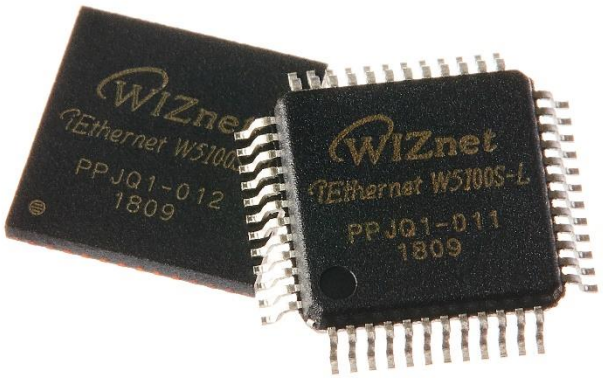
How to use Interrupt Application Note

Version 1.0.0



For more information, visit our website at http://www.wiznet.io

Contents

List of Figures

List of Tables

Introduction

W5100S provides 1 Interrupt Pin (INTn) and HOST can know when an Ethernet Communication Event has occurred with INTn. When an Ethernet Communication Processing Event (IP Collision, WOL Magic Packet Reception, Data Transmission, Reception for each SOCKET, etc.) occurs, the INTn is asserted low.

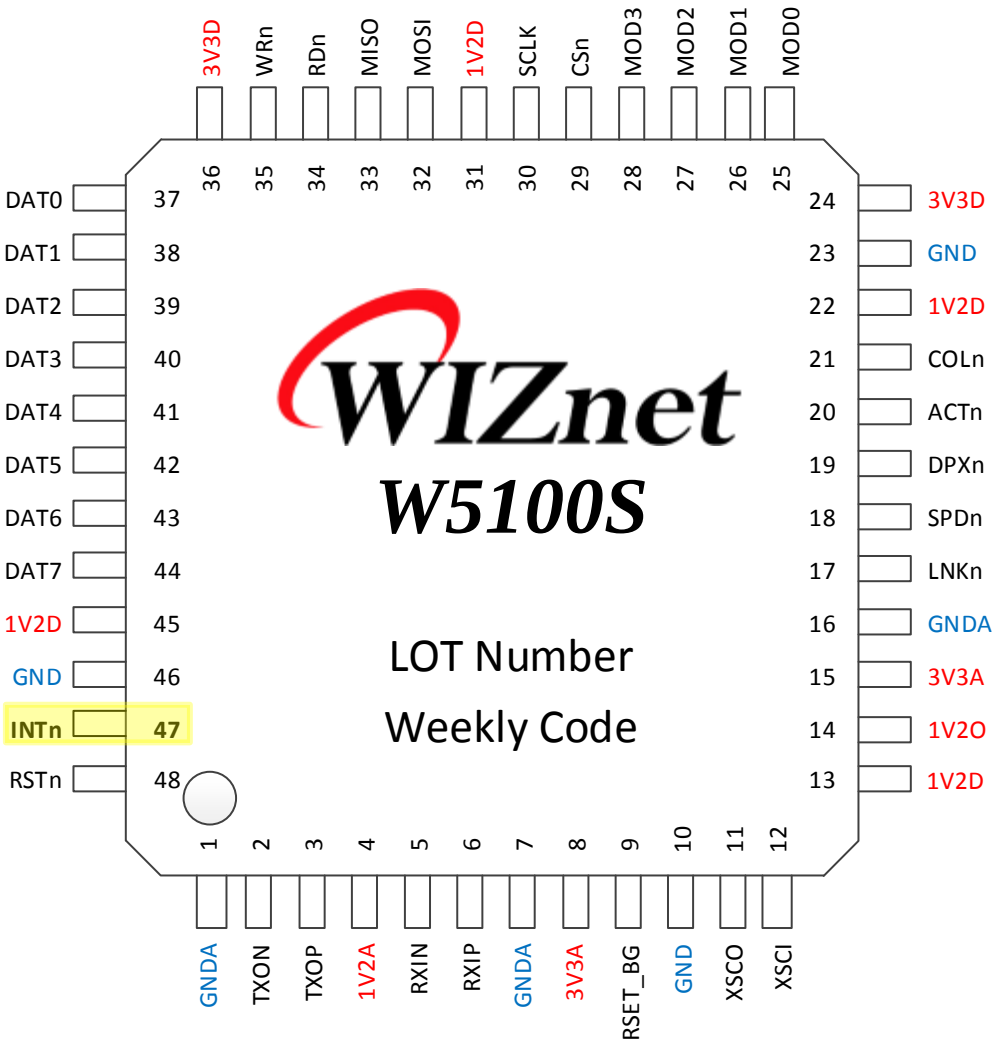


Figure 1 W5100S INTn Pin

INTn is enabled by default and it can be configured by setting IEN Bit in MR2 (Mode Register 2).

1. Interrupt configuration

1.1 Registers related to interrupt.

Table 1 Interrupt related registers

Symbol	Address	Description	Remarks
INTPTMR	0x0013-14	Interrupt Pending Time Register	
IR	0x0015	Interrupt Register	
IMR	0x0016	Interrupt Mask Register	
IR2	0x0020	Interrupt Register 2	
IMR2	0x0021	Interrupt Mask Register 2	
MR2	0x0030	Mode Register 2	
SLIR	0x005E	SOCKET-less Command Interrupt Register	
SLIMR	0x005F	SOCKET-less Command Interrupt Mask Register	
Sn_IR	0x0403+0x100*n	SOCKET n Interrupt Register	
Sn_IMR	0x042C+0x100*n	SOCKET n Interrupt Mask Register	

Table 1 shows the Registers that associated with Interrupt functions. Please see the W5100S Datasheet for the detail description of each Register. There are three Types of Interrupt Registers. First is IR (Interrupt Register). Interrupt Register describes Event Occurrence. Second is IMR (Interrupt Mask Register). IMR Bit corresponds to 1:1 to IR Bit. If the corresponding IMR Bit is set to '1', INTn is asserted to Low when the corresponding Event occurs. Lastly, there is MR2 (Mode Register2) that enables and disables the INTn. INTn can be asserted to Low if the MR2 [IEN] is set to '1' when an Interrupt occurs.

1.2 Operation of INTn

The INTn indicates to HOST whether an Event occurred. INTn is asserted to Low when Event occurs. In this time, if the internal counter set by INTPTMR is not 0, INTn is not asserted to Low until the internal counter becomes 0. When the Event Processing in HOST is completed, the Interrupt can be cleared by setting the corresponding IR Bit to '1'. Then INTn is de-asserted to High.

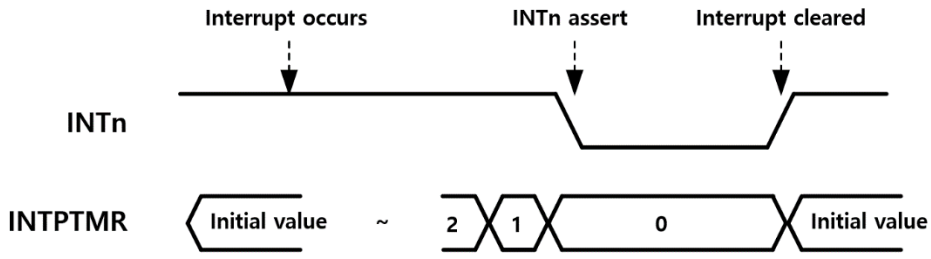


Figure 2 Interrupt and INTPTMR

1.3 Common interrupt

W5100S can generate IP Conflict Interrupt, Port Unreachable Interrupt, PADT/LCPT Receive Interrupt and SOCKET n Event Interrupt via IR. These Interrupts are enabled by setting each Bit in IMR. If IMR Bit is not set to '1', INTn cannot be asserted to Low even if IR Bit changes to '1'.

W5100S can generate MAGIC Packet Receive Interrupt through IR2. This Interrupt is enabled by setting the corresponding bit in IMR2.

For example, for IP Conflict Interrupts, follow the procedure below.

• Register Configuration

```
{
start:
    MR2 |= 1<<6;    // enable INTn (MR2[IEN] == 1)
    IMR |= 1<<7;    // enable CONFLICT Interrupt Mask Register
end
}
```

• IP Conflict Error occurred (In HOST's Interrupt Handler)

```
{
start:
    if(IR && 1<<7) //check IR[CONFLICT] Interrupt flag
    //Do Something!
    IR |= 1<<7; //clear CONFLICT Interrupt flag
end
}
```

1.4 SOCKET Interrupt

W5100S provides an Interrupt to detect the Status Event of each SOCKET. This Interrupt Sn_IR is enabled by setting the corresponding bit in Sn_IMR to '1'. If a SOCKET Event that configured in Sn_IMR occurs, the corresponding bit in Sn_IR and Sn_INT Bit in IR are set to

‘1’. In this time, if IEN Bit in MR2 or corresponding Interrupt Mask bit in IMR is disabled, INTn cannot be asserted to Low. When INTn is asserted to Low by a SOCKET Event, HOST must figure out which SOCKET generates the Event through IR and which Interrupt occurred through Sn_IR.

For example, if HOST want to handle Receive Interrupt of SOCKET 0, HOST must follow the procedure as follows.

• Register Configuration

```
{
start:
    MR2 |= 1<<6;    //enable MR2[IEN] – enable INTn
    IMR |= 1<<0;     // enable IMR[S0_INT] – enable SOCKET 0 Interrupt
    S0_IMR |= 1<<2; // enable RECV Interrupt Mask Bit
end
}
```

• RECV Event occurred (HOST Interrupt Handler that connected to INTn)

```
{
start:
    if(IR && 1<<0)   // SOCKET 0 Interrupt occurs?
    if(S0_IR && 1<<2) // RECV Interrupt occurs?
    //Do Something!
    IR |= 1<<0;     // clear SOCKET 0 Interrupt Bit
    S0_IR |= 1<<2; //clear SOCKET 0 RECV Interrupt Bit
end
}
```

There are 5 interrupts for the socket status. Those are SENDOK, TIMEOUT, RECV, DISCO, CON. For details, refer to W5100S Datasheet.

1.5 SOCKET-less Command Interrupt

W5100S has functions to transmit PING and ARP Packet, called SOCKET-less Command. SOCKET-less Command generates three kinds of Interrupt through SLIR (SOCKET-less Interrupt Register) - TIMEOUT, PING, ARP. This SLIR is enabled by setting the corresponding bit in SLIMR to ‘1’. When a SOCKET-less Event occurs, the corresponding bit in SLIR is set to ‘1’ and INTn is asserted to Low.

For example, if HOST want handle PING Interrupt, HOST must follow the procedure as follows.

• Register Configuration

```
{
start:
    MR2 |= 1<<6;    //enable MR2[IEN] - enable INTn
    SLIMR |= 1<<0; //enable SLIMR[SLCMD_PING] - enable PING Interrupt
end
}
```

• PING Interrupt Handler

```
{
start:
    if(SLIR && 1<<0) //PING Interrupt ?
    //Do Something!
    SLIR |= 1<<0;    //clear PING Interrupt flag
end
}
```

For details, refer to the W5100S Datasheet.

2. HOST-side Interrupt Handling

2.1 HOST-side Interrupt Configuration

HOST refers to MCU (Micro Controller Unit) connected to W5100S. The MCU has components that can detect pin state or pin state changes called External Interrupt unit. To configure the External Interrupt, Hardware Initialization and Interrupt Handler are required. The Interrupt Handler is the Software that is executed in the MCU when an Event occurs on the W5100S.

2.2 Packet Receive Interrupt Example

This example uses STM32F1XX MCU and ioLibrary. (Official library for WIZnet Ethernet IC) Assume that INTn signal connected to the GPIOB 1 pin of the STM32F1XX and RECV Interrupt enabled

• Initialize External Interrupt Hardware Unit

```
{
    void InitializeExternalInterrupt(void)
    {
        GPIO_InitTypeDef GPIO_InitStructure;
        EXTI_InitTypeDef EXTI_InitStructure;

        /* GPIO initialize */
        GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
        GPIO_InitStructure.GPIO_Pin = GPIO_Pin_1;
        GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
        GPIO_Init(GPIOB, & GPIO_InitStructure);

        /* External interrupt initialize */
        GPIO_EXTILineConfig(GPIOB_PortSourceGPIOB, GPIO_PinSource1);
        EXTI_InitStructure.EXTI_Line = EXTI_Line8;
        EXTI_InitStructure.EXTI_Mode = EXTI_Mode_Trigger_Falling;
        EXTI_InitStructure.EXTI_LineCmd = ENABLE;
        EXTI_Init(&EXTI_InitStructure);
    }
}
```

• W5100S Interrupt Configuration

```
{
```

```
    MR2 |= 1<<6;    // MR2[6] IEN Bit - enable INTn
    IMR |= 1<<0;    // IMR[0]S0_INT Bit - enable SOCKET 0 Interrupt
}
```

• Interrupt Handler

```
{
    void EXTI1_IRQHandler(void)
    {
        //setMR2(getMR2() & ~(1<<6));    //Global Interrupt Disable
        if(EXTI_GetITStatus(EXTI_Line1) == SET) //check the Interrupt
        {
            //(1)Set Global Interrupt flag
            interruptflag = 1;
            setSn_IR(0xff);    //clear SOCKET n Interrupt
        }
        //clear External Interrupt flag
        EXTI_ClearFlag(EXTI_Line1);
    }
  (1) It is not recommended to execute too many commands in the interrupt handler. This
  can cause serious problem to your system. Just sets the flag in the interrupt handler and
  executes the functions outside the interrupt handler.
}
```

2.3 SOCKET-less Command Interrupt Example

This example uses STM32F1XX MCU and ioLibrary. (Official library for WIZnet Ethernet IC) Assume that INTn connected to the GPIOB 1 pin of the STM32F1XX and RECV Interrupt enabled.

External Interrupt Configuration is the same with *2.2 Packet Receive Interrupt Example*.

• Interrupt Handler

```
{
    void EXTI1_IRQHandler(void)
    {
        //setMR2(getMR2() & ~(1<<6)); //Global Interrupt Disable
        if(EXTI_GetITStatus(EXTI_Line1) == SET) //check the Interrupt
        {
```

```
                    //(1)Set Global Interrupt flag
                    interruptflag = 1;
                    setSLIR(0xff);    //clear SOCKET Interrupt
            }
            //clear External Interrupt flag
            EXTI_ClearFlag(EXTI_Line1);
        }
    (1) It is not recommended to execute too many commands in the interrupt handler. This
    can cause serious problem to your system. Just sets the flag in the interrupt handler and
    executes the functions outside the interrupt handler.
}
```

3. ETC

3.1 Precautions when using the interrupt

The W5100S SPI Frame consists of at least 3 Bytes. They are Address Phase, Control Phase, Data Phase in order. This order must be followed.

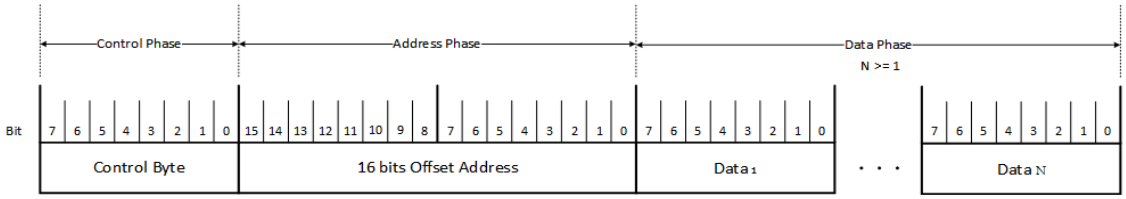


Figure 3 SPI Frame

But Interrupt can break this SPI frames. Let's suppose an Interrupt occurs while HOST are reading something from W5100S (During SPI frame). And if there are any Code to control SPI signal in the Interrupt Handler, it breaks SPI frame. For avoid this Problem, the Interrupt Handler Function must not contain any Code related to SPI signal or Interrupts must be prevented before the end of SPI frame.

This Frame called 'Critical Section'. Using reg_wizchip_cris_cbfunc() Function, behavior to be performed in the 'Critical Section' can be registered.

3.2 Precautions when using the RTOS

Using the RTOS may cause the same Problem in *3.1 Precautions when using the interrupt*. If the Task switched in the middle of a Frame Operation and the same resource is used, the Frame may be corrupted. For prevent this, Task switching of RTOS must be stopped while the Frame is in Progress. Or Mutex, Semaphore Function of the RTOS must be used.

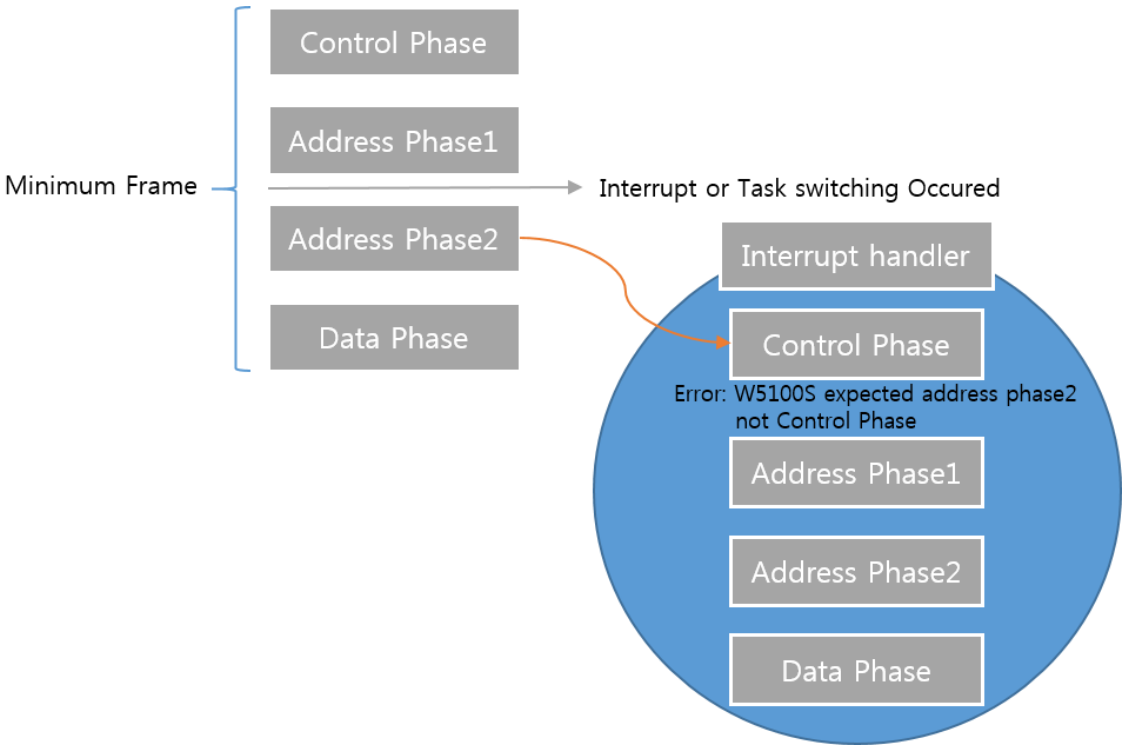


Figure 4 Behavior when an interrupt or task switching occurs

4. Document Revision History

Version	Date	Descriptions
Ver. 1.0.0	1APR2018	Initial Release