



## User Manual for P Series (SPI Interface)

**3.5", 4.3", 5.0", 7.0"**

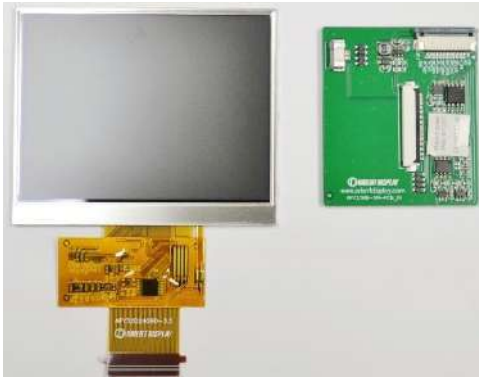
Revision 0

Orient Display
TFT with SPI Interface
TN Display, 12 o' clock Viewing Direction
Medium Brightness
Top: -20~+70°C; Tstr: -30~+80°C
RoHS compliant
Controller <a href="#">NP4185</a>

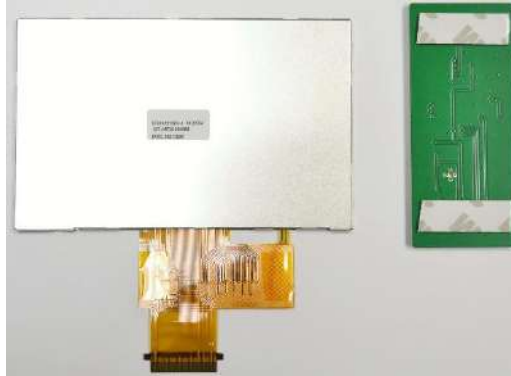
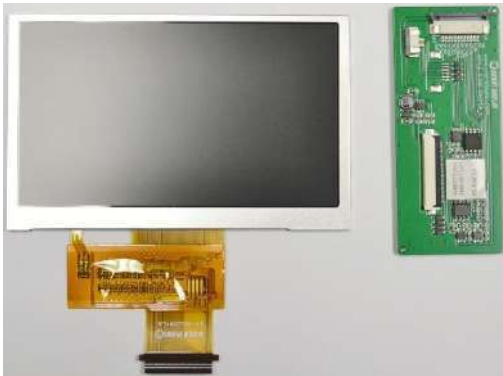


# P SERIES (SPI INTERFACE) USER MANUAL

**AFY320240B0-3.5N12NTM-SPI:**



**AFY480272B0-4.3N12NTM-SPI:**



**AFY800480B1-5.0N12NTM-SPI:**



**AFY800480B1-7.0N12NTM-SPI:**



## Product Feature

1. Interface: 4-line SPI, an additional line for logic control, total 5 GPIO.
2. Touch panel: RTP and CTP
3. The flash on board for storing UI pictures can be mapped to the flash on master board, user can directly use the flash on master board through the SPI interface, in this way the content in board can be updated remotely.  
And PCBA also is also including FLASH memory, drive circuit for programmable adjustment of backlight brightness, and power management circuit; RTP/CTP control circuit.
4. Support up to 800\*480 resolution.
5. The way to use is similar with the serial port screen with the instruction, and the UI development can be completed by sending relevant instructions and control information through the SPI interface.
6. Operation voltage: 5.0V  
Logic voltage is 3.3V (TTL)

## Product Feature

Compared with the traditional serial screen, there are below major differences

1. There are only two instructions, an ordinary packet instruction, and a dedicated instruction for quick point tracing. The instruction is very simple and accurate.
2. The onboard FLASH for storing UI pictures can support up to 128Mbit.
3. The onboard FLASH can be configured as an external FLASH of the user MCU through software, and the user MCU can directly operate the FLASH through the SPI interface, so that the content in the UI FLASH can be updated in the background.

## The area of applications

Mainly for small appliances, smart home, control industry, etc.

## Instructions description

```
//*****//  
// SPI timing request 1: when FLASH_SW=1, it is equivalent that the flash is the user's MCU interface. The SPI timing is the standard 4-wire timing. At  
this time, the SPI FLASH is directly operated, and the SPI flash is readable and writable.  
  
// SPI timing request 2: When FLASH_SW=0, the SPI timing is also a standard 4-wire timing but has no readback function, and the busy flag must be  
judged after the command/data is sent (output by the SPI_MISO pin), and the send command is as follows:  
  
// 1. First set FLASH_CS to be 0 (First, FLASH_CS is pulled low)  
// 2. Send commands (ie 12 configuration parameters), if you need to follow the display data, continue to send the display data  
  
// 3. After send commands are all finished, the display controller inside needs to be processed. For this purpose, the busy flag status must be judged  
(FLASH_MISO=1 is busy)  
  
// When judging the busy flag, an exit mechanism is required to prevent the controller from waiting when the set parameters do not correspond to  
the written parameters.  
  
// The exit mechanism is an appropriate delay. It is recommended that the delay reaches 120ms and is still busy, then it is forced to exit. Please refer  
to SPI_Check_Busy()  
  
// 4. at last, set FLASH_CS to be 1 (Final FLASH_CS is pulled high)  
  
// PS 1: The FLASH_SW pin of the display controller is pull-down by default. If you do not need to operate the SPI FLASH, you can not process this
```

pin

```
// PS 2: For the user board, configure FLASH_MISO as the busy flag input detection pin

// PS 3: The maximum input buffer of the display controller is 4096 bytes. If the one-time input is larger than 4096 bytes, it needs to be split and
transferred.

//***** The following are several basic application functions *****//

// Calculate the cache address according to the coordinate point

//uint32_t Get_XY2DramAddr(uint16_t X,uint16_t Y)
//{
//  uint32_t Dram_addr;
//  Dram_addr = ( X+Y*1536 )*2;  // Each line of bytes is fixed to 1536*2 bytes

//  return(Dram_addr);
//}
//-----
// The description of common command parameters is as follows (12 parameters need to be sent at one time for each command, that is, sent in a
function package): If you need to modify the screen brightness, you can modify it at the same time as you are refreshing the screen

//void Set_NP4185_Comd(uint8_t COMD,uint16_t Dram_addr,uint16_t Flash_addr,uint16_t H_size,uint16_t V_size,uint8_t BL_PWM)
//{
//  FLASH_ReadWriteByte( COMD );          // Command[0x01:User SPI data to Dram to display = The user SPI directly writes the display data to
the display cache and refreshes the screen]
//                                          // [0x04:Flash data to Dram to display = Read data from flash and write to display cache refresh]

//  FLASH_ReadWriteByte( (uint8_t)(Dram_addr>>16) );    // Dram_address[19:16] Specify the user's SPI interface to start writing display data to
a certain address in the display cache and directly refresh the screen

//  FLASH_ReadWriteByte( (uint8_t)(Dram_addr>>8) );      // Dram_address[15:8]   Specify the user's SPI interface to start writing display data
to a certain address in the display cache and directly refresh the screen

//  FLASH_ReadWriteByte( (uint8_t)(Dram_addr) );         // Dram_address[7:0]    Specify the user's SPI interface to start writing display data to
a certain address in the display cache and directly refresh the screen
//  FLASH_ReadWriteByte( (uint8_t)(Flash_addr>>16) );    // flash_address[19:16] Specify the controller to read display data from a certain
address of SPI FLASH for display. When the user SPI interface operates the display buffer, this address is invalid (can be any value)
//  FLASH_ReadWriteByte( (uint8_t)(Flash_addr>>8) );     // flash_address[15:8] Specify the controller to read display data from a certain
address of SPI FLASH for display. When the user SPI interface operates the display buffer, this address is invalid (can be any value)

//  FLASH_ReadWriteByte( (uint8_t)(Flash_addr) );       // flash_address[7:0]   Specify the controller to read display data from a certain address
of SPI FLASH for display. When the user SPI interface operates the display buffer, this address is invalid (can be any value)

//  FLASH_ReadWriteByte( (uint8_t)((V_size-1)>>8) );    // V_size-1[15:8]      set the quantity of pixel in perpendicular
//  FLASH_ReadWriteByte( (uint8_t)(V_size-1) );        // V_size-1[7:0]
//  FLASH_ReadWriteByte( (uint8_t)((H_size-1)>>8) );    // H_size-1[15:8]      set the quantity of pixel in horizontal
//  FLASH_ReadWriteByte( (uint8_t)(H_size-1) );        // H_size-1[7:0]
//  FLASH_ReadWriteByte( BL_PWM );                     // Backlight lightness[255~0] set the level of backlight brightness, highest 225, and 0 is
closed backlight
//}
//-----
// Quick point drawing command

//void Set_NP4185_DotComd(uint16_t X,uint16_t Y,uint16_t color)
//{
//  uint16_t Dram_addr;
//  Dram_addr=Get_XY2DramAddr(X,Y);
//  FLASH_CS_L();
```

```

// FLASH_ReadWriteByte( 0x08 );          // Command[0x08:]
// FLASH_ReadWriteByte( (uint8_t)(Dram_addr>>16) );    // Dram_address[19:16]    Specify the user's SPI interface to start writing display data
to a certain address in the display cache and directly refresh the screen
// FLASH_ReadWriteByte( (uint8_t)(Dram_addr>>8) );      // Dram_address[15:8]    Specify the user's SPI interface to start writing display data
to a certain address in the display cache and directly refresh the screen

// FLASH_ReadWriteByte( (uint8_t)(Dram_addr) );        // Dram_address[7:0]    Specify the user's SPI interface to start writing display data to a
certain address in the display cache and directly refresh the screen
// FLASH_ReadWriteByte(color);    // Display font color


// FLASH_ReadWriteByte(color>>8);
// SPI_Check_Busy();
// FLASH_CS_H();
//}
//-----
//Busy flag judgment

//uint8_t SPI_Check_Busy(void)
//{
//    uint8_t ucErrTime=0;
//    while(FLASH_MISO_GET()==1)
//    {
//        LL_mDelay(10);
//        ucErrTime++;
//        if(ucErrTime>250)          //It is recommended to be greater than 120ms (the time to update a frame is about 100ms at 800*480 resolution)

//            return 1;
//    }
//    return 0;
//}
//*****IMPORTANT*****//
// When the user's SPI interface directly flashes the screen to the display controller (0x01 or 0x08 command), the maximum buffer of the controller's
receiving area is 4096 bytes and the processing speed is very fast, but still need to pay attention to the following:
// 1. The maximum SPI clock is 8MHz, and the display data can be continuously written in bursts at or below 8MHz
// 2. Use the function of drawing points as little as possible. After all, it is slow to send data one point at a time with coordinate parameters, but there
is a trick: if it is a continuous area, the displayed data can be pre-made into a Display area blocks, send in bursts at once, very fast

```

[illegible]



[illegible]



[illegible]