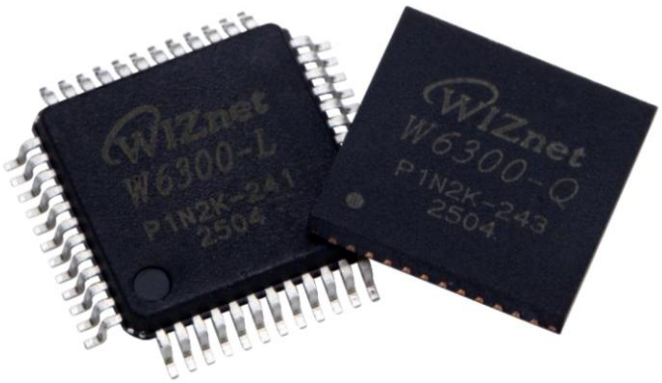
How to use Interrupt Application Note

Version 1.0.0



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

Contents

List of Figures

List of Tables

Introduction

W6300 은 1 개의 Interrupt Pin (INTn)을 제공하며 User 는 INTn 을 통해 Event 가 발생되는 시점을 정확하게 파악하여 효과적인 프로그래밍을 할 수 있다. W6300 의 Ethernet Packet 통신 처리(IP 충돌감지, WOL Magic Packet 수신, 각 통신 SOCKET 별 데이터 송수신 등)를 위한 Event 가 발생할 경우 INTn 이 Low 로 Assert 된다.

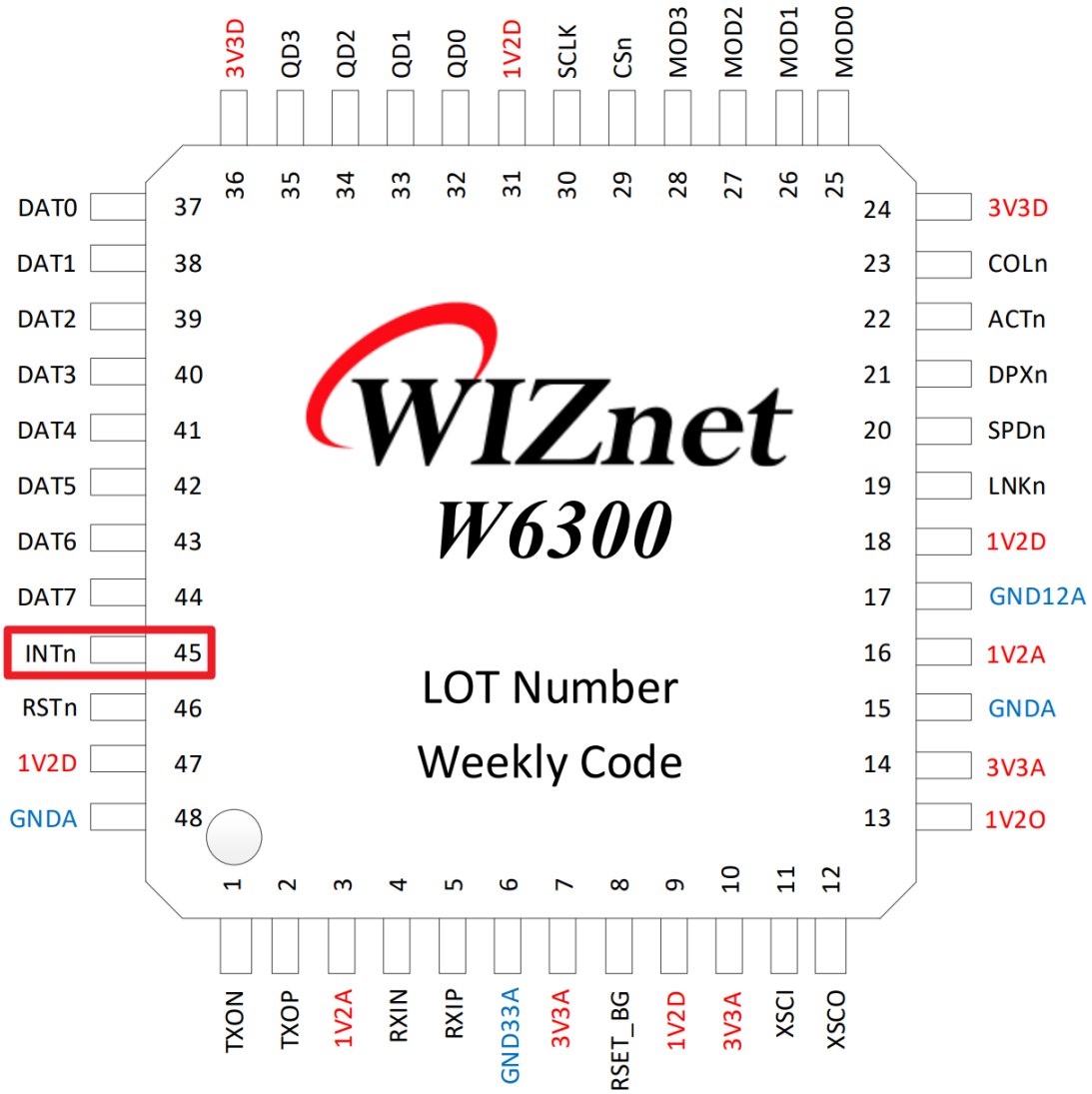


Figure 1 W6300 INTn Pin

INTn 의 기본 설정 값은 Enable 이며 SYCR1(System Config Register 1)의 IEN bit 를 통하여 Enable/ Disable 설정이 가능하다.

1. Interrupt configuration

1.1 Registers related to interrupt

Table 1 Interrupt 관련 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	0x2108	IR Clear Register
SIMR	0x2114	SOCKET Interrupt Mask Register
SLIMR	0x2124	SOCKET-less Interrupt Mask Register
SLIRCLR	0x2128	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_IRCLIR	0x0028	Sn_IR Clear Register

Table 1 은 인터럽트와 관련 Registers 를 나타낸 것이며 각 Register 들의 자세한 설명은 W6300 Datasheet 를 참조하라.

Interrupt Register 들은 크게 네 가지로 나누어 진다. 첫 번째, Interrupt Register 는 발생된 Event 를 확인할 수 있는 Register 이다. 두 번째, IR Clear Register 는 Event 발생 후 Interrupt Register 를 Clear 해주는 Register 이다. 세 번째, Interrupt Mask Register 는 Interrupt Register 에 1:1 대응되는 Masking Register 로, 해당 Bit 가 1 로 Assert 되어 있어야 해당 Event 가 발생했을 때 INTn 이 Low 로 Assert 된다. 마지막으로, INTn 자체의 동작을 결정하는 Register 가 있다.

SYCR1 (System Config Register 1)는 INTn 의 Enable/Disable 을 결정하는 IEN Bit 를 지니고 있다. SYCR1 의 IEN 이 Enable (SYCR1[IEN] = '1') 상태여야만 Event 가 발생하면 INTn 핀이 Low 로 Assert 된다.

1.2 INTn 의 동작

INTn 은 Event 발생 여부를 HOST 에게 전압 레벨의 변화로 알려준다. INTn 의 전압 레벨은 기본적으로 High 상태를 유지하고 Event 발생 시 Low 상태로 Assert 된다. 이 때, Interrupt Pending Time Value 가 0 이 아니면 INTPTMR(Interrupt Pending Time Register)에 설정된 시간 이후 Low 상태로 Assert 된다. Host 에서의 Event 처리가 끝나면 해당 Interrupt Register Bit 에 '1'을 설정하여 Interrupt 가 clear 할 수 있으며 INTn 의 상태는 High 로 변한다.

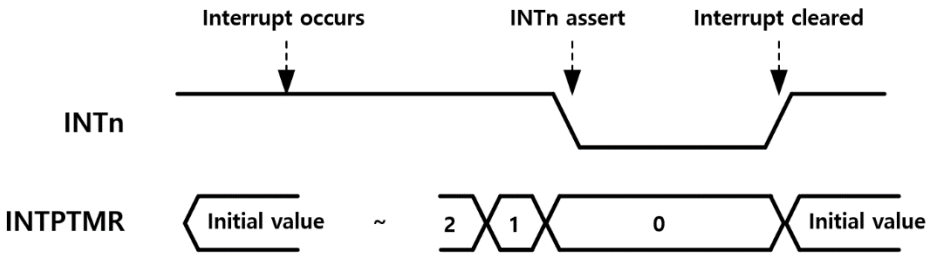


Figure 2 Interrupt and INTPTMR

1.3 Common interrupt

IR 은 MAGIC Packet 수신, IP Conflict, Port Unreachable, PADT/LCPT Receive 의 Event 를 감지하는 Interrupt 를 제공한다. IR 과 1:1 대응되는 IMR 의 Bit 를 '1'로 설정하여 해당 Interrupt 를 활성화한다. IMR 가 0 이면 Event 발생으로 인해 IR 의 각 비트가 1 로 변해도 INTn 은 Low 로 Assert 되지 않는다.

만약 IP Conflict Interrupt 를 확인하고자 한다면 다음과 같은 과정을 따라야 한다.

• 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

W6300 는 SOCKET 의 상태 변화 Event 를 감지하는 Interrupt 를 제공한다. Sn_IR 과 1:1 대응되는 Sn_IMR 의 Bit 를 '1'로 설정하여 해당 Interrupt 를 활성화시킬 수 있다. Sn_IMR 에 설정된 SOCKET Event 가 발생하면 Sn_IR 의 해당 Bit 가 '1'로 assert 되고 SIR 의 Sn_INT Bit 도 '1'로 assert 된다. 이 때, SYCR1 의 IEN Bit 나 SIMR 의 해당 Interrupt Mask Bit 가 비활성화 상태이면 INTn 이 Low 로 assert 되지 않는다. SOCKET Event 로 인해 INTn 이 Low 로 assert 되면 HOST 는 SIR 을 읽어 몇 번째 SOCKET 에서 Event 가 발생했는지 확인한 후 Sn_IR 을 통해 어떤 Interrupt 가 발생했는지 확인해야만 한다.
만약 0 번 소켓의 RECV Interrupt 를 확인하고자 한다면 다음과 같이 과정을 따라야 한다.

• Register Configuration

```
{
start:
    SYCR1 |= 1<<7;   //enable SYCR1[IEN] - enable INTn
    SIMR |= 1<<0;    // enable SIMR[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(SIR && 1<<0)   // SOCKET 0 Interrupt occurs?
    if(S0_IR && 1<<2) // RECV Interrupt occurs?
    //Do Something!
    S0_IRCLR |= 1<<2; //clear SOCKET 0 RECV Interrupt Bit
end
}
```

Socket 의 상태 변화에 따른 Event 는 SENDOK, TIMEOUT, RECV, DISCN, CON 이 있으며 자세한 사항은 W6300 Datasheet 를 참조하라.

1.5 Socket-less command interrupt

W6300 은 SOCKET Open 없이 ND(Neighbor Discovery), PING, ARP 등을 송수신할 수 있는 기능이 있다. 이에 따른 Interrupt 기능도 추가되었다. SLIR (SOCKET-less Interrupt Register)은 SLCR 에 대한 응답 Event 를 감지하는 Interrupt 를 제공한다. SLIR 과 1:1 대응되는 SLIMR 의 Bit 를 '1'로 설정하여 해당 Interrupt 를 활성화시킬 수 있다. SLIMR 에

설정된 SOCKET-less Command 의 응답 Event 가 발생하면 SLIR 의 해당 Bit 가 '1'로 assert 되고 INTn 이 Low 로 Assert 된다.

EX) SOCKET-less Command 를 이용해 PING Packet 을 전송하고 PING 응답에 대한 Interrupt 를 확인하고자 한다면 다음과 같은 과정을 따라야 한다.

• 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
}
```

SOCKET-less Command 응답 Event 는 RA, RS, NS, PING6, ARP6, PING4, ARP4, TOUT 이 있으며 자세한 사항은 W6300 Datasheet 를 참조하라.

2. HOST 에서의 Interrupt 처리

2.1 HOST 측에서의 Interrupt 설정

HOST 는 W6300 과 연결되어 제어하는 MCU(Micro controller unit)를 의미한다. MCU 는 일반적으로 Pin 의 상태 변화(전압 Level 변화, Edge 등)를 감지하여 그에 맞는 동작을 할 수 있는 Hardware Component 를 내장하고 있다. External Interrupt unit 혹은 Event Controller Driver 등 다양한 이름으로 사용되고 있으나 동작은 거의 비슷하므로 본 문서에서는 External Interrupt 라고 명칭 하도록 하겠다. External Interrupt 를 구성하기 위해서는 MCU 의 관련 Hardware 초기화와 Interrupt Handler 가 필요하다. Interrupt Handler 는 W6300 에서 Event 가 발생했을 때에 실행되는 Software 를 말한다.

2.2 Packet receive Interrupt Example

해당 Example 은 STM32F1XX 시리즈 MCU 기준이며 WIZnet Ethernet IC 공식 library 인 ioLibrary 를 사용하였다.
W6300 의 INTn 핀이 STM32F1XX 의 GPIOB 의 1 번 핀에 연결되어 있고 Socket 0 의 RECV Interrupt 만 활성화 되어 있을 경우.

• 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);
 }
```

```
}
```

• W6300 Interrupt Configuration

```
{
    SYCR1 |= 1<<7;    // SYCR1[7] IEN Bit - enable INTn
    SIMR |= 1<<0;    // SIMR[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

해당 Example 은 STM32F1XX 시리즈 MCU 기준이며 WIZnet Ethernet IC 의 공식 library 인 ioLibrary 를 사용하여 작성하였다.
W6300 의 INTn 핀이 STM32F1XX 의 GPIOB 의 1 번 핀에 연결되어 있고 SOCKET-less Command 의 PING 의 응답만 활성화되어 있는 경우.

External Interrupt 를 활성화하는 부분은 *2.2 Packet receive Interrupt Example* 과 동일하다.

• 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. 기타

3.1 Interrupt 사용시 주의점

W6300 의 Read/Write 동작은 하나의 통신 Frame 으로 구성된다. (Datasheet 의 External Interface 항목 참고) 이 Frame 은 Instruction → Address → Dummy → Data 순서로 전송되며, 이 순서가 깨지지 않아야 정상적인 접근이 가능하다. 특히 QSPI Quad 모드에서는 아래 그림과 같은 Frame 형식을 반드시 유지해야 한다.

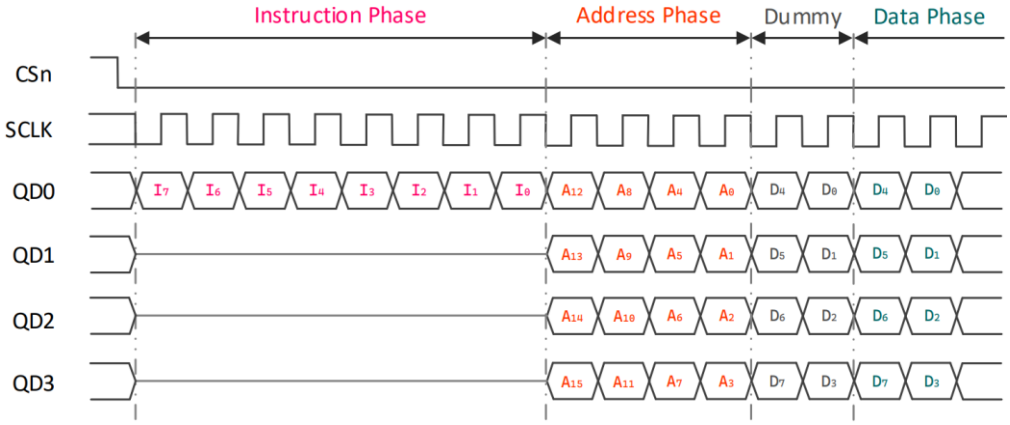


Figure 3 QSPI Quad Frame

Interrupt 를 사용하는 경우, 이 Frame 이 전송 도중 끊어질 수 있다는 점에 주의해야 한다. 예를 들어 W6300 에서 데이터를 읽기 위해 Frame 을 전송하던 중 Interrupt 가 발생하면, MCU 는 현재 동작을 중단하고 Interrupt handler 로 이동한다. 이때 handler 안에서 W6300 레지스터에 접근하거나, 같은 SPI/QSPI 버스를 공유하는 다른 IC 에 접근하면 진행 중이던 W6300 Frame 은 더 이상 유효하지 않게 된다. 이를 방지하려면 하나의 Frame 전송이 끝날 때까지는 같은 자원을 사용하는 Interrupt 가 실행되지 않도록 보호해야 한다. 즉, Frame 이 깨지지 않는 최소 단위를 기준으로, 해당 구간에서는 Interrupt 접근을 막는 설계가 필요하다.

반드시 순차적으로 수행되어야 할 Frame 을 일반적으로 Critical section 이라고 부르며 ioLibrary 에 포함된 reg_wizchip_cris_cbfunc() 함수를 통해 Critical section 이 수행될 때의 동작을 설정할 수 있다.

(참고 : https://github.com/Wiznet/ioLibrary_Driver/blob/f6406a7fd2e9e527f702320929dc751ecd5b707a/Ethernet/W6300/w6300.c#L63)

처음의 예와 같이 RECV Interrupt 를 활성화 한 상태에서 Send 를 수행한다고 가정하자. 이 때 Send 동작의 최소 Frame 이 시작되기 전에 Interrupt 는 Disable 되어 있어야만 한다.

그리고 Send 의 최소 Frame 이 끝나면 Interrupt 를 Enable 시켜 Frame 이 종료된 후 Interrupt_handler 가 동작할 수 있도록 프로그래밍해야 한다.

3.2 RTOS 에서 Interrupt 사용 시 주의점.

RTOS 도 Task 의 전환 때문에 *3.1 Interrupt 사용시 주의점* 에서 설명한 문제점과 동일한 현상이 발생할 수 있다. 마찬가지로 최소 Frame 이 동작하는 중간에 Task switching 이 일어나 다른 Task 를 수행할 때 같은 자원(Bus)을 사용하게 되면 Frame 이 손상되어 정상적인 데이터 통신을 할 수 없게 된다. 이를 막기 위해 reg_wizchip_cris_cbfunc()를 이용해 Scheduler 를 Lock/Unlock 하는 함수를 등록하거나 RTOS 에서 제공하는 Mutex(Mutual exclusive) 등의 기능을 활용하는 것이 바람직하다.

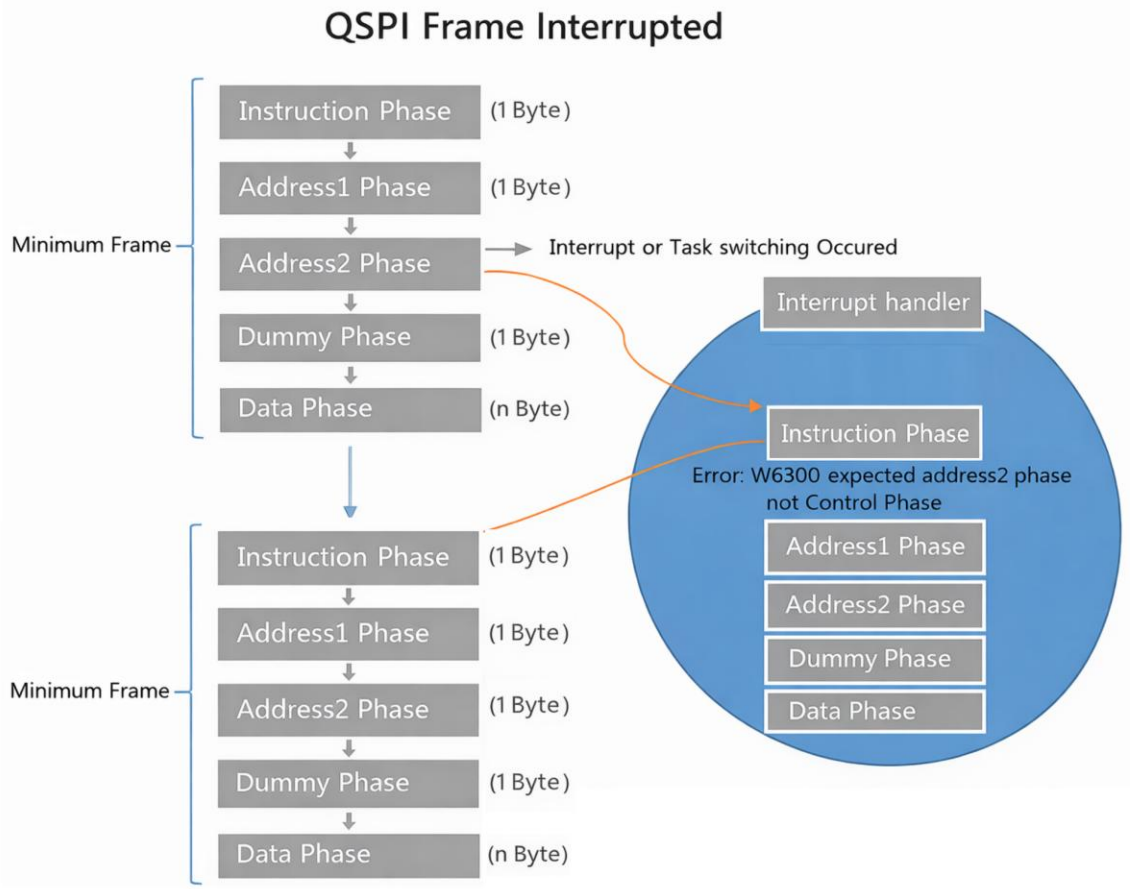


Figure 4 Behavior when an interrupt or task switching occurs

4. Document Revision History

Version	Date	Descriptions
Ver. 1.0.0	Dec, 2025	Initial Release

Technical support : https://maker.wiznet.io/forum
Sales & Distribution: sales@wiznet.io

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