WINSTAR_CTP Application Note

Capacitor Touch Panel

Version 0.2
Date: 2013/07/24

WINSTAR_CTP Application Note

WINSTAR_CTP Application Note

1. Record of Revision

Revision Date	Page	Contents	Editor
2013/04/19	-	New Release	Lisa
2013/05/24	P6&P10	Add Operation mode and Power on sequence	Lisa
2013/07/24	P7&P9	Modified the access status and description for registers	Lisa

2. Features

- Interface is I2C only.
- Operating Voltage is 2.8V to 3.6V.
- Operating Temperature Range is -40 degree to +85 degree.
- Supports up to 8.9" touch panel, now 3.5", 4.3", 5.7", and 7" are available.
- The resolution is 896x640 dots for 3.5".
- The resolution is 1280x768 dots for 4.3".
- The resolution is 1408x1024 dots for 5.7"
- The resolution is 1792x1024 dots for 7".
- Built-in 8051-based MCU with 28KB Program Memory, 6KB Data Memory and 256B Internal Data Space.

2.1 Electrical Characteristics

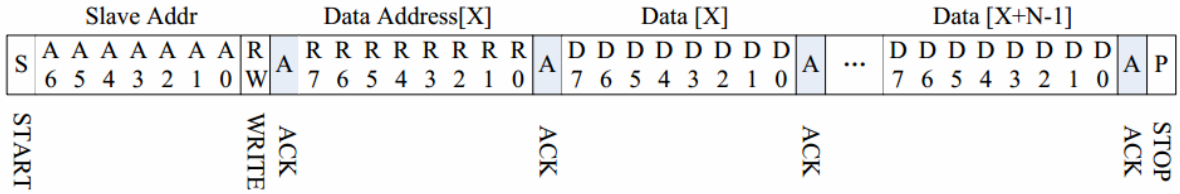
Item	Symbol	Condition	Min.	Typ.	Max.	Unit
Operating Temperature Range	Top	Absolute Max	-40	-	+85	° C
Storage Temperature Range	Tst	Absolute Max	-50	-	+95	° C
Supply Voltage	VDD		2.8	-	3.3	V
Supply Current – Operating	Iopr	Ta=25°C, VDD=2.8V	-	6.0	-	mA
Supply Current – Hibernate	Islp	Ta=25°C, VDD=2.8V	-	0.03	-	mA
"H" Level input	VIH		0.7*VDD	-	VDD	V
"L" Level input	VIL		VSS	-	0.3*VDD	V
"H" Level output	VOH		0.7*VDD	-	VDD	V
"L" Level output	VOL		VSS	-	0.3*VDD	V

3. Pin Define

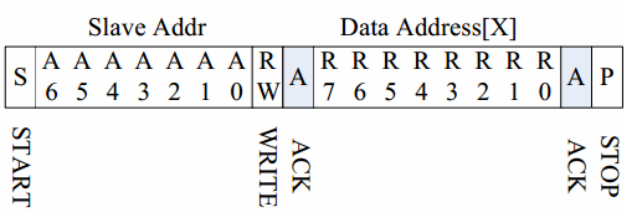
Pin No.	Symbol	Function Description
1	VSS	Ground
2	VDD	Power Supply
3	SCL	Serial I2C Clock（Require pull-up resistor）
4	NC	
5	SDA	Serial I2C Data（Require pull-up resistor）
6	NC	
7	/RST	Reset signal
8	/WAKE	External interrupt signal from host
9	/INT	Interrupt signal from touch panel module to host
10	VSS	Ground

3.1 I2C Read/Write description

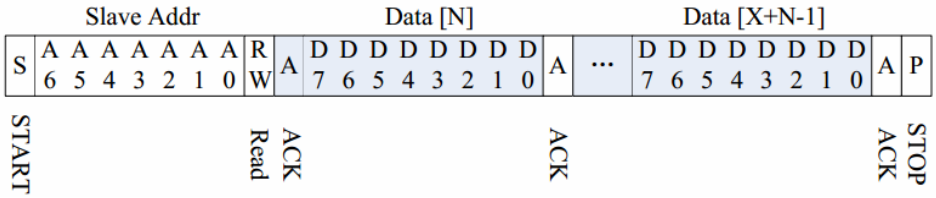
Write N bytes to I2C slave where Slave address is 0x38.



Set Data Address

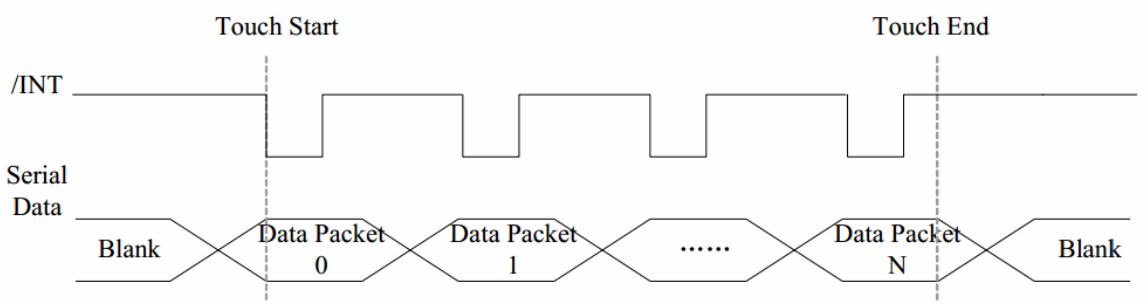


Read N bytes from I2C slave



3.2 /INT

An interrupt signal to inform the host that touch data is ready for read. /INT signal will be low if there is a touch detected.



4. Operation Mode

4.1 Active mode

When in this mode, the chip actively scans the panel. The default scan rate is 60 frames per second. The host processor can configure FT5x06 to speed up or to slow down.

4.2 Monitor Mode

When in this mode, the chip scans the panel at a reduced speed. The default scan rate is 25 frames per second and the host processor can increase or decrease this rate. When in this mode, most algorithms are stopped. A simpler algorithm is being executed to determine if there is a touch or not. When a touch is detected, the chip shall enter the Active mode immediately to acquire the touch information quickly. During this mode, the serial port is closed and no data shall be transferred with the host processor.

4.3 Hibernate Mode

In this mode, the chip is set in a power down mode. It shall only respond to the “WAKE” or “RESET” signal from the host processor. The chip therefore consumes very little current, which help prolong the standby time for the portable devices.

5. Capacitive Touch Registers

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Host Access
00h	DEVIDE_MODE		Device Mode [2..0]							R
01h	GEST_ID	Gesture ID [7..0]								R
02h	TD_STATUS						Touch Points [3..0]			R
03h	TOUCH1_XH	Event Flag				1st Touch X Position MSB [11..8]				R
04h	TOUCH1_XL	1st Touch X Position LSB [7..0]								R
05h	TOUCH1_YH	Touch ID [3..0]				1st Touch Y Position MSB [11..8]				R
06h	TOUCH1_YL	1st Touch Y Position LSB [7..0]								R
09h	TOUCH2_XH	Event Flag				2nd Touch X Position MSB [11..8]				R
0Ah	TOUCH2_XL	2nd Touch X Position LSB [7..0]								R
0Bh	TOUCH2_YH	Touch ID [3..0]				2nd Touch Y Position MSB [11..8]				R
0Ch	TOUCH2_YL	2nd Touch Y Position LSB [7..0]								R
0Fh	TOUCH3_XH	Event Flag				3rd Touch X Position MSB [11..8]				R
10h	TOUCH3_XL	3rd Touch X Position LSB [7..0]								R
11h	TOUCH3_YH	Touch ID [3..0]				Touch ID [3..0]				R
12h	TOUCH3_YL	3rd Touch Y Position LSB [7..0]								R
15h	TOUCH4_XH	Event Flag				4th Touch X Position MSB [11..8]				R
16h	TOUCH4_XL	4th Touch X Position LSB [7..0]								R
17h	TOUCH4_YH	Touch ID [3..0]				4th Touch Y Position MSB [11..8]				R
18h	TOUCH4_YL	4th Touch Y Position LSB [7..0]								R
1Bh	TOUCH5_XH	Event Flag				5th Touch X Position MSB [11..8]				R
1Ch	TOUCH5_XL	5th Touch X Position LSB [7..0]								R
1Dh	TOUCH5_YH	Touch ID [3..0]				5th Touch Y Position MSB [11..8]				R
1Eh	TOUCH5_YL	5th Touch Y Position LSB [7..0]								R
80h	ID_G_THGROUP	valid touching detect threshold								R/W
81h	ID_G_THPEAK	valid touching peak detect threshold								R/W
82h	ID_G_THCAL	the threshold when calculating the focus of touching								R/W
83h	ID_G_THWATER	the threshold when there is surface water								R
84h	ID_G_TEMP	the threshold of temperature compensation								R
85h	ID_G_THDIFF	the threshold whether the coordinate is different from original								R/W
86h	ID_G_CTRL						Power Control Mode [1..0]			R/W

Address	Name	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	Host Access
87h	ID_G_TIME_ENTER_MONITOR	the timer for entering monitor status								R/W
88h	ID_G_PERIODACTIVE					Period Active [3..0]				R/W
89h	ID_G_PERIODMONITOR	the timer of entering idle when in monitor status								R/W
A4h	ID_G_MODE	the interrupt status to host								R
A5h	ID_G_PMODE	Power Consume Mode								R/W

5.1 Capacitive Touch Register Description

Register No	Register Name	Bits	Value	Description
00h	Device Mode	[2:0]	000b 100b	Normal Operating Mode (Factory used) Test Mode -read raw data (reserved)
01h	Gesture ID	[7:0]	48h 49h 00h	Zoom In Zoom Out No Gesture
02h	Touch Points	[3:0]	000b 001b 010b 011b 100b 101b	0 touch points detected 1 touch point detected 2 touch points detected 3 touch points detected 4 touch points detected 5 touch points detected
03h	Touch 1 Event Flag	[7:6]	00b 01b 10b 11b	Put Down Put Up Contact Reserved
03h 04h 05h 06h	TOUCH1_XH TOUCH1_XL TOUCH1_YH TOUCH1_YL	[3:0] [7:0] [3:0] [7:0]	0h - 1h 00h - FFh 0h - 1h 00h - FFh	Upper 4 bits of X touch coordinate Lower 8 bits of X touch coordinate Upper 4 bits of Y touch coordinate Lower 8 bits of Y touch coordinate
09h	Touch 2 Event Flag	[7:6]	00b 01b 10b 11b	Put Down Put Up Contact Reserved
09h 0Ah 0Bh 0Ch	TOUCH2_XH TOUCH2_XL TOUCH2_YH TOUCH2_YL	[3:0] [7:0] [3:0] [7:0]	0h - 1h 00h - FFh 0h - 1h 00h - FFh	Upper 4 bits of X touch coordinate Lower 8 bits of X touch coordinate Upper 4 bits of Y touch coordinate Lower 8 bits of Y touch coordinate
0Fh	Touch 3 Event Flag	[7:6]	00b 01b 10b 11b	Put Down Put Up Contact Reserved
0Fh 10h 11h 12h	TOUCH3_XH TOUCH3_XL TOUCH3_YH TOUCH3_YL	[3:0] [7:0] [3:0] [7:0]	0h - 1h 00h - FFh 0h - 1h 00h - FFh	Upper 4 bits of X touch coordinate Lower 8 bits of X touch coordinate Upper 4 bits of Y touch coordinate Lower 8 bits of Y touch coordinate
15h	Touch 4 Event Flag	[7:6]	00b 01b 10b 11b	Put Down Put Up Contact Reserved
15h 16h 17h 18h	TOUCH4_XH TOUCH4_XL TOUCH4_YH TOUCH4_YL	[3:0] [7:0] [3:0] [7:0]	0h - 1h 00h - FFh 0h - 1h 00h - FFh	Upper 4 bits of X touch coordinate Lower 8 bits of X touch coordinate Upper 4 bits of Y touch coordinate Lower 8 bits of Y touch coordinate
1Bh	Touch 5 Event Flag	[7:6]	00b 01b 10b 11b	Put Down Put Up Contact Reserved
1Bh 1Ah 1Ch 1Eh	TOUCH5_XH TOUCH5_XL TOUCH5_YH TOUCH5_YL	[3:0] [7:0] [3:0] [7:0]	0h - 1h 00h - FFh 0h - 1h 00h - FFh	Upper 4 bits of X touch coordinate Lower 8 bits of X touch coordinate Upper 4 bits of Y touch coordinate Lower 8 bits of Y touch coordinate
80h	ID_G_THGROUP	[7:0]	00h - FFh	Valid touching detect threshold Default: 16h Actual value will be 4 times register's value
81h	ID_G_THPEAK	[7:0]	00h - FFh	valid touching peak detect threshold Default: 3Ch
82h	ID_G_THCAL	[7:0]	00h - FFh	Touch focus threshold (reserved) Default: E9h
83h	ID_G_THWATER	[7:0]	00h - FFh	threshold when there is surface water Default: 01h
84h	ID_G_THTEMP	[7:0]	00h - FFh	threshold of temperature compensation Default: 01h
85h	ID_G_THDIFF	[7:0]	00h - FFh	Touch difference threshold Default: A0h Actual value is 32 times the register's value
86h	ID_G_CTRL	[1:0]	00h 01h	Power Control Mode: Not Auto Jump Power Control Mode: Auto Jump
87h	ID_G_TIME_ENTER_MONITOR	[7:0]	00h - FFh	Delay to enter 'Monitor' status (s) Default: 0Ah
88h	ID_G_PERIODACTIVE	[3:0]	3h -Eh	Period of 'Active' status (ms) Default: 06h
89h	ID_G_PERIODMONITOR	[7:0]	1Eh -FFh	Timer to enter 'idle' when in 'Monitor' (ms) Default: 28h
A4h	ID_G_MODE	[0:0]	00h 01h	Interrupt status: Enable interrupt to host Interrupt status: Disable interrupt to host
A5h	ID_G_PMODE	[1:0]	00h 01h 03h	'Active' Mode Default: 01h 'Monitor' Mode 'Hibernate' Mode

6. I2C Timing Sequence

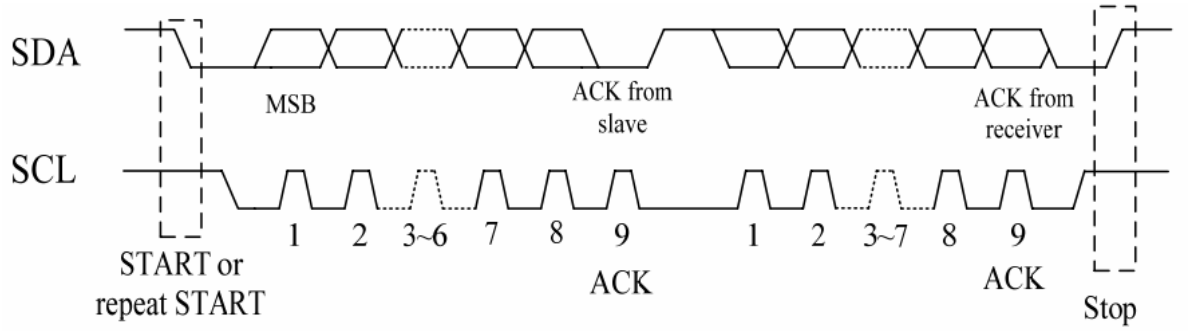
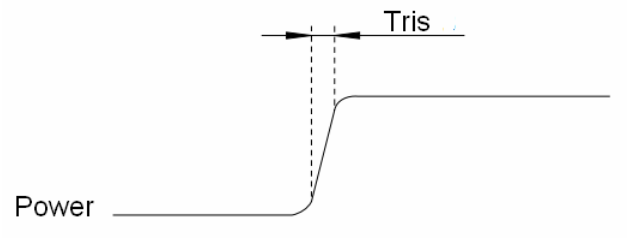


Table 6-1 I2C timing characteristics

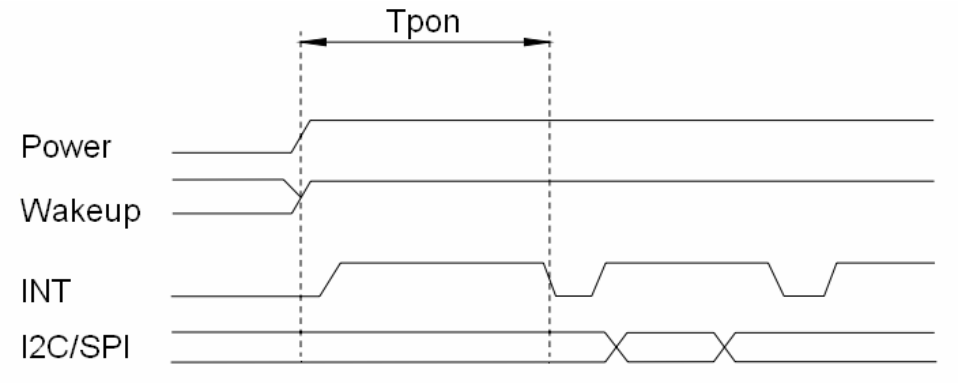
Parameter	Min	Max	Unit
SCL frequency	0	400	KHz
Bus free time between a STOP and START condition	4.7	-	us
Hold time (repeated) START condition	4.0	-	us
Data setup time	250	-	ns
Setup time for a repeated START condition	4.7	-	us
Setup Time for STOP condition	4.0	-	us

7. Power on, Reset, and Wakeup sequence

Wake, INT and I2C should be pulled down to be low before powering on. The signal of waking up should be set to be high after powering on. INT signal will be sent to the host after initializing all parameters and then start to report points to the host.



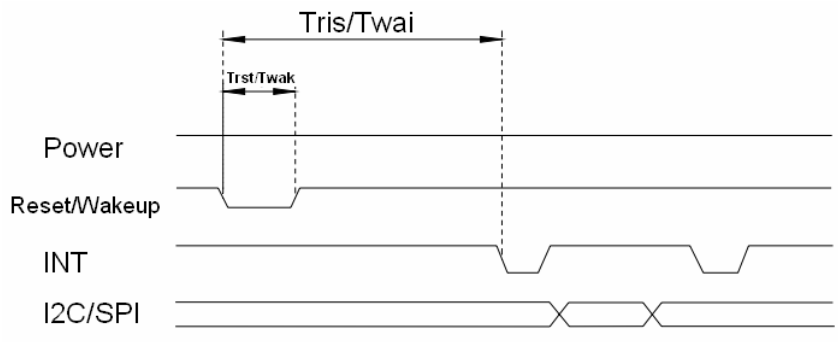
Power on time



Power on sequence

Reset time must be enough to guarantee reliable reset, the time of starting to report point after resetting approach to the time of starting to report point after powering on.

Wake time must be enough to wake up the system, the time of starting to report point after waking approach to the time of starting to report point after powering on

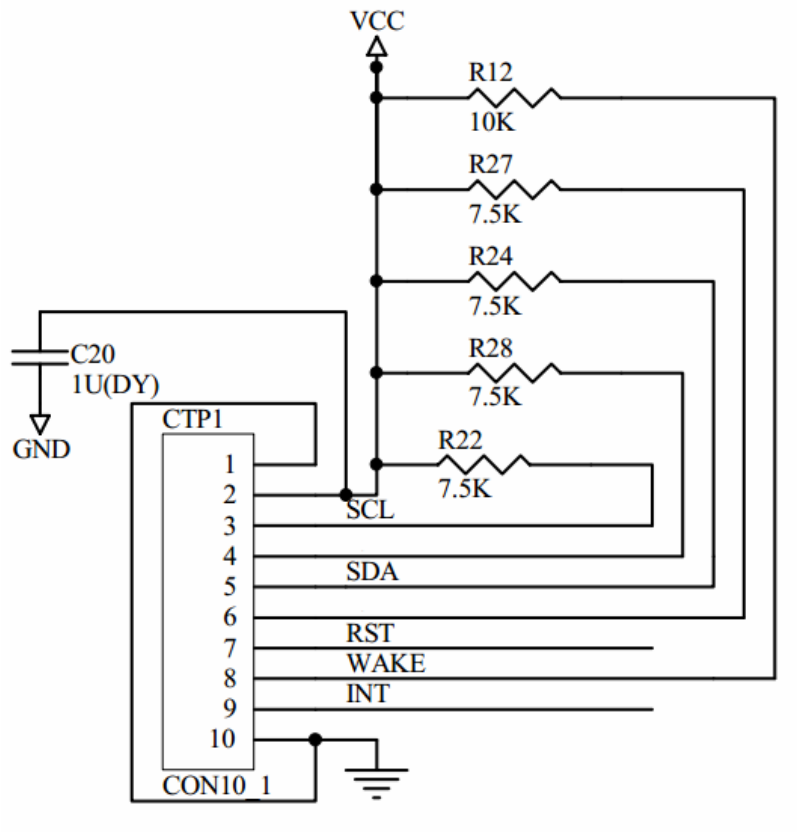


Reset/Wakeup sequence

Table 7-1 Power on/Reset/Wakeup parameters

Parameter	Description	Min	Max	Unit
Tris	Rise time from 0.1VDD to 0.9VDD	-	10	ms
Tpon	Time of starting to report point after powering on	300		ms
Trsi/ Twai	Time of starting to report point after resetting/ waking	300		ms
Trst/ Twak	Reset/Wakeup time	5		ms

8. Application Circuit



9. Reference Initial code

```
#include<c8051F340.h>
#include<stdio.h>
#include<string.h>

#define   SDA       0x80
#define   SCL       0x40
#define   TP_INT    0x04
#define   TP_RST    0x07


typedef struct
{
        unsigned char X1[2];
        unsigned char Y1[2];
        volatile int INT;
}Touch_P;

Touch_P TP;

void Reset_TP(void);
void Proccess_TP_INT(void);
int Read_Touch_Point(unsigned char Point);
void Init_I2C(void);
int I2C_Send_Write(unsigned char Addr, unsigned char Addr_1);
void I2C_Send_Byte(unsigned char Data);
void I2C_Clock(void);
void I2C_Start(void);
void I2C_Stop(void);
void I2C_Ack(void);
unsigned char I2C_Get_Byte(void);
int I2C_Send_Read(unsigned char Addr);
int I2C_Send_Read_Hex(unsigned char Addr);


int main(void)
{
        Reset_TP();
        Init_I2C();
        TP.INT = 0;

        while(1)
        {
                        Proccess_TP_INT();
        }
}
```

```
void Reset_TP(void)
{
	P1 &= ~TP_RST;
	Delay(30);
	P1 |= TP_RST;
	Uart0_Send_String("TP Test\r\n");
}

void Proccess_TP_INT(void)
{
	unsigned char TP_Point;

	if((P1 & TP_INT) && TP.INT == 0)
	{
			TP.INT = 1;
	}

	if((!(P1 & TP_INT)) && TP.INT == 1)
	{
			TP_Point = I2C_Send_Read(0x02);
			Read_Touch_Point(TP_Point);
			TP.INT = 0;
	}
}

int Read_Touch_Point(unsigned char Point)
{
	int i,reg = 0x03;

	for(i = 0;i < Point;i++)
	{
			TP.X1[0] = (I2C_Send_Read(reg++) & 0xCF);
			TP.X1[1] = I2C_Send_Read(reg++);

			TP.Y1[0] = (I2C_Send_Read(reg++) & 0x0F);
			TP.Y1[1] = I2C_Send_Read(reg);

			reg += 3;
	}
	return 1;
}

void Init_I2C(void)
{
	P0MDOUT |= 0xC0;
	P0 |= 0xC0;
}
```

```
int I2C_Send_Write(unsigned char Addr, unsigned char Addr_1)
{

	I2C_Start();
	I2C_Send_Byte(0x70);
	I2C_Ack();
	I2C_Send_Byte(Addr);
	I2C_Ack();
	I2C_Stop();

	I2C_Start();
	I2C_Send_Byte(0x70);
	I2C_Ack();
	I2C_Send_Byte(Addr);
	I2C_Ack();
	I2C_Send_Byte(Addr_1);
	I2C_Ack();
	I2C_Stop();

	return 0;
}

int I2C_Send_Read(unsigned char Addr)
{
	unsigned char Buff;

	I2C_Start();
	I2C_Send_Byte(0x70);
	I2C_Ack();
	I2C_Send_Byte(Addr);
	I2C_Ack();
	I2C_Stop();

	I2C_Start();
	I2C_Send_Byte(0x71);
	I2C_Ack();
	Buff = I2C_Get_Byte();
	I2C_Stop();

	return Buff;
}

int I2C_Send_Read_Hex(unsigned char Addr)
{
	unsigned char Buff;

	I2C_Start();
	I2C_Send_Byte(0x70);
	I2C_Ack();
	I2C_Send_Byte(Addr);
```

```
	I2C_Ack();
	I2C_Stop();

	I2C_Start();
	I2C_Send_Byte(0x71);
	I2C_Ack();
	Buff = I2C_Get_Byte();
	I2C_Stop();

	Uart0_Send_Byte(Un_Zip(Buff,1));
	Uart0_Send_Byte(Un_Zip(Buff,0));

	return Buff;
}

void I2C_Clock(void)
{
	Delay(1);
	P0 |= SCL;
	Delay(1);
	P0 &= ~SCL;
	Delay(1);
}

void I2C_Start(void)
{
	Delay(1);
	P0 &= ~SDA;			//Start
	Delay(1);
	P0 &= ~SCL;
	Delay(1);
}

void I2C_Stop(void)
{
	Delay(1);
	P0 &= ~(SCL | SDA);
	Delay(1);

	P0 |= SCL;
	Delay(1);
	P0 |= SDA;
	Delay(1);
}

void I2C_Ack(void)
{
	P0 &= ~SDA;
	Delay(1);
	P0 |= SCL;
```

```
	Delay(1);
	P0 &= ~SCL;
	Delay(1);
	P0 |= SDA;
	Delay(1);
}

void I2C_Send_Byte(unsigned char Data)
{
	int i;

	for(i = 0;i < 8;i++)
	{
			if(1 == ((Data << i) / 0x80))
				P0 |= SDA;

			if(0 == ((Data << i) / 0x80))
				P0 &= ~SDA;

			I2C_Clock();
	}
}

unsigned char I2C_Get_Byte(void)
{
	unsigned char I2C_Data = 0x80,Buff = 0x00;
	int i;

	for(i = 0;i < 8;i++){
			Delay(1);
			P0 |= SCL;
			Delay(1);

			if(P0 & 0x80)
				Buff |= I2C_Data;

			I2C_Data = (I2C_Data >> 1);
			Delay(1);
			P0 &= ~SCL;
	}
	return Buff;
}
```