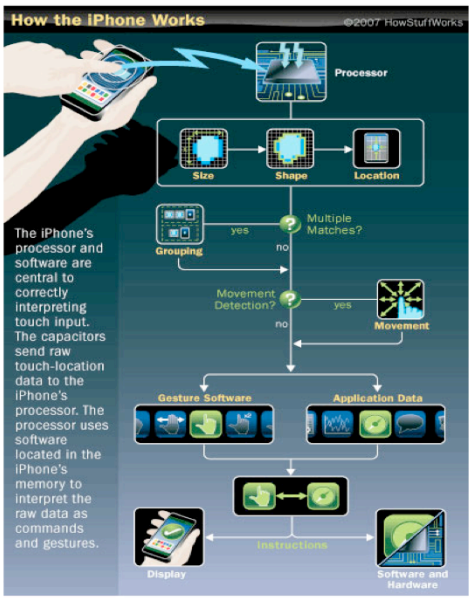
Touch Screen Controller – Overview (1/2)



Référence: Tracy V. Wilson and Wesley Fenlon, « How the iphone works », electronics.howstuffworks.com

Lien avec plus d'infos: http://electronics.howstuffworks.com/iphone3.htm

Touch Screen Controller – Overview (2/2)



Les contrôleurs touch screen normalement s'occupent de la détection des évènements « touch »

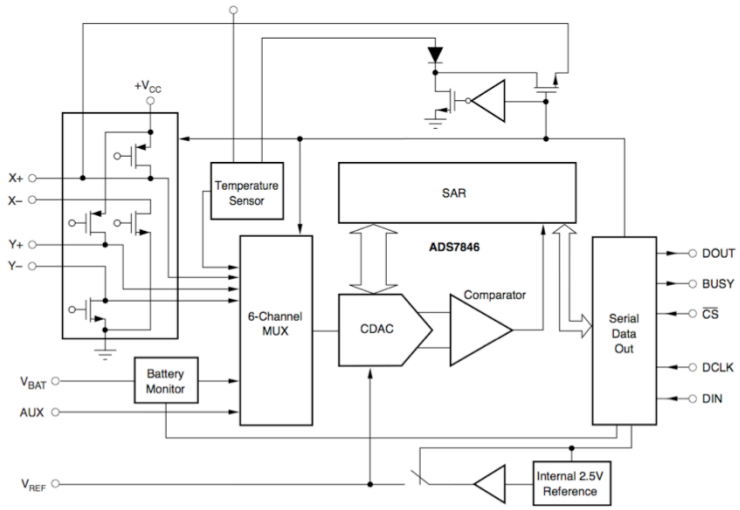
Ils font du pre-processing pour bien identifier les coordonnées des points touchés et ils les envoient au main processeur.

Le logiciel dans le main processeur, en analysant la séquence et le nombre des coordonnées reçues, gère les «gestures» (slide left / right, zoom in / out, etc)

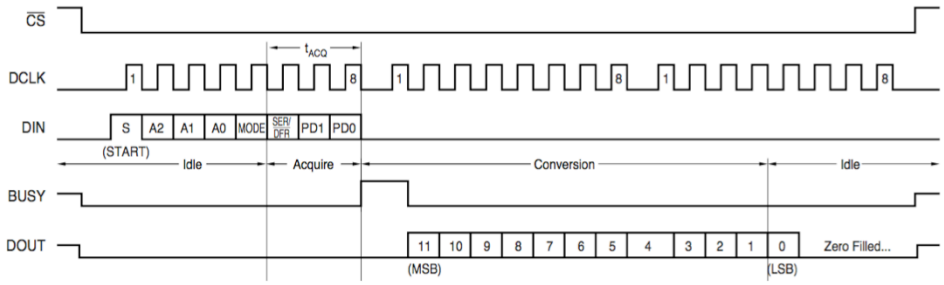
Référence: Tracy V. Wilson and Wesley Fenlon, « How the iphone works », electronics.howstuffworks.com

Lien avec plus d'infos: http://electronics.howstuffworks.com/iphone3.htm

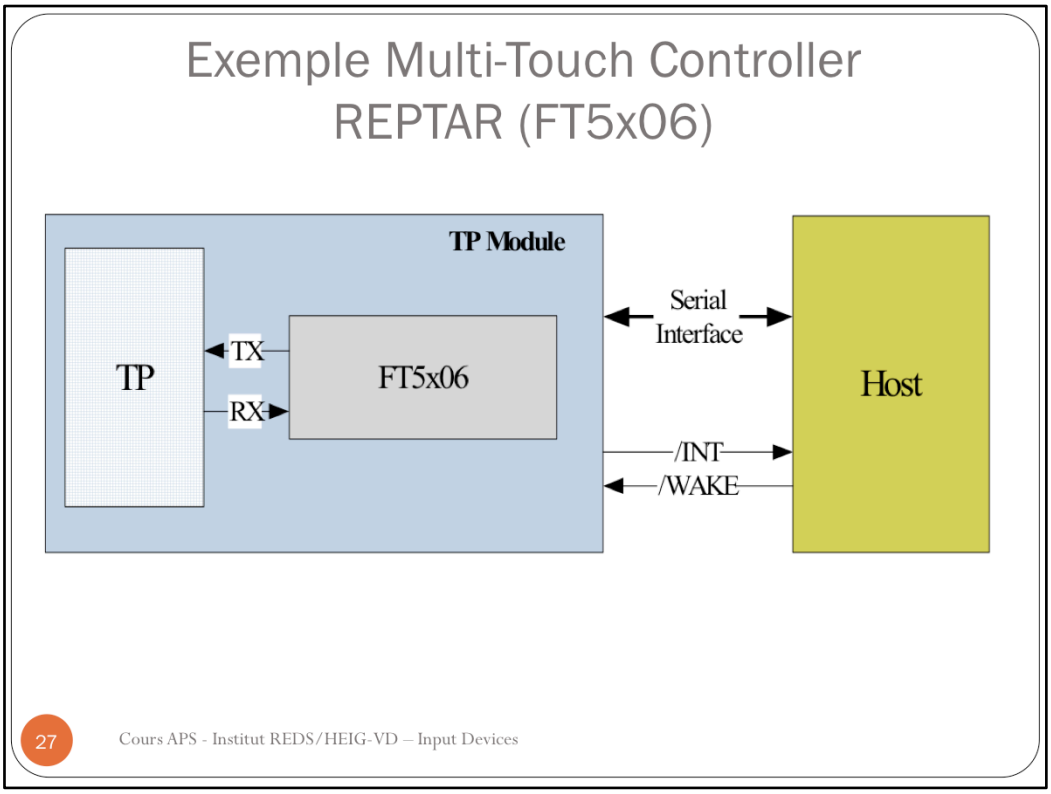
Touch Screen Controller – Single Touch (1/2)



Touch Screen Controller – Single Touch (2/2)

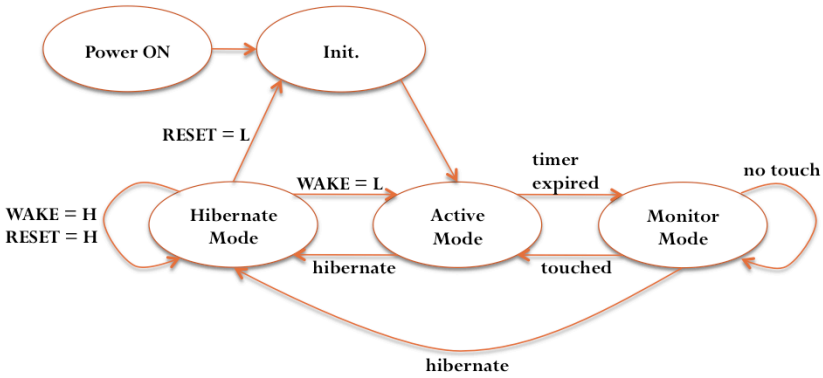


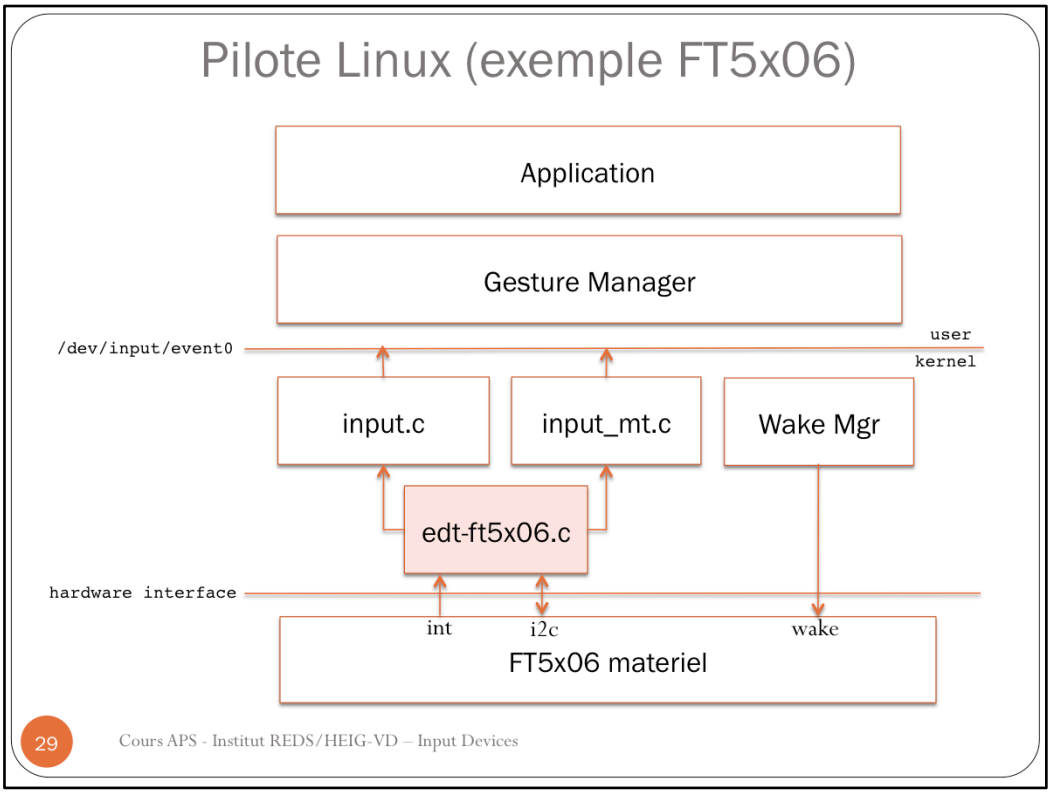
A2	A1	A0	V_{BAT}	AUX_{IN}	TEMP	Y–	X+	Y+	Y-POSITION	X-POSITION	Z_1-POSITION	Z_2-POSITION	X-DRIVERS	Y-DRIVERS
0	0	0			+IN (TEMP0)								Off	Off
0	0	1					+IN		Measure				Off	On
0	1	0	+IN										Off	Off
0	1	1					+IN				Measure		X–, On	Y+, On
1	0	0				+IN						Measure	X–, On	Y+, On
1	0	1						+IN		Measure			On	Off
1	1	0		+IN									Off	Off
1	1	1			+IN (TEMP1)								Off	Off



Plus d'info: http://www.displayfuture.com/Display/datasheet/controller/FT5x06.pdf

Touch Screen Controller – Power States Example





Plus d'info sur le repertoire REPTAR, cherchez le file edt-ft5x06.c (find . –name "edt-ft5x06.c ")

Exemple: Linux Single Touch Protocol

Here is what a minimal event sequence for a one-contact touch would look:

```
   ABS_EVT ABS_X x
   ABS_EVT ABS_Y y
   ABS_SYN
   ABS_EVT ABS_X x
   ABS_EVT ABS_Y y
   ABS_SYN
   ABS_EVT ABS_X x
   ABS_EVT ABS_Y y
   ABS_SYN
```

But also

```
   ABS_EVT ABS_Y Y
   ABS_EVT ABS_X x
   ABS_SYN
   ABS_EVT ABS_Y Y
   ABS_EVT ABS_X x
   ABS_SYN
   ABS_EVT ABS_Y Y
   ABS_EVT ABS_X x
   ABS_SYN
```

Exemple: Linux Multi Touch Protocol (1/2)

```
Here is what a minimal event sequence for a two-contact touch would look
like for a type A device:

   ABS_MT_POSITION_X x[0]
   ABS_MT_POSITION_Y y[0]
   SYN_MT_REPORT
   ABS_MT_POSITION_X x[1]
   ABS_MT_POSITION_Y y[1]
   SYN_MT_REPORT
   SYN_REPORT

The sequence after moving one of the contacts looks exactly the same; the
raw data for all present contacts are sent between every synchronization
with SYN_REPORT.

Here is the sequence after lifting the first contact:

   ABS_MT_POSITION_X x[1]
   ABS_MT_POSITION_Y y[1]
   SYN_MT_REPORT
   SYN_REPORT

And here is the sequence after lifting the second contact:

   SYN_MT_REPORT
   SYN_REPORT
```

Exemple: Linux Multi Touch Protocol (2/2)

```
Here is what a minimal event sequence for a two-contact touch would look
like for a type B device:

   ABS_MT_SLOT 0
   ABS_MT_TRACKING_ID 45
   ABS_MT_POSITION_X x[0]
   ABS_MT_POSITION_Y y[0]
   ABS_MT_SLOT 1
   ABS_MT_TRACKING_ID 46
   ABS_MT_POSITION_X x[1]
   ABS_MT_POSITION_Y y[1]
   SYN_REPORT

Here is the sequence after moving contact 45 in the x direction:

   ABS_MT_SLOT 0
   ABS_MT_POSITION_X x[0]
   SYN_REPORT

Here is the sequence after lifting the contact in slot 0:

   ABS_MT_TRACKING_ID -1
   SYN_REPORT
```

Pilote Linux (code exemple) (1/5)

```
struct rx_touch_data {
          __be16 header;
          u8 len;
          u8 ntouches;
          u8 reserved;
          struct ft5x06_xy_coordinate {
#if defined(__LITTLE_ENDIAN_BITFIELD)
                    u8 x_high:4;
                    u8 event:4;
#else
                    u8 event:4;
                    u8 x_high:4;
#endif
                    u8 x_low;

#if defined(__LITTLE_ENDIAN_BITFIELD)
                    u8 y_high:4;
                    u8 tid:4;
#else
                    u8 tid:4;
                    u8 y_high:4;
#endif
                    u8 y_low;
          } __packed p[MAX_TOUCHES];
          u8 crc;
} __packed;
```

```
struct ft5x06 {
          struct i2c_client *client;
          struct input_dev *input;
          int gpio_reset;
          int gpio_irq;
          char model[MODEL_FW_LEN];
          char fw_version[MODEL_FW_LEN];
          int num_x;
          int num_y;
          /* mutex used to prevent access problems
           * when switching between modes */
          struct mutex mutex;
          bool factory_mode;
};
```


Plus d'info sur le repertoire REPTAR, cherchez le file edt-ft5x06.c (find . –name "edt-ft5x06.c ")

Pilote Linux (code exemple) (2/5)

```
static void __devinit ft5x06_init_input_dev(struct ft5x06 *priv)
{
          struct i2c_client *client = priv->client;
          struct device *dev = &client->dev;
          struct input_dev *input = priv->input;

          priv->num_x = ft5x06_register_read(priv, W_REGISTER_NUM_X);
          priv->num_y = ft5x06_register_read(priv, W_REGISTER_NUM_Y);

          __set_bit(EV_SYN, input->evbit);
          __set_bit(EV_KEY, input->evbit);
          __set_bit(EV_ABS, input->evbit);
          __set_bit(BTN_TOUCH, input->keybit);

          /* Single touch */
          input_set_abs_params(input, ABS_X, 0, priv->num_x * SENSOR_RESOLUTION - 1, 0, 0);
          input_set_abs_params(input, ABS_Y, 0, priv->num_y * SENSOR_RESOLUTION - 1, 0, 0);


          /* Multi touch */
          input_mt_init_slots(input, MAX_TOUCHES);
          input_set_abs_params(input, ABS_MT_POSITION_X, 0, priv->num_x * SENSOR_RESOLUTION - 1, 0, 0);
          input_set_abs_params(input, ABS_MT_POSITION_Y, 0, priv->num_y * SENSOR_RESOLUTION - 1, 0, 0);
          input->name = priv->model;
          input->id.bustype = BUS_I2C;
          input->dev.parent = dev;

          input_set_drvdata(input, priv);
}
```


Plus d'info sur le repertoire REPTAR, cherchez le file edt-ft5x06.c (find . –name "edt-ft5x06.c ")

Pilote Linux (code exemple) (3/5)

```
static irqreturn_t ft5x06_isr(int irq, void *irq_data)
{
          struct ft5x06 *priv = irq_data;
          struct i2c_client *client = priv->client;
          struct device *dev = &client->dev;
          struct rx_touch_data frm;
          int rc;

          memset(&frm, 0, sizeof(frm));

          rc = ft5x06_i2c_cmd(client, CMD_GET_TOUCH_DATA, NULL, 0,
                              &frm, sizeof(frm));
          if (rc < 0) {
                    dev_err(dev, "Unable to get touch data, error: %d\n", rc);
                    goto out;
          }

          ft5x06_report_events(priv, &frm);

out:
          return IRQ_HANDLED;
}
```

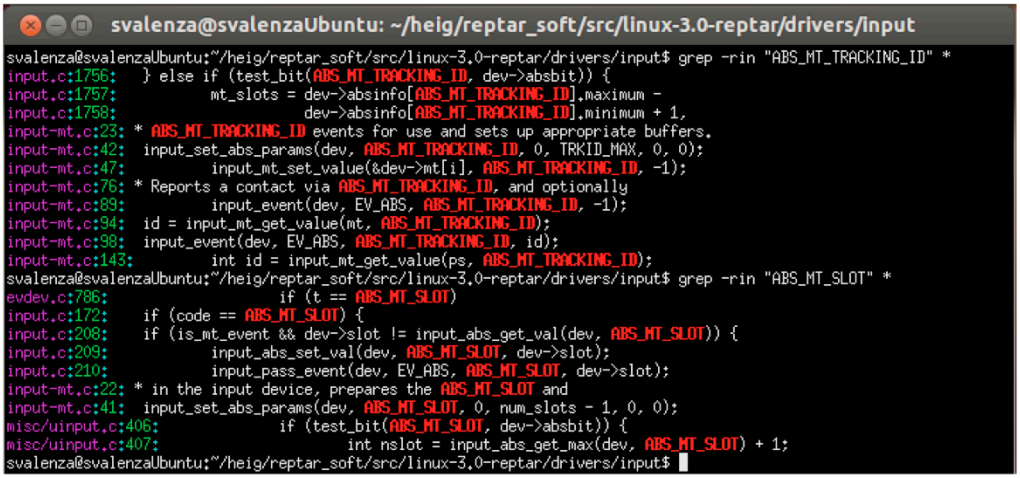

Plus d'info sur le repertoire REPTAR, cherchez le file edt-ft5x06.c (find . –name "edt-ft5x06.c ")

Pilote Linux (code exemple) (4/5)

```
static void ft5x06_report_events(const struct ft5x06 *priv, const struct rx_touch_data *frm)
{
          …
          FOREACH_TOUCH(p, frm) {
                    if (p->event == EVENT_TOUCH_RESERVED)
                              continue;

                    if (p->tid > MAX_TOUCHES - 1)
                              continue;

                    touch[p->tid] = 1;

                    input_mt_slot(input, p->tid);
                    input_mt_report_slot_state(input, MT_TOOL_FINGER,
                                                        p->event != EVENT_TOUCH_UP);

                    if (p->event != EVENT_TOUCH_UP) {
                              x = (p->x_high << 8) | p->x_low;
                              y = (p->y_high << 8) | p->y_low;

                              input_report_abs(input, ABS_MT_POSITION_X, x);
                              input_report_abs(input, ABS_MT_POSITION_Y, y);
                    }
          }
          …
          input_mt_report_pointer_emulation(input, false);
          input_sync(input);
}
```


Plus d'info sur le repertoire REPTAR, cherchez le file edt-ft5x06.c (find . –name "edt-ft5x06.c ")

Pilote Linux (code exemple) (5/5)



Plus d'info sur le repertoire REPTAR, cherchez le file edt-ft5x06.c (find . –name "edt-ft5x06.c ")

Références

- Texas Instruments, "**AM/DM37x Multimedia Device**", Technical Reference Manual