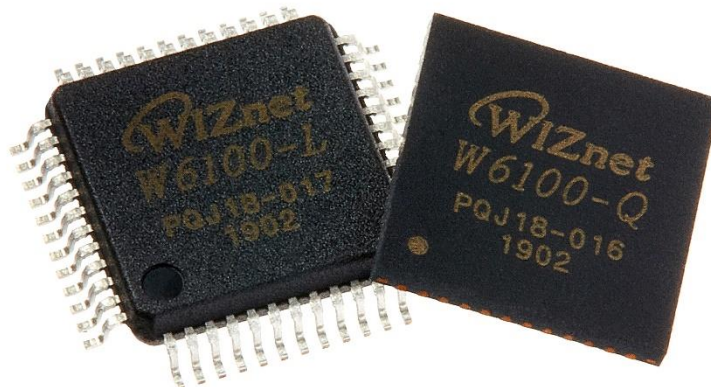


How to use Interrupt Application Note

Version 1.0.0



© 2019 WIZnet Co., Ltd. All Rights Reserved.

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

Contents

1. Interrupt configuration	6
1.1 Registers related to interrupt.....	6
1.2 Operation of INTn	6
1.3 Common interrupt	7
1.4 SOCKET Interrupt.....	7
1.5 SOCKET-less Command Interrupt.....	8
2. HOST-side Interrupt Handling	10
2.1 HOST-side Interrupt Configuration	10
2.2 Packet Receive Interrupt Example	10
2.3 SOCKET-less Command Interrupt Example	11
2 ETC	13
3.1 Precautions when using the interrupt	13
3.2 Precautions when using the RTOS	13
3 Document Revision History	15

List of Figures

Figure 1 W6100 INTn Pin.....	5
Figure 2 Interrupt and INTPTMR register	7
Figure 3 SPI Frame.....	13
Figure 4 Behavior when an interrupt or task switching occurs	14

List of Tables

Table 1 Interrupt related registers	6
---	---

Introduction

W6100 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.

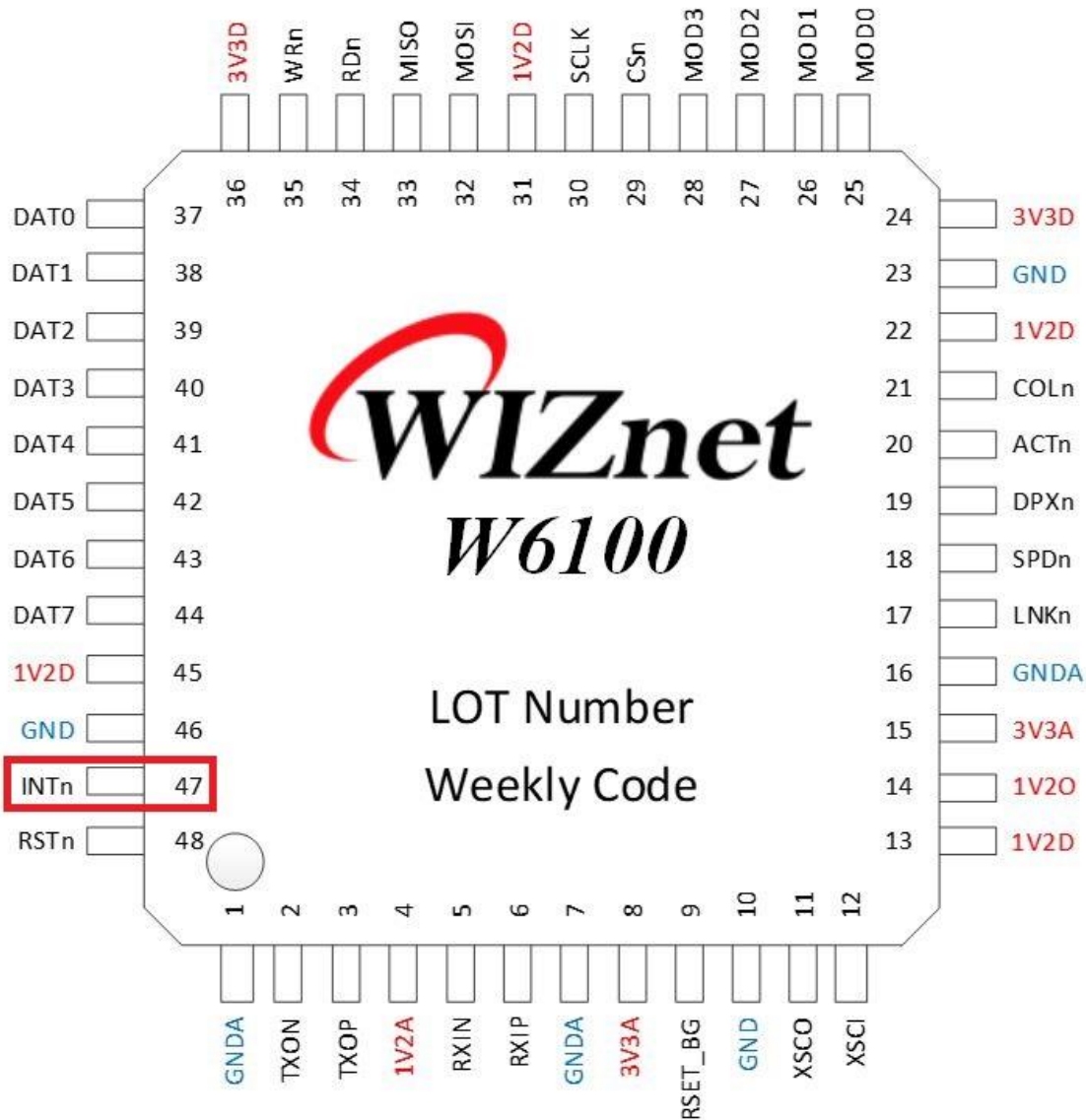


Figure 1 W6100 INTn Pin

INTn is enabled by default and it can be configured by setting IEN Bit in SYCR1 (System Config Register 1).

1. Interrupt configuration

1.1 Registers related to interrupt.

Table 1 Interrupt related registers

Symbol	Address Offset	Description
SYCR1	0x2005	System Config Register 1
IR	0x2100	Interrupt Register
SIR	0x2101	SOCKET Interrupt Register
SLIR	0x2102	SOCKET-less Interrupt Register
IMR	0x2104	Interrupt Mask Register
IRCLR	0x2104	IR Clear Register
SIMR	0x2105	SOCKET Interrupt Mask Register
SLIMR	0x2106	SOCKET-less Interrupt Mask Register
SLIRCLR	0x2108	SLIR Clear Register
INTPTMR	0x41C5-0x41C6	Interrupt Pending Time Register
Sn_IR	0x0020	SOCKET n Interrupt Register
Sn_IMR	0x0024	SOCKET n Interrupt Mask Register
Sn_IRCLR	0x0028	Sn_IR Clear Register

Table 1 shows the Registers that associated with Interrupt functions. Please see the W6100 Datasheet for the detail description of each Register. There are four 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 SYCR1 (Mode Register2) that enables and disables the INTn. INTn can be asserted to Low if the SYCR1 [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.

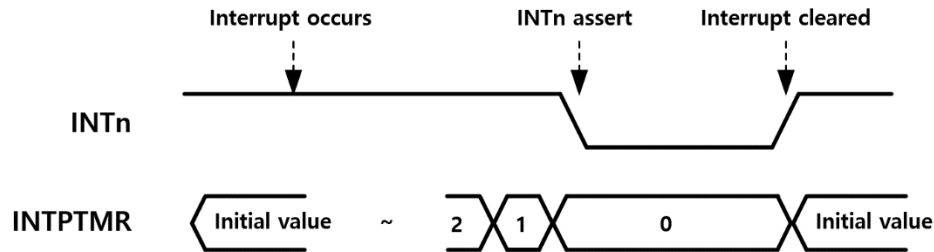


Figure 2 Interrupt and INTPTMR

1.3 Common interrupt

W6100 can generate MAGIC Packet Receive Interrupt, IP Conflict Interrupt, Port Unreachable Interrupt and PADT/LCPT Receive 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'.

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

- Register Configuration

```
{
start:
    SYCR1 |= 1<<7; // enable INTn (SYCR1[IEN] == 1)
    IMR |= 1<<2; // enable CONFLICT Interrupt Mask Register
end
}
```

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

```
{
start:
    if(IR && 1<<2) //check IR[CONFLICT] Interrupt flag
        //Do Something!
    IRCLR |= 1<<2; //clear CONFLICT Interrupt flag
end
}
```

1.4 SOCKET Interrupt

W6100 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 SIR are set to '1'. In this time, if IEN Bit in SYCR1 or corresponding Interrupt Mask bit in SIMR 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 SIR 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:
    SYCR1 |= 1<<7; //enable SYCR1[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_IRCLR |= 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 W6100 Datasheet.

1.5 SOCKET-less Command Interrupt

W6100 has functions to transmit ND(Neighbor Discovery), PING and ARP Packet, called SOCKET-less Command. SOCKET-less Command generates kinds of Interrupt through SLIR (SOCKET-less Interrupt Register) - RA, RS, NS, PING6, ARP6, PING4, ARP4, TOUT. 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:
    SYCR1 |= 1<<7;  //enable SYCR1[IEN] - enable INTn
    SLIMR |= 1<<5;  //enable SLIMR[PING4] - enable PING4 Interrupt
end
}
```

- PING Interrupt Handler

```
{
start:
    if(SLIR && 1<<5) //PING Interrupt ?
        //Do Something!
    SLIRCLR |= 1<<5;  //clear PING4 Interrupt flag
end
}
```

For details, refer to the W6100 Datasheet.

2. HOST-side Interrupt Handling

2.1 HOST-side Interrupt Configuration

HOST refers to MCU (Micro Controller Unit) connected to W6100. 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 W6100.

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);
    }
}
```

- W6100 Interrupt Configuration

```
{
```

```
SYCR1 |= 1<<7; // SYCR1[6] IEN Bit - enable INTn
SIMR |= 1<<0; // IMR[0]S0_INT Bit - enable SOCKET 0 Interrupt
}
```

- Interrupt Handler

```
{
void EXTI1_IRQHandler(void)
{
    //setSYCR1(getSYCR1() & ~(1<<7)); //Global Interrupt Disable
    if(EXTI_GetITStatus(EXTI_Line1) == SET) //check the Interrupt
    {
        //(1)Set Global Interrupt flag
        interruptflag = 1;
        setSn_IRCLR(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)
{
    //setSYCR1(getSYCR1() & ~(1<<7)); //Global Interrupt Disable
    if(EXTI_GetITStatus(EXTI_Line1) == SET) //check the Interrupt
    {
```

```
        //(1)Set Global Interrupt flag
        interruptflag = 1;
        setSLIRCLR(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 W6100 SPI Frame consists of at least 3 Bytes. They are Address Phase, Control Phase, Data Phase in order. This order must be followed.

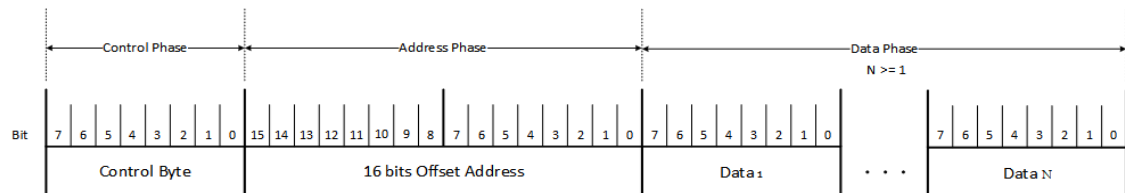


Figure 3 SPI Frame

But Interrupt can break this SPI frames. Let's suppose an Interrupt occurs while HOST are reading something from W6100 (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.

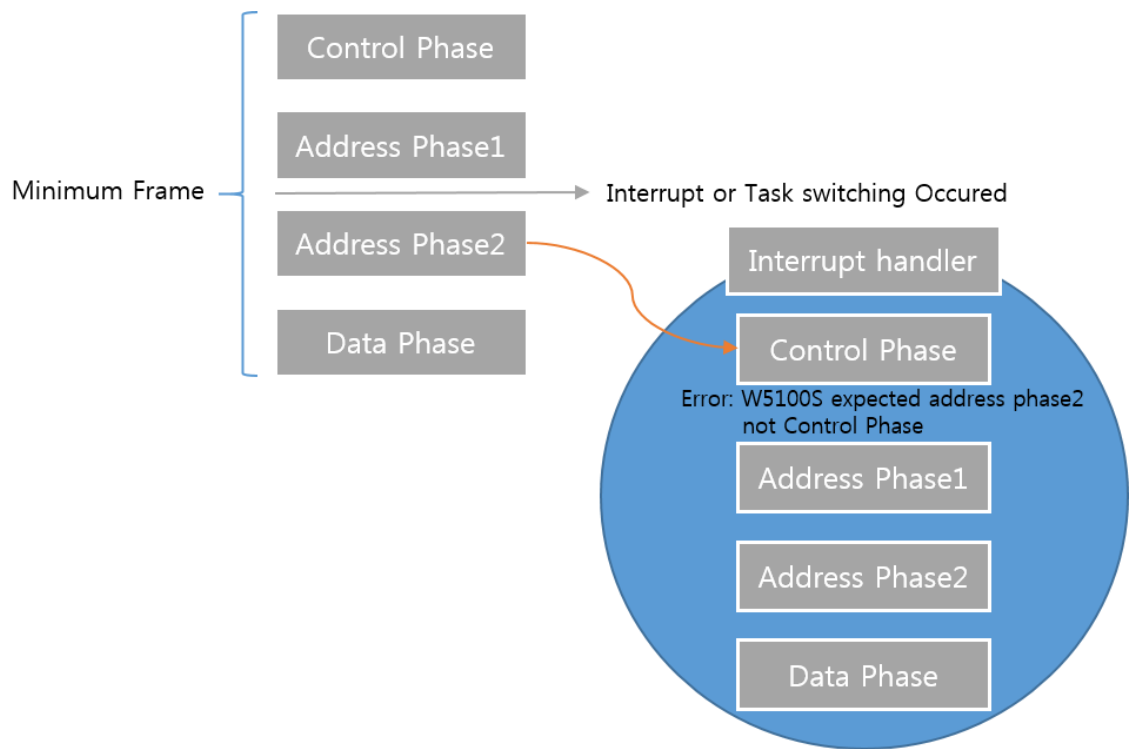


Figure 4 Behavior when an interrupt or task switching occurs

4. Document Revision History

Version	Date	Descriptions
Ver. 1.0.0	Feb, 2019	Initial Release

Copyright Notice

Copyright 2019 WIZnet Co., Ltd. All Rights Reserved.

Technical Support: <https://forum.wiznet.io/>

Sales & Distribution: <mailto:sales@wiznet.io>

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