

Project Number F181

Abstract

Why do we need a touch screen controller? Most of the embedded systems in use these days have some kind of a GUI. The trend seems to be to emulate desktop GUI for the convenience of the user. Hence the touch screen is a necessary item today. We will talk about the analog resistive touch screens and a touch screen controller implemented in an 8-pin Motorola 68HC098QT4. The picture and the block diagram below show the concept. You just need to add the controller IC between the touch screen and your system.

The controller converts the user touch inputs to RS232 data and sends it to your system. So what are the advantages of using the touch screen controller? These come to mind immediately.

1. Reduce development time, faster to market due to the modular approach. Partitioning system tasks is always a good idea for keeping to the project schedules and effective code and hardware reuse. Compared to the cost of firmware, hardware is cheap in many situations.
2. Reduce system and software complexity. Easy integration due to the fact that the controller sends messages to the main processor via the RS232 interface (no level shifting needed since the interface is digital).
3. Retrofit an existing system easily (with just a RS232 parser update).
4. Lack of ADC's on the main processor.
5. Current drive capability of the main processor.
6. A high volume application.

The complete code is included with the project. Here is some code samples from the project.

```
/* MODULE Cpu. */
/* =====
** Function : Cpu_Interrupt
** Description : The routine to trap the unused interrupt
** =====
__interrupt void Cpu_Interrupt(void)
{
}
/*****
Function1 : Init_Low_level
Inputs : None
Outputs : None
Author : Deleted by censorship bureau
Date : 04/17/2003
Description : Initialization of CPU
Changes : None
Status : : Tested OK
*****/
void __low_level_init(void)
{
/* ### MC68HC908QT4 "Cpu" init code ... */
/* CONFIG1: COPRS=1,LVISTOP=0,LVIRSTD=0,LVIPWRD=0,LVI5OR3=1,SSREC=1,STOP=1,COPD=1 */
CONFIG1 = 0x8f; /* Set the CONFIG1 register */
/* CONFIG2: IRQPUD=0,IRQEN=0,Unused=0,OSCOPT1=0,OSCOPT0=0,Unused=0,Unused=0,RSTEN=0 */
CONFIG2 = 0; /* Set the CONFIG2 register */
OSCSTAT = 0; /* Disable the external oscillator */
OSCTRIM = *(byte*)0xFFC0; /* Initialize OSCTRIM register from a non volatile memory */
__EI(); /* Disable interrupts */
}
/* END Cpu. */
*****/
Function1 : AD1_MainMeasure
Inputs : ADC Channel Selection
Outputs : Measured ADC value
```

Author : Deleted by censorship bureau

Date : 04/17/2003

Description : ADC measurement

Changes : None

Status: : Tested OK

*****/

```
static byte AD1_MainMeasure(byte Channel)
```

```
{
  ADSCR = Channels[Channel]; /* Start ADC measurement */
  while (!ADSCR_COCO); /* Wait for AD conversion complete */
  AD1_OutV[Channel] = ADR; /* Save measured value */
  ADSCR = 31;
  return AD1_OutV[Channel];
}
```

Function1 : Y_Measure_init

Inputs : None

Outputs : None

Author : Deleted by censorship bureau

Date : 04/17/2003

Description : Initialization for Y coordinate measurement

Changes : None

Status: : Tested OK

*****/

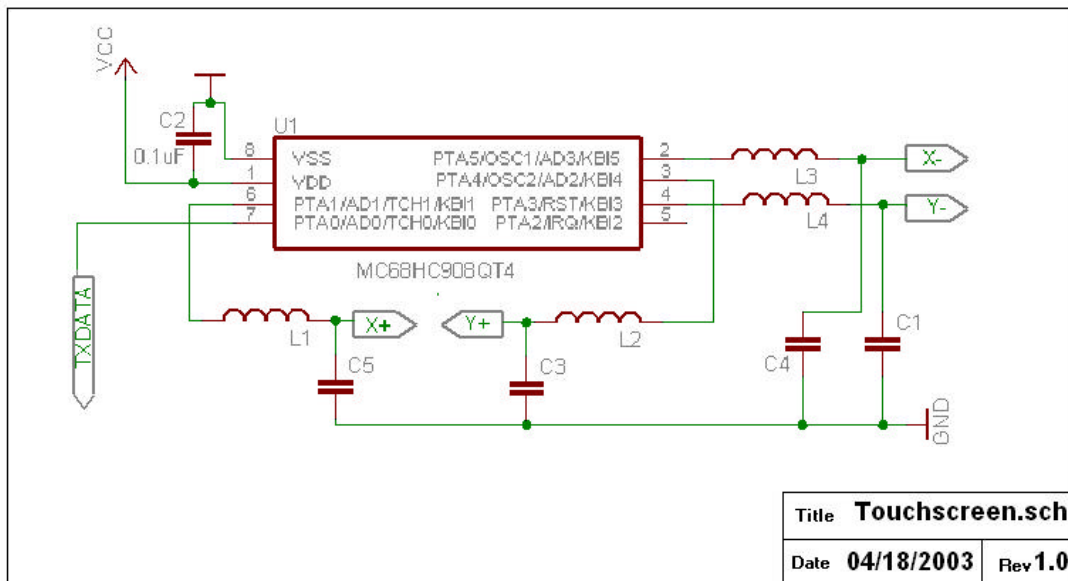
```
void Y_Measure_Init(void)
```

```
{
  /* PTA: PTA4=0, PTA3=0 : Initial value*/
  output( PTA, 0 );
  /* DDRA: DDRA4=1, DDRA3=1,DDRA1=0, DDRA5=0 */
  output( DDRA, 24 );
  /* Set PTA4 */
  PTA_PTA4 = 1;
}
```



Block diagram and schematic are in the PDF

Circuit diagram



Debugging



Firmware design

The statechart below depicts the system operation.

