Ethernet Protocol Stack Chip CH395

Datasheet
Version: 1E
http://wch.cn

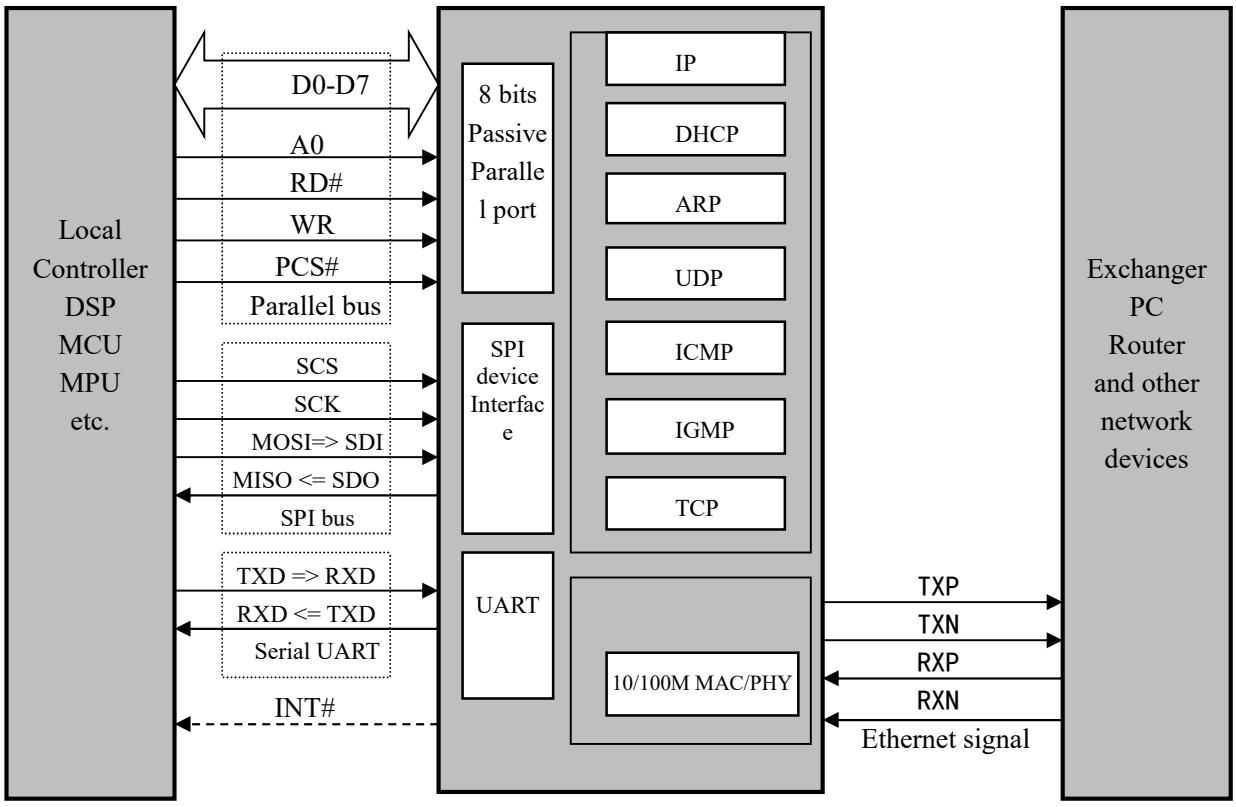
1. Overview

CH395 is an Ethernet protocol stack management chip, which is used for Ethernet communication in MCU system.

CH395 chip has 10/100M Ethernet Media Access Control (MAC) and Physical Layer (PHY), is fully compatible with IEEE802.3 10/100M protocol, and has built-in Ethernet stack firmware such as IP, DHCP, ARP, ICMP, IGMP, UDP and TCP. MCU system can perform network communication conveniently through CH395 chip.

CH395 supports three communication interfaces: 8-bit parallel port, SPI interface or asynchronous serial port. Controllers such as DSP/MCU/MPU can control CH395 chip for Ethernet communication through any of the above communication interfaces.

The figure below is an application block diagram of CH395.



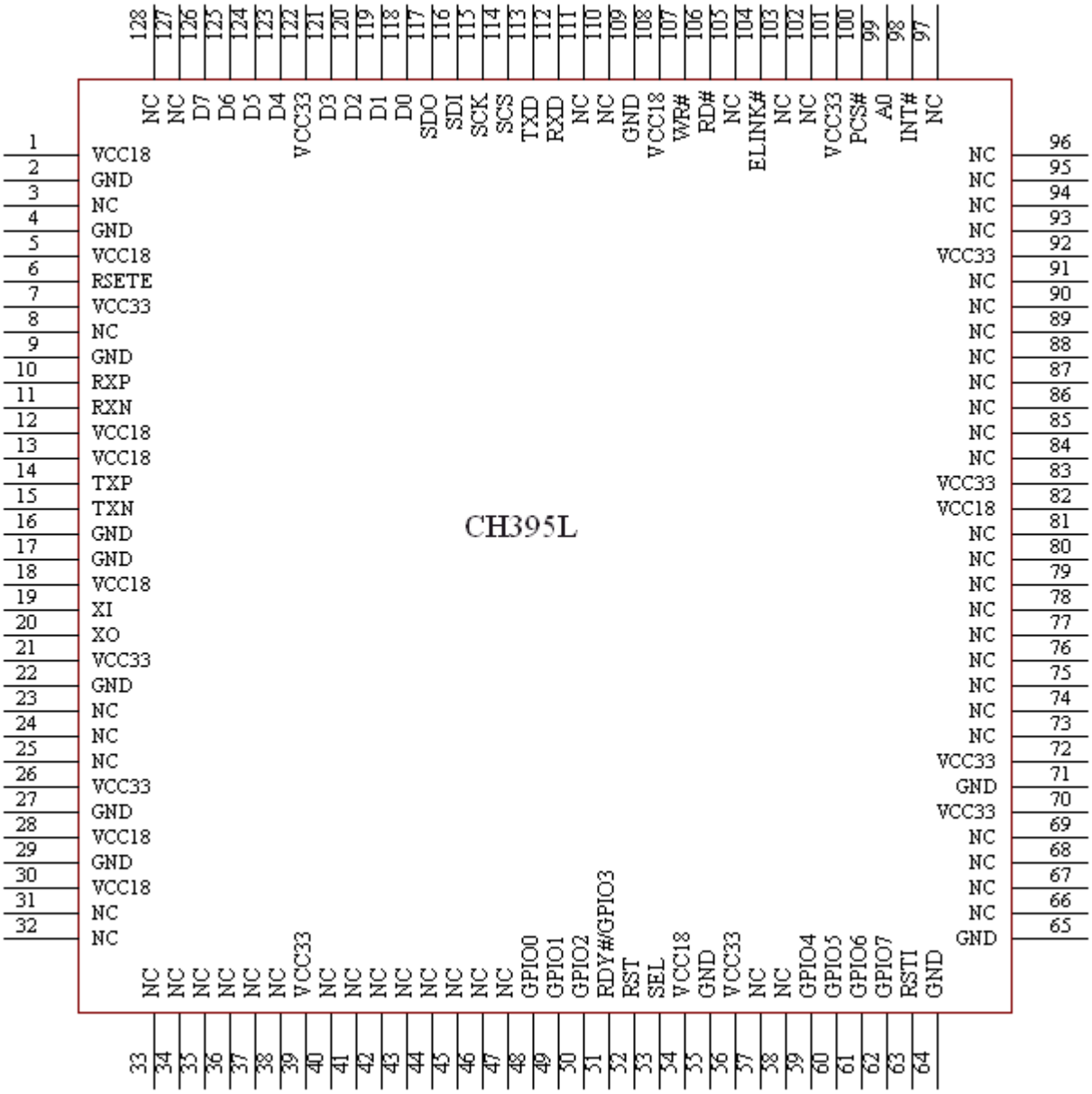
2. Features

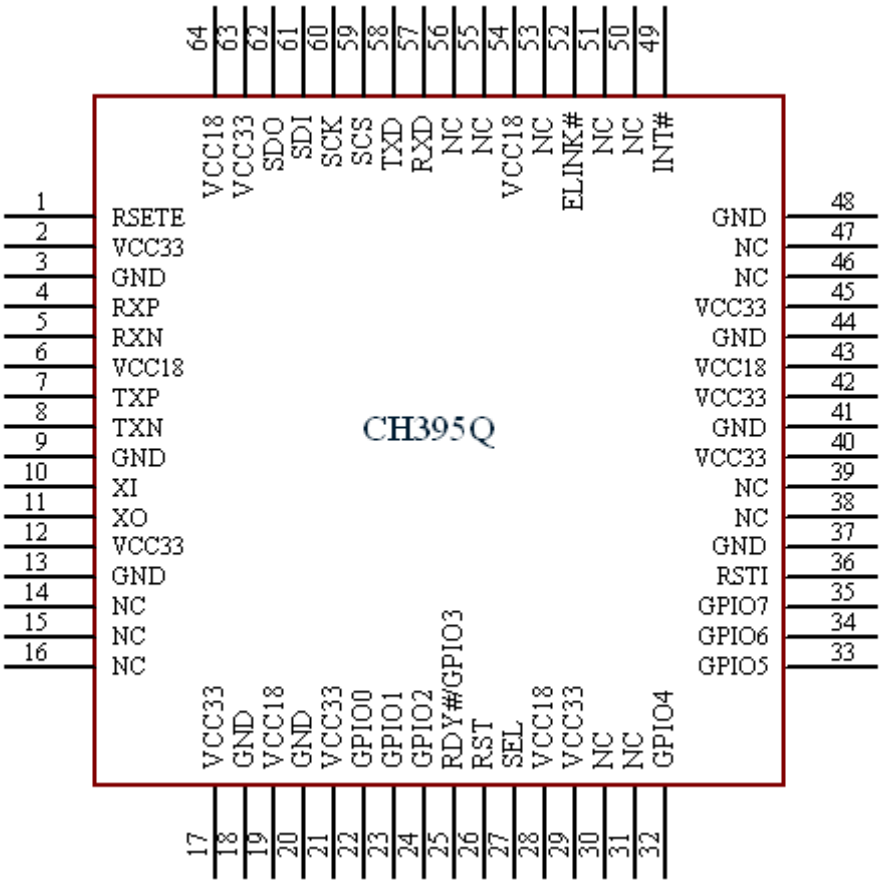
- Internal Ethernet MAC layer and PHY layer.
- Support 10/100M, full/half duplex self-adaption, and be compatible with 802.3 protocol.
- Support multiple modes of address filtering.
- Fully compatible with 802.3X full-duplex flow control and half-duplex back pressure flow control.
- Support automatic MDI/MDIX line conversion.
- Built-in TCP/IP protocol stack, and support IPv4, DHCP, ARP, ICMP, IGMP, UDP and TCP

protocols.

- Provide 8 independent Socket pairs, simultaneously transmit and receive data.
- Provide a high-speed 8-bit passive parallel interface and support the parallel data bus connected to MCU.
- Provide SPI device interface with the maximum speed of 30MHz, and support SPI serial bus connected to MCU.
- Provide UART with the maximum speed of 3Mbps, support the serial port connected to MCU, and support the dynamic adjustment of communication baud rate.
- Support low power mode.
- Built-in 24K RAM can be used for Ethernet data transmission and receiving, and each Socket transceiver buffer can be freely configured.
- Built-in 4KB EEPROM.
- Support 8-channel GPIO.
- Provide LQFP64M and LQFP128 lead-free package, and be compatible with RoHS.

3. Package





Chip model	Chip package	
	Name	Description
CH395L	LQFP128	LQFP package; 128 pins; package body 14x14mm
CH395Q	LQFP64M	LQFP package; 64 pins; package body 10 x10mm

4. Pins

CH395L Pin No.	CH395Q Pin No.	Pin Name	Pin Type	Description
7, 21, 26, 39, 56, 70, 72, 83, 92, 101, 122	2, 12, 17, 21, 29, 40, 42, 45, 63	VCC33	Power	3.3V positive power input terminal, 0.1uF power decoupling capacitor connected externally
1, 5, 12, 13, 18, 28, 30, 54, 82, 108	6, 19, 28, 43, 54, 64	VCC18	Power	1.8V positive power input terminal, 0.1uF power decoupling capacitor connected externally
2, 4, 9, 16, 17, 22, 27, 29, 55, 64, 65, 71, 109	3, 9, 13, 18, 20, 37, 41, 44, 48	GND	Power	Common ground

3, 8, 23, 24, 25, 31, 32, 33, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46, 47, 57, 58, 66, 67, 68, 69, 73, 74, 75, 76, 77, 78, 79, 80, 81, 84, 85, 86, 87, 88, 89, 90, 91, 93, 94, 95, 96, 97, 102, 103, 105, 110, 111, 127, 128	14, 15, 16, 30, 31, 38, 39, 46, 47, 50, 51, 53, 55, 56	NC	-	Reserved pins, suspended
6	1	RSETE	Input	Externally connect 12K-18K resistor to ground
10	4	RXP	Ethernet signal	Ethernet RXP signal
11	5	RXN	Ethernet signal	Ethernet RXN signal
14	7	TXP	Ethernet signal	Ethernet TXP signal
15	8	TXN	Ethernet signal	Ethernet TXN signal
19	10	XI	Input	Input terminal of the crystal oscillator, an external 30MHz crystal is required
20	11	XO	Output	Inverted output terminal of crystal oscillator, an external 30MHz crystal is required
48	22	GPIO0	Bidirectional Three-state	GPIO0, default input
49	23	GPIO1	Bidirectional Three-state	GPIO1, default input
50	24	GPIO2	Bidirectional Three-state	GPIO2, default input
51	25	GPIO3/RDY#	Bidirectional Three-state	GPIO3, default output, output low level after CH395 reset
52	26	RST	Output	Power-on rest output and external reset output, active high
53	27	SEL	Input	Interface configuration input during chip internal reset, built-in pull-up resistor
59	32	GPIO4	Bidirectional Three-state	GPIO4, default input
60	33	GPIO5	Bidirectional Three-state	GPIO5, default input
61	34	GPIO6	Bidirectional Three-state	GPIO6, default input
62	35	GPIO7	Bidirectional Three-state	GPIO7, default input

63	36	RSTI	Input	External reset input, active low, built-in pull-up resistor
98	49	INT#	Output	Interrupt request output, active low, built-in pull-up resistor
99	-	A0	Input	Parallel port address input, command port and data port distinguished, built-in pull-up resistor; when A0 is 1, write commands; when A0 is 0, read and write data
100	-	PCS#	Input	Chip selection control input of parallel port, active low, built-in pull-up resistor
106	-	RD#	Input	Read strobe input of parallel port, active low, built-in pull-up resistor
107	-	WR#	Input	Write strobe input of parallel port, active low, built-in pull-up resistor
112	57	RXD	Input	Serial data input of asynchronous serial interface, built-in pull-up resistor
113	58	TXD	Input Output	Interface configuration input during chip internal reset, built-in pull-up resistor, and serial data output of asynchronous serial port after chip reset
114	59	SCS	Input	Chip selection input of SPI interface, active low, built-in pull-up resistor
115	60	SCK	Input	Serial clock input of SPI interface, built-in pull-up resistor
116	61	SDI	Input	Serial data input of SPI interface, built-in pull-up resistor
117	62	SDO	Three-status output	Serial data output of SPI interface
118～121 123～126	-	D0～D7	Bidirectional Three-state	8-bit bidirectional data bus on parallel port, built-in pull-up resistor
104	52	ELINK	Output	Ethernet connection communication indicator lamp drive pin

(Note: The pins marked in gray in this table can withstand 3.3V input voltage, and the unmarked pins can only withstand 3.3V and 5V input.)

5. Commands

For the data in this datasheet, the suffix B is a binary number, and the suffix H is a hexadecimal number. Otherwise, it is a decimal number.

Double-word data (32 bits in total) with low bytes in the front (Little-Endian) refers to: the lowest byte (bits 7 to 0), followed by the lower byte (bits 15 to 8), followed by the higher byte (bits 23 to 16), followed by the highest byte (bits 31 to 24).

Data stream is a data block consisting of several consecutive bytes with a total length of at least 0 and at most 4096.

The number in the brackets for the input data and output data in the following table is the number of bytes of the parameter. If there are no brackets, it will be 1 byte by default.

MCU referred to in this datasheet is basically applicable to DSP or MCU/MPU/SCM, etc.

Socket Pair includes a quaternion of source IP, source port, destination IP and destination port, uniquely identifying both connection sides of Internet. It is Socket for short in this datasheet. CH395 provides 8 sockets at the same time, and their index values are 0, 1, 2, 3, 4, 5, 6 and 7 respectively.

The high and low bytes of IP and MAC addresses agreed in this datasheet may be different from some documents, for convenience only:

For example, the IP address is 192.168.1.2, where 192 is the lowest byte and 2 is the highest byte. Low bytes (IP) in this datasheet are in front.

For example, MAC address is 00.01.02.03.04.05, where 00 is the lowest byte and 05 is the highest byte. Low bytes (MAC) in this datasheet are in front.

For all commands including IP address input or output in this datasheet, IP low bytes are in front.
For all commands including MAC address input or output in this datasheet, MAC low bytes are in front.

Code	Command name CMD_	Input Data	Output data	Command purpose
01H	GET_IC_VER		Version	Get the chip and firmware versions
02H	SET_BAUDRATE	Baud rate coefficient (3)	(Wait for 1mS) Operation status	Set communication baud rate
03H	ENTER_SLEEP			Enter the low-power sleep suspended state
05H	RESET_ALL		(Wait for 50mS)	Execute hardware reset
19H	GET_GLOB_INT_STATUS_ALL		Global interrupt status (2)	Get the global interrupt status
06H	CHECK_EXIST	Any data	Bitwise NOT	Test the communication interface and operation status
20H	SET_PHY	PHY connection mode		Set PHY connection mode
21H	SET_MAC_ADDR	MAC address (6)		Set MAC address
22H	SET_IP_ADDR	IP address (4)		Set IP address
23H	SET_GWIP_ADDR	Gateway address (4)		Set gateway IP address
24H	SET_MASK_ADDR	Subnet mask (4)		Set subnet mask
25H	SET_MAC_FILT	Filter mode		Set MAC filter mode
		HASH0(4)		
		HASH1(4)		
26H	GET_PHY_STATUS		PHY status	Get PHY status
27H	INIT_CH395			CH395 chip initialization
28H	GET_UNREACH_IPPORT		Inaccessible information (8)	Get inaccessible IP, ports, and protocols
29H	GET_GLOB_INT_STATUS		Global interrupt status	Get the global interrupt status
2AH	SET_RETRAN_COUNT	Number of retries		Set the number of retries, 20 to the maximum

2BH	SET_RETRAN_PERIOD	Retry cycle (2)		Set the retry cycle, 1000MS to the maximum
2CH	GET_CMD_STATUS		Command execution status	Get command execution status
2DH	GET_REMOT_IPP_SN	Socket index	IP and port (6)	Get remote (destination) IP and port
2EH	CLEAR_RECV_BUF_SN	Socket index		Clear the receive buffer of Socket
2FH	GET_SOCKET_STATUS_SN	Socket index	Socket status	Get Socket status
30H	GET_INT_STATUS_SN	Socket index	Socket interrupt	Get Socket interrupt status
31H	SET_IP_ADDR_SN	Socket index Destination IP (4)		Set destination IP address of Socket
32H	SET_DES_PORT_SN	Socket index Destination port (2)		Set destination port of Socket
33H	SET_SOUR_PORT_SN	Socket index Source port (2)		Set source port of Socket
34H	SET_PROTO_TYPE_SN	Socket index Protocol type		Set working mode of Socket
35H	OPEN_SOCKET_SN	Socket index		Turn on Socket
36H	TCP_LISTEN_SN	Socket index		Start Socket monitoring
37H	TCP_CONNECT_SN	Socket index		Start Socket connection
38H	TCP_DISNCONNECT_SN	Socket index		Disconnct Socket TCP
39H	WRITE_SEND_BUF_SN	Socket index Length (2) Data stream (N)		Transmit buffer write data to Socket
3BH	GET_RECV_LEN_SN	Socket index	Length (2)	Gets the length of Socket received data
3CH	READ_RECV_BUF_SN	Socket index Length (2)	Data stream (N)	Receive data from Socket receive buffer
3DH	CLOSE_SOCKET_SN	Socket index		Close socket.
3EH	SET_IPRAW_PRO_SN	Socket index IP protocol field		Set the protocol field of Socket IP packet
3FH	PING_ENABLE	Enable flag		PING enable
40H	GET_MAC_ADDR		MAC address (6)	Get MAC address
41H	DHCP_ENABLE	Enable flag		Start (stop) DHCP
42H	GET_DHCP_STATUS		DHCP status	Get DHCP status
43H	GET_IP_INF		IP and other information	Get IP, MASK, DNS and other information

50H	SET_TCP_MSS	TCP MSS（2）		Set TCP MSS
51H	SET_TTL	TTL		Set TTL value, 128 to the maximum
52H	SET_RECV_BUF	Socket index Starting block address Number of blocks		Set Socket receive buffer
53H	SET_SEND_BUF	Socket index Starting block address Number of blocks		Set Socket transmit buffer
55H	SET_FUN_PARA	Functional parameters (4)		Set functional parameters
56H	SET_KEEP_LIVE_IDLE	4-byte time parameter		Set KEEPLIVE free time
57H	SET_KEEP_LIVE_INTVL	4-byte time parameter		Set KEEPLIVE timeout
58H	SET_KEEP_LIVE_CNT	Number of retries		Set the number of KEEPLIVE timeout retries
59H	SET_KEEP_LIVE_SN	Socket index Allocation		Set Socket KEEPLIVE
E9H	EEPROM_ERASE			Erase EEPROM
EAH	EEPROM_WRITE	EEPROM address (2) Length Data stream (N)		Write EEPROM
EBH	EEPROM_READ	EEPROM address (2) Length	Data stream (N)	Read EEPROM
ECH	READ_GPIO_REG	GPIP0 register address	GPI0 register value	Read GPIO Register
EDH	WRITE_GPIO_REG	GPIP0 register address GPIP0 register value		Write GPIO Register

For the command marked in gray of the table, it is necessary to execute a certain time and query the execution status of the command. MCU can get the status through GET_CMD_STATUS. (Refer to CH395INC.H for status definition)

5.1. CMD_GET_IC_VER

This command is used to get the chip and firmware versions. 1 byte of data returned is the version number, the bit 7 is 0, the bit 6 is 1, and the bits 5-0 are the version number. If the returned value is 41H, remove bits 7 and 6, and the version number will be 01H. It is called chip version in this text.

5.2. CMD_SET_BAUDRATE

This command is used to set the baud rate of CH395 for serial communication. When CH395 works in serial communication mode, the default communication baud rate is set by the level combination of SDO, SDI and SCK pins (refer to Section 6.4 of this datasheet) after reset. When these pins are suspended, the baud rate is 9600bps by default. If MCU supports high communication speed, the serial communication baud rate can be dynamically regulated through this command. This command requires the input of three data, namely, baud rate coefficient 0, baud rate coefficient 1 and baud rate coefficient 2. The following table shows the corresponding relationship with baud rates.

Baud rate coefficient 2	Baud rate coefficient 1	Baud rate coefficient 0	Communication baud rate (bps)	Error
00H	12H	C0H	4800	0%
00H	25H	80H	9600	0%
00H	4BH	00H	19200	0%
00H	96H	00H	38400	0%
00H	E1H	00H	57600	0%
01H	2CH	00H	76800	0.6%
01H	C2H	00H	115200	0.9%
07H	08H	00H	460800	0%
0EH	10H	00H	921600	7.1%
01H	86H	A0H	100000	0%
0FH	42H	40H	1000000	7.6%
2DH	C6H	C0H	3000000	0%
Calculation formula: BaudRate = (Baud rate coefficient 2 <<16) + (Baud rate coefficient 1 << 8) + Baud rate coefficient 0				

Usually, the communication baud rate is set within 1mS. After completion, CH395 outputs the operation state at the newly set communication baud rate. Therefore, MCU shall adjust its own communication baud rate in time after sending the command.

5.3. CMD_ENTER_SLEEP

This command enables CH395 chip in a low-power sleep suspended state. When MCU writes a new command to CH395 (no data input command, such as CMD_GET_IC_VER), it will exit the low-power state. For the parallel port and SPI interface communication modes, active SCS chip selection will also cause CH395 to exit the low-power state, so MCU shall immediately disable the SCS chip selection after sending the command CMD_ENTER_SLEEP.

In sleep state, MAC and PHY of CH395 will be in power off mode and disconnect Ethernet.

Typically, it takes several milliseconds for CH395 to exit the low-power state.

5.4. CMD_RESET_ALL

This command enables CH395 to perform a hardware reset. Typically, hardware reset is completed within 50mS.

5.5. CMD_CHECK_EXIST

This command is used to test the communication interface and working state to check whether CH395 is working properly. This command needs to input 1 byte of data, which can be any data. If CH395 is working properly, the output data of CH395 will be the bitwise reverse of the input data. For example, if the input data is 57H, the output data will be A8H.

5.6. CMD_SET_PHY

This command is used to set Ethernet PHY connection mode of CH395. The connection mode is automated negotiation mode by default. This command needs to input 1 byte of data, which is the connection mode code:

Disconnect PHY when the connection mode code is 01H;
PHY is 10M full duplex when the connection mode code is 02H;
PHY is 10M half duplex when the connection mode code is 04H;
PHY is 100M full duplex when the connection mode code is 08H;
PHY is 100M half duplex when the connection mode code is 10H;
PHY is automated negotiation when the connection mode code is 20H.

When CH395 receives this command, it will reset MAC and PHY and reconnect according to the newly set connection mode. If Ethernet is already connected, it will be disconnected and reconnected.

5.7. CMD_SET_MAC_ADDR

This command is used to set MAC address for CH395. It is necessary to input 6 bytes of MAC, with low bytes of MAC address in front. CH395 chip will store MAC address in the internal EEPROM. It will take 100mS to execute this command.

MAC address assigned by IEEE has been burned when CH395 chip is delivered. If it is not necessary, please do not set MAC address.

5.8. CMD_SET_IP_ADDR

This command is used to set IP address for CH395. It is necessary to input 4 bytes of IP address, with low bytes of IP in front. For all commands including IP input or output in this datasheet, IP low bytes are in front. This will not be explained below.

5.9. CMD_SET_GWIP_ADDR

This command is used to set the gateway address for CH395. It is necessary to input 4 bytes of IP address.

5.10. CMD_SET_MASK_ADDR

This command is used to set the subnet mask for CH395. It is necessary to input 4 bytes of mask for this command. It is 255.255.255.0 by default and may not be set.

5.11. CMD_SET_MAC_FILT

This command is used to set MAC filter mode. There are a variety of MAC filter modes set. This command requires the input of 9 bytes of data. The first byte is the filter mode, and the meanings of the data are as follows:

Bit	Name	Description
[5:7]	-	Reserved
4	SEND_ENABLE	Send enable
3	RECV_ENABLE	Receive enable
2	RECV_MULTIPKT	Receive multicast packets
1	RECV_ALL	Receive all data
0	RECV_BROADPKT	Receive broadcast packets

1 means ON, and 0 means OFF. RECV_BROADPKT, RECV_ENABLE and SEND_ENABLE are enabled by default after CH395 is reset.

The following table shows the meanings of bits:

RECV_ENABLE	RECV_ALL	RECV_BROADPKT	RECV_MULTIPKT	Description
1	0	0	0	Receive data packets matched with MAC address
1	1	X	X	Receive all data packets
1	0	1	0	Receive data packets matched with MAC address Receive broadcast data packets
1	0	0	1	Receive data packets matched with MAC address Receive multicast data packets
1	0	1	1	Receive data packets matched with MAC address Receive broadcast data packets Receive multicast data packets
0	X	X	X	Disable receiving

Bytes 2-5 are HASH0 (HASH table 0), bytes 6-9 are HASH1 (HASH table 1), and HASH0 and HASH1 are only valid when multicast is enabled.

HASH0 and HASH1 together form a 64-bit HASH table, bits 0-31 are HASH0 and bits 32-63 are HASH1.

HASH table calculation method: Use standard Ethernet redundancy check (CRC32) to calculate a 32-bit CRC value for the multicast address, use the high 6 bits of the CRC value as the index value, and write the corresponding bit of the HASH table as 1. For example, the bit 0 of HASH1 shall be written as 1 when the calculated high 6 bits of the CRC for the multicast address are 32.

5.12. CMD_GET_PHY_STATUS

This command is used to get PHY connection status. After receiving this command, CH395 will query the current PHY connection status and output 1-byte PHY connection status code:

PHY is disconnected when the connection status code is 01H;
PHY connection is 10M full duplex when the connection status code is 02H;
PHY connection is 10M half duplex when the connection status code is 04H.
PHY connection is 100M full duplex when the connection status code is 08H;
PHY connection is 100M half duplex when the connection status code is 10H.

5.13. CMD_INIT_CH395

This command is used to initialize CH395, including initializing MAC, PHY and TCP/IP stack of CH395. Generally, it takes 350mS to execute the command. MCU can send GET_CMD_STATUS to query whether the execution has finished and the execution status.

5.14. CMD_GET_UNREACH_IPPORT

This command is used to get an inaccessible IP, ports and protocol type. CH395 will generate an inaccessible interrupt when an inaccessible message is received. MCU can use this command to get inaccessible information. After receiving this command, CH395 will output 1 byte of inaccessible code, 1 byte of protocol type, 2 bytes of port number (low bytes in front), and 4 bytes of IP in turn. MCU can judge whether the protocol, port or IP is inaccessible according to the inaccessible codes. For inaccessible codes, refer to RFC792 (CH395INC.H defines four common inaccessible codes).

5.15. CMD_GET_GLOB_INT_STATUS

This command is used to get the global interrupt status. CH395 will output 1 byte of global interrupt status after receiving this command. Global interrupt status is defined as follows:

Bit	Name	Description
7	GINT_STAT_SOCK3	Socket3 interrupt
6	GINT_STAT_SOCK2	Socket2 interrupt
5	GINT_STAT_SOCK1	Socket1 interrupt
4	GINT_STAT_SOCK0	Socket0 interrupt
3	GINT_STAT_DHCP	DHCP interrupt
2	GINT_STAT_PHY_CHANGE	PHY status change interrupt
1	GINT_STAT_IP_CONFLI	IP conflict
0	GINT_STAT_UNREACH	Inaccessible interrupt

① GINT_STAT_UNREACH: Inaccessible interrupt. When CH395 receives ICMP inaccessible interrupt message, it saves the IP address, port and protocol type of the inaccessible IP packet in the inaccessible information table, and then generates an interrupt. When the MCU receives the interrupt, it can send the command GET_UNREACH_IPPORT to get the inaccessible information.

② GINT_STAT_IP_CONFLI: IP conflict interrupt. This interrupt is generated when CH395 detects that its IP address is the same as that of other network devices in the same network segment.

③ GINT_STAT_PHY_CHANGE: PHY change interrupt. This interrupt is generated when PHY connection of CH395 changes, for example, PHY state changes from the connected state to the disconnected state or from the disconnected state to the connected state. MCU can send GET_PHY_STATUS command to get the current PHY connection status.

④ GINT_STAT_DHCP: DHCP interrupt. If MCU enables DHCP function of CH395, CH395 will generate this interrupt. MCU can send the command CMD_GET_DHCP_STATUS to get the DHCP status. If the status is 0, it will indicate success; otherwise, it will indicate timeout failure.

⑤ GINT_STAT_SOCK0 - GINT_STAT_SOCK3: Socket interrupt. When there is an interrupt event in Socket, CH395 will generate this interrupt. MCU needs to send GET_INT_STATUS_SN to get the interrupt status of Socket. Please refer to GET_INT_STATUS_SN.

When this command is completed, CH395 will set INT# pin to high level and clear the global interrupt status.

5.16. CMD_GET_GLOB_INT_STATUS_ALL

This command is used to get the global interrupt status. CH395 will output 2 bytes of global interrupt status after receiving this command. Global interrupt status is defined as follows:

Bit	Name	Description
[12：18]	-	Reserved
11	GINT_STAT_SOCK7	Socket7 interrupt
10	GINT_STAT_SOCK6	Socket6 interrupt

9	GINT_STAT_SOCK5	Socket5 interrupt
8	GINT_STAT_SOCK4	Socket4 interrupt
7	GINT_STAT_SOCK3	Socket3 interrupt
6	GINT_STAT_SOCK2	Socket2 interrupt
5	GINT_STAT_SOCK1	Socket1 interrupt
4	GINT_STAT_SOCK0	Socket0 interrupt
3	GINT_STAT_DHCP	DHCP interrupt
2	GINT_STAT_PHY_CHANGE	PHY status change interrupt
1	GINT_STAT_IP_CONFLI	IP conflict
0	GINT_STAT_UNREACH	Inaccessible interrupt

For bits 0-7, please refer to Section 5.15.

GINT_STAT_SOCK4 - GINT_STAT_SOCK7: Socket interrupt. When there is an interrupt event in Socket, CH395 will generate this interrupt. MCU needs to send GET_INT_STATUS_SN to get the interrupt status of Socket. Please refer to GET_INT_STATUS_SN.

CH395 can get the interrupt status through two commands CMD_GET_GLOB_INT_STATUS and CMD_GET_GLOB_INT_STATUS_ALL. The low 8-bit interrupt status can be only gotten through the former command, and the full interrupt status can be gotten through the latter command. It shall be noted that any version of chip supports the command CMD_GET_GLOB_INT_STATUS. If the chip version number is more than or equal to 0X44 and Socket4 -- Socke7 is used, only CMD_GET_GLOB_INT_STATUS_ALL can be used. If the chip version number is less than 0X44, the command CMD_GET_GLOB_INT_STATUS_ALL will not be supported.

5.17. CMD_SET_RETRAN_COUNT

This command is used to set the number of retries. It is necessary to input 1 byte of number of retries. The allowable maximum value is 20. If the input data is more than 20, it will be processed as 20. The default number of retries is 12, and retries are only valid in TCP mode.

5.18. CMD_SET_RETRAN_PERIOD

This command is used to set the retry cycle. It is necessary to input 2 bytes of number of cycles of (with low bytes in front) in milliseconds. The allowable maximum value is 1000. The total retry time is N * M, N is the number of retries, and M is the retry cycle. The default retry cycle is 500MS and retries are only valid in TCP mode.

5.19. CMD_GET_CMD_STATUS

This command is used to get the command execution status. CH395 will output 1 byte of data, which is the command execution state. The command execution status is as follows:

Code	Name	Description
00H	CH395_ERR_SUCCESS	Success
10H	CH395_ERR_BUSY	Busy, the command is being executed
11H	CH395_ERR_MEM	Memory Management error
12H	CH395_ERR_BUF	Buffer error
13H	CH395_ERR_TIMEOUT	Timeout
14H	CH395_ERR_RTE	Route error
15H	CH395_ERR_ABRT	Connection suspended

16H	CH395_ERR_RST	Connection reset
17H	CH395_ERR_CLSD	Connection closed
18H	CH395_ERR_CONN	No connection
19H	CH395_ERR_VAL	Value error
1AH	CH395_ERR_ARG	Parameter error
1BH	CH395_ERR_USE	Used
1CH	CH395_ERR_IF	MAC error
1DH	CH395_ERR_ISCONN	Connected
20H	CH395_ERR_OPEN	Opened

If MCU receives CH395_ERR_BUSY, indicating that CH395 is executing the command, MCU shall delay more than 2 milliseconds and send the command CMD_GET_CMD_STATUS again to get the status.

For the commands marked in gray of the command code table, it is necessary to send CMD_GET_CMD_STATUS to get the execution status.

5.20. CMD_GET_REMOT_IPP_SN

This command is used to get the remote IP address and port number. It is necessary to input 1 byte of Socket index value. CH395 will output 4 bytes of IP address and 2 bytes of port number (low bytes in front). After Socket works in TCP Server mode and the connection is established, MCU can get the remote IP address and port number through this command.

5.21. CMD_CLEAR_RECV_BUF_SN

This command is used to clear the Socket receive buffer. It is necessary to input 1 byte of Socket index value. Upon receiving this command, CH395 will reset the receiving length of this Socket, and the receiving pointer will point to the buffer head.

5.22. CMD_GET_SOCKET_STATUS_SN

This command is used to get Socket status. It is necessary to input a 1 byte of Socket index value. CH395 will output a 2-byte status code when receiving this command.

The first status code is the status code of Socket. The status code of Socket is defined as follows:

Code	Name
00H	SOCKET_CLOSED
05H	SOCKET_OPEN

The second status code is TCP status code, which is only meaningful when TCP mode has been on. TCP status code is defined as follows:

Code	Name	Description
00H	TCP_CLOSED	Closed
01H	TCP_LISTEN	Monitoring
02H	TCP_SYN_SENT	SYN sent
03H	TCP_SYN_RCVD	SYN received
04H	TCP_ESTABLISHED	TCP connection established
05H	TCP_FIN_WAIT_1	The active closing side first sends FIN
06H	TCP_FIN_WAIT_2	The active closing side receives an ACK from FIN
07H	TCP_CLOSE_WAIT	The passive closing side receives FIN

08H	TCP_CLOSING	Closing
09H	TCP_LAST_ACK	The passive closing side sends FIN
0AH	TCP_TIME_WAIT	2MLS waiting status

TCP status is defined in TCP/IP protocol. Please refer to TCP/IP protocol for the detailed meaning.

5.23. CMD_GET_INT_STATUS_SN

This command is used to get the interrupt status of Socket. It is necessary to input 1 byte of Socket index value. After receiving this command, CH395 will output 1 byte of Socket interrupt code. The interrupt code bits are defined as follows:

Bit	Name	Description
7	-	Reserved
6	SINT_STAT_TIM_OUT	Timeout
5	-	Reserved
4	SINT_STAT_DISCONNECT	TCP disconnected
3	SINT_STAT_CONNECT	TCP connected
2	SINT_STAT_RECV	Receive buffer not empty
1	SINT_STAT_SEND_OK	Send Success
0	SINT_STAT_SENBUF_FREE	Transmit buffer free

① SINT_STAT_SENBUF_FREE, transmit buffer free interrupt. After MCU writes data to the transmit buffer of Socket, CH395 will quickly copy the data to the internal protocol stack or MAC buffer, in order to encapsulate the data. When the data copying is finished, this interrupt will be generated. MCU can continue to write the subsequent data to the transmit buffer. After MCU writes data to the transmit buffer of Socket once, it must write the next data until the interruption is generated.

② SINT_STAT_SEND_OK, send OK interrupt. This interrupt indicates that the data packet is sent successfully. This interrupt will be generated after Sokcet sends a packet of data successfully. After MCU writes data to Socket buffer once, CH395 may encapsulate the data into several data packets for sending, so several send OK interrupts may be generated.

③ SINT_STAT_CONNECT, TCP linkage interrupt, active only in TCP mode. It indicates that TCP connection is successful, and MCU can transmit data only after the interrupt is generated.

④ SINT_STAT_DISCONNECT, TCP disconnection interrupt, only active in TCP mode, indicating TCP disconnection.

⑤ SINT_STAT_TIM_OUT. This interrupt will be generated in TCP mode when a timeout occurs during a TCP connection, disconnection, data sending and other processes. This interrupt will be generated if the data is not sent successfully in IPRAW or UDP mode.

After the interrupts SINT_STAT_DISCONNECT and SINT_STAT_TIM_OUT are generated, CH395 takes different actions depending on whether the FUN_PARA_FLAG_SOCKET_CLOSE bit is 1 or 0. If FUN_PARA_FLAG_SOCKET_CLOSE is 0, CH395 will actively set the Socket state to the OFF state and clear all relevant buffers after the above two interrupts are generated. Otherwise, it will not do any operation on the Socket status and the relevant buffer, so as to facilitate the external MCU to read the residual data after TCP is disconnected or timed out. When the external MCU reads the data, it must send a close command to close the Socket.

5.24. CMD_SET_IP_ADDR_SN

This command is used to set the destination IP address of Socket. It is necessary to input 1 byte of Socket index value and 4 bytes of destination IP address. When Socket works in IPRAW, UDP, or TCP Client mode, the destination IP must be set before the command CMD_OPEN_SOCKET_SN is sent.

5.25. CMD_SET_DES_PORT_SN

This command is used to set the Socket destination port. It is necessary to input 1 byte of Socket index value and 2 bytes of destination port (the low bytes are in front). When Socket works in UDP or TCP Client mode, this value must be set.

5.26. CMD_SET_SOUR_PORT_SN

This command is used to set the source port of Socket. It is necessary to input 1 byte of Socket index value and 2 bytes of source port (low bytes in front). If two or more Sockets are in the same mode, the source port numbers must not be the same. For example, Socket 0 is in UDP mode, the source port number is 600, and Socket 1 is also in UDP mode. The source port number 600 cannot be used again, otherwise it may cause the opening failure.

5.27. CMD_SET_PROTO_TYPE_SN

This command is used to set the working mode of Socket. It is necessary to input 1 byte of Socket index value and 1 byte of working mode. The working mode is defined as follows:

Code	Name	Description
03H	PROTO_TYPE_TCP	TCP mode
02H	PROTO_TYPE_UDP	UDP mode
01H	PROTO_TYPE_MAC_RAW	MAC original message mode
00H	PROTO_TYPE_IP_RAW	IP original message mode

This command must be executed before CMD_OPEN_SOCKET_SN. Refer to 8.3 Application Reference Steps for detailed steps.

5.28. CMD_OPEN_SOCKET_SN

This command is used to open Socket and use the necessary steps of Socket. It is necessary to input 1 byte of Socket index value. After sending this command, MCU shall send GET_CMD_STATUS to query the command execution status. After opening Socket in UDP, IPRAW or MACRAW mode and returning successfully, data transmission can be performed. Before this command is sent, necessary settings must be made for destination IP, protocol type, source port, destination port, etc. Please refer to 8.3 Application Reference Steps for detailed steps.

5.29. CMD_TCP_LISTEN_SN

This command is only valid in TCP mode, enabling the Socket to be in the monitoring mode, namely, TCP Server mode. It is necessary to input a 1 byte of Socket index value. This command must be executed after OPEN_SOCKET_SN. After sending this command, MCU shall send GET_CMD_STATUS to query the command execution status.

In TCP Server mode, the Socket will always detect connection events, and the interrupt SINT_STAT_CONNECT will be generated until the connection is successful. Only one connection can be established for each Socket. If an eligible connection event is received again, Socket will send TCP RESET to the remote end tried to be connected.

5.30. CMD_TCP_CONNECT_SN

This command is only valid in TCP mode, enabling the Socket to be in the connection mode, namely, TCP Client mode. It is necessary to input 1 byte of Socket index value. After sending this command, MCU shall send GET_CMD_STATUS to query the command execution status.

After this command is received, the Socket will initiate the connection event, and the interrupt SINT_STAT_CONNECT will be generated after the successful connection. The interrupt SINT_STAT_TIM_OUT will be generated if an exception occurs during the connection or the connection fails after a certain amount of time. MCU receives this interrupt. If it needs to connect again, it will be necessary to reopen the Socket and execute TCP_CONNECT_SN.

5.31. CMD_TCP_DISNCONNECT_SN

This command is only valid in TCP mode to disconnect the current TCP. It is necessary to input 1 byte of Socket index value. After sending this command, MCU shall send GET_CMD_STATUS to query the command execution status. SINT_STAT_DISCONNECT will be generated when the current TCP is successfully disconnected.

5.32. CMD_WRITE_SEND_BUF_SN

This command is used to write data to Socket transmit buffer. It is necessary to input 1 byte of Socket index value, 2 bytes of length (low bytes in front) and several bytes of data stream. The length of input data must not be larger than the size of transmit buffer. However, in MACRAW mode, the maximum length of input data can only be 1514, and any redundant data will be discarded. After the external MCU writes the data, CH395 will encapsulate the data packet according to the working mode of Socket, and then send it. Before MCU receives SINT_STAT_SENBUF_FREE, it is not allowed to write data into Socket transmit buffer again.

5.33. CMD_GET_RECV_LEN_SN

This command is used to get the valid data length of the current receive buffer. It is necessary to input 1 byte of Socket index value. CH395 outputs 2 bytes after receiving this command (low bytes in front).

5.34. CMD_READ_RECV_BUF_SN

This command is used to read data from Socket receive buffer. It is necessary to input 1 byte of Socket index value and 2 bytes of length (low bytes in front). CH395 will output several bytes of data stream based on the length value. In actual application, the command RECV_LEN_SN can be firstly sent to get the actual effective length of the current buffer. The length of the read data can be less than the actual effective length of the buffer, the unread data is still reserved in the receive buffer, and MCU can continue to read through this command.

In MACRAW mode, the processing modes are different. In MACRAW mode, the receive buffer is a frame buffer, which can cache only 1 frame of Ethernet data. After CH395 processes the command READ_RECV_BUF_SN, Socket0 receive buffer will be cleared, so MCU shall read all valid data of the buffer at a time.

5.35. CMD_CLOSE_SOCKET_SN

This command is used to close Socket. It is necessary to input a 1 byte of Socket index value. After Socket is closed, the receive buffer and transmit buffer of Socket are emptied, but the configuration information is still reserved, and you just need to open the Socket again when using the Socket the next time.

In TCP mode, CH395 will automatically disconnect TCP before turning off Socket.

5.36. CMD_SET_IPRAW_PRO_SN

This command is only valid in IPRAW mode. It is necessary to input 1 byte of Socket index value and 1 byte of protocol code in IP packet protocol field. This command must be executed before OPEN_SOCKET_SN.

If multiple sockets adopt IPRAW mode, the protocol code cannot be used repeatedly. For example, if both Socket 0 and Socket 1 adopt IPRAW mode, the protocol codes of the two sockets must not be the same,

otherwise it may cause the failure to turn on Socket.

IPRAW has a higher data processing priority than UDP and TCP, so the protocol code must not be the same as other sockets. For example, Socket 0 adopts IPRAW mode, and the protocol code is 17 (UDP protocol). Socket 1 adopts UDP mode. This may cause Socket 1 data to be received by Socket 0.

5.37. CMD_PING_ENABLE

This command is used to turn PING on or off. It is necessary to input a 1-byte flag. If the flag is 1, it will indicate that PING is on; if the flag is 0, it will indicate that PING is off.

5.38. CMD_GET_MAC_ADDR

This command is used to get MAC address. When receiving this command, CH395 will output 6 bytes of MAC address.

5.39. CMD_DHCP_ENABLE

This command is used to start or stop DHCP. It is necessary to input a 1-byte flag. If the flag is 1, it will indicate that DHCP is on; if the flag is 0, it will indicate that DHCP is off. CH395 must be initialized before DHCP is started.

After DHCP is started, CH395 will broadcast DHCPDISCOVER message to the network to discover DHCP Server, request the address and other configuration parameters after finding DHCP Server, and then generate GINT_STAT_DHCP interrupt. MCU can send GET_DHCP_STATUS command to get DHCP status. If the status code is 0, it will indicate success, and MCU can send the command GET_IP_INF to get IP, MASK and other information. If the status code is 1, it will indicate error, which is generally caused by timeout, for example, no DHCP Server is found.

DHCP is always in a working state after startup unless it receives a DHCP shutdown command from MCU. During this process, if DHCP Server reassigns a configuration to CH395 and the configuration is different from the original configuration, CH395 will still generate an interrupt.

After timeout interrupt is generated, if DHCP Server is not found, CH395 will continue to send DHCPDISCOVER message at an interval of about 16 seconds.

It takes about 20MS to execute this command. MCU can send GET_CMD_STATUS to query whether the execution has finished and the execution status

5.40. CMD_GET_DHCP_STATUS

This command is used to get the status of DHCP. Generally, after MCU receives the interrupt, this command will be generated to get the execution status of DHCP. After receiving this command, CH395 outputs DHCP status codes. There are two status codes of 0 and 1, which have the following meanings:

If the status code is 0, it will indicate success, and MCU can send the command GET_IP_INF to get IP, MASK and other information.

If the status code is 1, it will indicate error, which is generally caused by timeout, for example, no DHCP Server is found.

5.41. CMD_GET_IP_INF

This command is used to get IP, GatewayIP, MASK, DNS, etc. After receiving this command, CH395 will output 20 bytes of data in turn, namely, 4 bytes of IP address, 4 bytes of GatewayIP, 4 bytes of subnet MASK, 4 bytes of DNS1 (primary DNS), and 4 bytes of DNS2 (secondary DNS).

After DHCP, this command can be sent to get the current CH395 information. If some configurations are not obtained in DHCP, this configuration will be 0. For example, DNS is not always allocated in the local

network DHCP, and both DNS1 and DNS2 are 0 when this command is sent to get the configuration information.

5.42. CMD_SET_TCP_MSS

This command is used to set TCP MSS. It is necessary to input 2 bytes of TCP MSS value with a maximum of 1460 and a minimum of 60. This value shall be set before CH395 is initialized, and it is not allowed to be set after initialization.

5.43. CMD_SET_TTL

This command is used to set Socket TTL. It is necessary to input 1 byte of Socket index value and 1 byte of TTL value. It shall be set after the Socket is opened, and the maximum value is 128.

5.44. CMD_SET_RECV_BUF

This command is used to set the receive buffer of Socket. It is necessary to input 3 bytes of data, the first byte is the Socket index, the second byte is the starting block of the buffer, and the third byte is the number of blocks.

Block 0	Block 1	Block 2	……	Block 46	Block 47

The internal buffer structure of CH395, as shown above, consists of 48 blocks. The length of each block is 512 bytes. MCU can freely allocate the size of each Socket receive buffer. After the initialization of CH395, the buffer allocation is as follows:

Socket	Buffer	Start block	Number of blocks
0	Receive buffer	0	8
	Transmit buffer	8	4
1	Receive buffer	12	8
	Transmit buffer	20	4
2	Receive buffer	24	8
	Transmit buffer	32	4
3	Receive buffer	36	8
	Transmit buffer	44	4
4	Receive buffer	NULL (empty, unallocated, same below)	0
	Transmit buffer	NULL	0
5	Receive buffer	NULL	0
	Transmit buffer	NULL	0
6	Receive buffer	NULL	0
	Transmit buffer	NULL	0
7	Receive buffer	NULL	0
	Transmit buffer	NULL	0

It can be seen from the above table that after CH395 is initialized, all buffers are allocated to Sockets 0-3. There are 8 (4KB) receive buffers and 4 (2KB) transmit buffers. If the number of Sockets needed by MCU is more than 4, the buffers shall be reallocated.

5.45. CMD_SET_SEND_BUF

This command is used to set the transmit buffer of Socket. It is necessary to input 3 bytes of data, the first byte is the Socket index, the second byte is the starting block of the buffer, and the third byte is the number

of blocks.

Refer to Section 5.47 for the definition and allocation of buffers.

5.46. CMD_EEPROM_ERASE

This command is used to erase EEPROM. CH395 chip has a 4KB EEPROM. The data is 0XFF after erasing. Before write operation to EEPROM, all data in the destination area must be 0XFF.

5.47. CMD_EEPROM_WRITE

This command is used to write EEPROM. It is necessary to input 2 bytes of address, 1 byte of length, and several bytes of data stream. The length of byte stream must be no more than 64 bytes.

5.48. CMD_EEPROM_READ

This command is used to read EEPROM. It is necessary to input 2 bytes of address and 1 byte of length. CH395 will output several bytes of data stream based on the length. The length value must be no more than 64 bytes. The external MCU shall wait for 1MS to read the data after sending the length.

5.49. CMD_READ_GPIO_REG

This command is used to read GPIO register. It is necessary to input a 1 byte of register address. CH395 outputs a 1 byte of register value. All registers are 8-bit. Bits 0-7 correspond to GPIO 0-7 respectively. Its addresses and meanings are shown in the following table:

Address:	Name	Description
80H	GPIO_DIR_REG	Direction register, 1: output; 0: input
81H	GPIO_IN_REG	Input data register
82H	GPIO_OUT_REG	Output data register
83H	GPIO_CLR_REG	0: hold ; 1: clear
84H	GPIO_PU_REG	Pull-down register, 1: pull-up enable; 0: pull-up disable
85H	GPIO_PD_REG	Pull-up register, 1: pull-down enable; 0: pull-down disable

CH395 has 8 GPIOs. GPIO3 is output by default. After successful initialization of CH395, CH395 outputs low level, and the rest of GPIOs are inputs by default.

5.50. CMD_WRITE_GPIO_REG

This command is used to write GPIO register. It is necessary to input a 1 byte of register address and a 1 byte of register value.

5.51. CMD_SET_FUN_PARA

This command is used to set the functional parameters. It is necessary to input 4 bytes of parameters. The meanings of the parameters are as follows:

Bit	Name	Reset value	Description
0	-	-	Reserved, it must be 0
1	FUN_PARA_FLAG_TCP_SERVER	0	TCP server multi-connection mode enable bit, 0x44 and later version support
2	FUN_PARA_FLAG_LOW_PWR	0	Low power mode enable bit, 0x44 and later version support
3	FUN_PARA_FLAG_SOCKET_CLOSE	0	Socket OFF mode, 0x46 and later version support

4	FUN_PARA_FLAG_DISABLE_SEND_OK	0	Disable Socket SEND_OK interrupt, 0x46 and later version support
5:31			Reserved

FUN_PARA_FLAG_TCP_SERVER: TCP server multi-connection mode. This bit is 1, TCP server can be connected to multiple clients. For use method, please refer to 8.3.6 "Global configuration", active for all sockets.

FUN_PARA_FLAG_LOW_PWR: low power mode. This bit is 1. CH395 enters low power mode. The working mode of CH395 can be divided into standard mode and low power mode. The working current in the low power mode is about 50MA less than that in the standard mode, which is suitable for the occasions requiring high power.

FUN_PARA_FLAG_SOCKET_CLOSE: This parameter is mainly used for TCP. If this bit is 0, it will indicate that CH395 will actively turn off Socket after SINT_STAT_DISCONNECT or SINT_STAT_TIM_OUT interrupt is generated. If this bit is 1, it will indicate that MCU will turn off Socket after the above two interrupts are generated. If the Socket is turned off, CH395 will clear the internal buffer of the Socket and some related variables. In some applications, if CH395 turn off the Socket, the residual data may be forcibly cleared. In this case, MCU can read the data and then turn off the Socket.

FUN_PARA_FLAG_DISABLE_SEND_OK: Disable SINT_STAT_SEND_OK interrupt. If this bit is 1, CH395 will not generate the interrupt SINT_STAT_SEND_OK.

Note that the command CMD_SET_FUN_PARA must be set before initialization and cannot be set again once it is initialized. In addition, pay attention to the version number suitable for the parameter.

5.52. CMD_SET_KEEP_LIVE_IDLE

This command is used to set the KeepLive free time. It is necessary to input 4 bytes of time value in millisecond. This command is used to connect TCP. KeepLive free time refers to the time between no data sent and received on a TCP connection and KeepLive data packet sent. The default value is 20000.

5.53. CMD_SET_KEEP_LIVE_INTVL

This command is used to set KeepLive timout. It is necessary to input 4 bytes of time value in millisecond. This command is used to connect TCP. KeepLive timeout refers to the time for waiting for response after KeepLive data packet is sent. The default value is 15000.

5.54. CMD_SET_KEEP_LIVE_CNT

This command is used to set the number of KeepLive timeouts. It is necessary to input 1 byte of number of timeouts. This command is used to connect TCP. Number of KeepLive timeouts refers to the allowable maximum number of consecutive no response of KeepLive data packet. The default value is 9.

Assume that the free time of KeepLive is IDLE, the timeout is INTVL, and the number of timeouts is CNT.

If Socket connection enables KeepLive, Socket will start sending KeepLive packets when TCP connection is free (there is no data sent or received) within IDLE milliseconds. If the remote side responds ACK within INTVL milliseconds, the connection will be considered normal. Otherwise, the Socket considers the timeout after INTVL milliseconds and starts sending the KeepLive packet again. If no ACK packet is received within CNT times, the current connection will be considered to be disconnected and SINT_STAT_TIM_OUT or SINT_STAT_DISCONNECT will be generated.

If the three variables, IDLE, INTVL and CNT, are required to be configured, it shall be noted that IDLE must be larger than INTVL, and all of them are multiples of 500.

5.55. CMD_SET_KEEP_LIVE_SN

This command is used to enable or disable KeepLive of Socket. It is necessary to input a 1 byte of Socket index value and a 1 byte of configuration value. When the configuration value is 0, it indicates that the KeepLive function of Socket is disabled; when the configuration value is 1, it indicates that the KeepLive function is enabled. It is disabled by default.

When Socket is TCP client, use this command to enable the KeepLive function after Socket is created.
When Socket is TCP server, use this command to enable the KeepLive function after SINT_STAT_CONNECT is generated.

6. Functional Specification

6.1 Communication Interfaces of MCU

There are three communication interfaces supported between CH395 and MCU: 8-bit parallel interface, SPI synchronous serial interface and asynchronous serial port. During chip power on reset, CH395 will sample the statuses of SEL and TXD pins, and select the communication interface according to the combination of these two pin statuses. Refer to the following table (X in the table means that this bit is not concerned, 0 means low level, 1 means high level or suspended).

SEL pin	TXD pin	Select the communication interface
1	1	UART
1	0	SPI
0	1	8-bit parallel port
0	0	Wrong interface

The interrupt request of INT# pin output of CH395 chip is active at low level by default and can be connected to the interrupt input pin or ordinary input pin of MCU. MCU can get the interrupt request of CH395 in interrupt mode or query mode.

6.2. Parallel Interfaces

The parallel port signal line includes: 8-bit bidirectional data buses D7-D0, read strobe input pin RD#, write strobe input pin WR#, chip selection input pins PCS# and address input pin A0. PCS# pin of CH395 chip is driven by the address decoding circuit, which is used for device selection when MCU has multiple peripheral devices. CH395 chip can be easily hooked to the system buses of various 8-bit DSP and MCU through a passive parallel interface, and can coexist with multiple peripheral devices.

For MCU similar to the Intel parallel port timing sequence, RD# and WR# pins of CH395 chip can be connected to the read strobe output pin and write strobe output pin of MCU respectively. For MCU similar to Motorola parallel port time sequence, the RD# pin of the CH395 chip shall be connected to the low level, and the WR# pin shall be connected to the reading and writing direction output pin R/-W of MCU.

The following table is the truth table of the parallel port I/O operation (X in the table means that this bit is not concerned, and Z means that three states of CH395 are disabled).

PCS#	WR#	RD#	A0	D7-D0	Actual operation on CH395 chip
1	X	X	X	X/Z	CH395 is not selected, and no any operation is made
0	1	1	X	X/Z	Although selected, no any operation is made
0	0	1/X	1	Input	Write a command code to the command port of CH395
0	0	1/X	0	Input	Write a command code to the data port of CH395
0	1	0	0	Output	Read data from the data port of CH395

0	1	0	1	Output	Read interface status from the command port of CH395: Bit 7 is an interrupt flag, active low efficiency, equivalent to INT# pin

CH395 chip occupies two address bits. When A0 pin is at high level, write a new command, or read the interface status; when A0 pin is at low level, select the data port to read and write the data.

MCU reads and writes CH395 chip through an 8-bit parallel port. All operations are composed of a command code, several input data and several output data. Some commands do not need input data, and some commands do not have output data. The command operation steps are as follows:

① MCU writes the command code to the command port when A0 is 1;
② If the command has input data, write the input data in sequence when A0 is 0, one byte at a time;
③ If the command has output data, read the output data in sequence when A0 is 0, one byte at a time;
④ The command is completed. Execution may be required to be queried for some commands. MCU can be paused or go to ① to continue to execute the next command.

6.3. SPI

The SPI signal lines include: SPI chip selection input pin SCS, serial clock input pin SCK, serial data input pin SDI and serial data output pin SDO. CH395 can be hooked to SPI serial buses of various DSP and MCU with fewer connections through SPI serial interface by using less connecting wires, or be connected point-to-point over a longer distance.

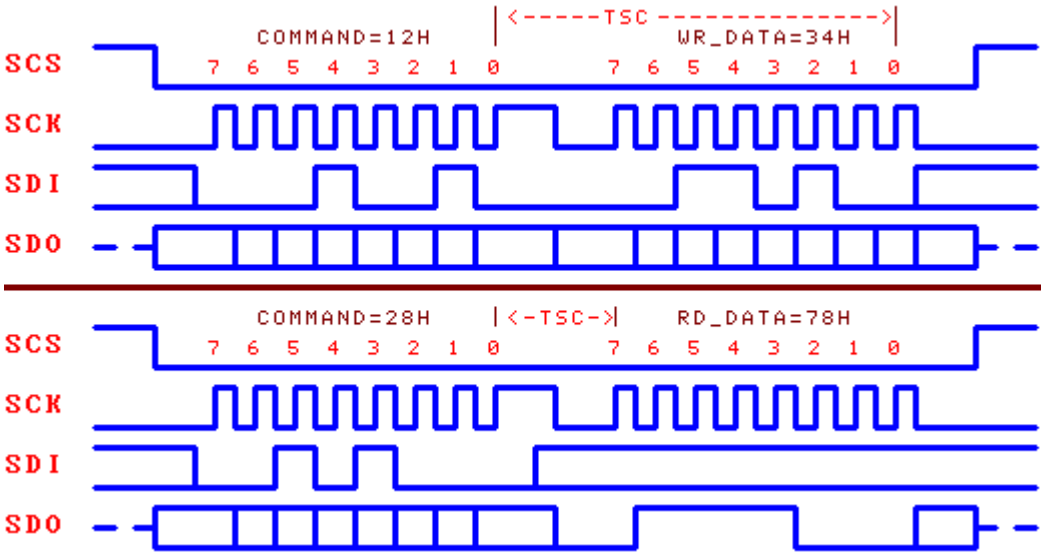
The SCS pin of CH395 chip is driven by the SPI chip selection output pin or the general output pin of MCU. SCK pin is driven by the SPI clock output pin SCK of MCU. SDI pin is driven by the SPI data output pin SDO or MOSI, and SDO pin is connected to the SPI data input pin SDI or MISO of MCU. For the hardware SPI interface, it is recommended that the SPI setting is CPOL=CPHA=0 or CPOL=CPHA=1, and the data bit sequence is MSB first. SPI interface of CH395 supports MCU to simulate SPI interface for communication with the common I/O pins.

SPI interface of CH395 supports SPI mode 0 and SPI mode 3. CH395 always inputs data from the rising edge of the SPI clock SCK, and outputs data from the falling edge of SCK when the output is allowed. The data bit sequence is MSB first, and 8 full bits are a byte.

Steps for SPI operation:

① MCU generates the SPI chip selection of CH395 chip, which is active at low level;
② MCU sends a byte of data in SPI output mode. CH395 always takes the first byte received after SPI chip selection SCS is valid as the command code and takes the subsequent bytes as data;
③ MCU delays for TSC time (about 1.5uS) to wait for SPI interface of CH395 to be free;
④ If it is a write operation, MCU sends a byte of write data to CH395. After waiting for SPI interface to be free, MCU continues to send several bytes of write data, and CH395 receives them in turn until MCU disables SPI chip selection.
⑤ If it is a read operation, MCU receives a byte of data from CH395. After waiting for SPI interface to be free, MCU continues to receive several bytes of data from CH395 until MCU disables SPI chip selection;
⑥ MCU disables the SPI chip selection of CH395 chip to end the current SPI operation.

The figure below is the SPI interface logic sequence diagram. The first one sends the command 12H and writes 34H, and the second one sends the command 28H and reads the data 78H.



6.4. UART

The signal line of UART includes serial data input pin RXD and serial data output pin TXD. CH395 can be connected point-to-point to DSP and MCU through a serial interface by using less connecting wires over a longer distance.

RXD and TXD of CH395 chip can be connected to the serial data output pin and serial data input pin of MCU respectively.

The serial data format of CH395 is the standard byte transmission mode, consisting of 1 start bit, 8 data bits and 1 stop bit.

CH395 not only supports hardware to set the default serial communication baud rate, but also supports MCU to select the appropriate communication baud rate through the command CMD_SET_BAUDRATE at any time. After each power on reset, the default serial communication baud rate of CH395 is set by the level combination of three pins SDO, SDI and SCK. Refer to the following table (0 represents low level, and 1 represents high level or suspended).

SDO pin	SDI pin	SCK pin	Default communication baud rate after power on reset
1	1	1	9600 bps
1	1	0	57600 bps
1	0	1	115200 bps
1	0	0	460800 bps
0	1	1	250000 bps
0	1	0	1000000 bps
0	0	1	3000000 bps
0	0	0	921600 bps

In order to distinguish the command code from the data, CH395 requires MCU to first send two synchronous code bytes (57H and ABH) through UART, then send the command code, and then send or receive the data. CH395 will check the interval between the above two synchronous code bytes and between the synchronous code and the command code. If the interval is greater than the serial input timeout of SER_CMD_TIMEOUT (approximately 40mS), CH395 will discard the synchronous code and the command packet. The operation steps of serial port command are as follows:

① MCU sends the first synchronous code 57H to CH395 through the serial port;

② MCU sends the second synchronous code 0ABH to CH395;

③ MCU sends the command code to CH395;
④ If the command has input data, MCU will send the input data to CH395 in turn, one byte at a time;
⑤ If the command has output data, MCU will receive the output data from CH395 in turn, one byte at a time;
⑥ The command is completed. Interrupt notification may be generated after some commands are executed, and the interrupt status code will be directly sent through the serial port. MCU can be paused or go to ① to continue to execute the next command.

6.5. Other Hardware

CH395 chip integrates 10/100M Ethernet MAC and PHY, CRC data check, passive parallel interface, SPI-Slave controller, asynchronous serial port, SRAM, high-speed MCU, firmware program, crystal oscillator and PLL frequency multiplier, power on reset circuit, etc.

ELINK# pin of CH395 chip is used for Ethernet status connection and communication indication. It can be externally connected to LED with a current limiting resistor connected in series to indicate connection and communication status.

RXP, RXN, TXP and TXN of CH395 chip are Ethernet signal lines. PHY of CH395 supports automatic switching of MDI/MDIX line, but only valid in automated negotiation mode.

CH395 chip has a built-in power on reset circuit. Generally, no external reset is required. RSTI pin is used to input an asynchronous reset signal from the outside; when RSTI pin is at low level, CH395 chip will be reset; when RSTI pin recovers to a high level, CH395 will continuously delay reset for about 35mS, and then enter the normal working status. In order to reliably reset and reduce external interference during the power-on period, a capacitor with a capacity of about 0.1uF can be connected across the RSTI pin and the ground.

RST pin of CH395 chip is an active high reset status output pin, which can be used to provide a power on reset signal to the external MCU. RST pin outputs high level when CH395 is reset during power-on or externally forced to be reset and during reset delay; after CH395 is reset and the communication interface is initialized, RST pin recovers to the low level.

When CH395 chip works normally, 30MHz clock signal shall be provided for it externally. CH395 chip has a built-in crystal oscillator and an oscillating capacitor. Generally, the clock signal is generated by the built-in oscillator of CH395 through a crystal stable frequency oscillator, and the peripheral circuit is only required to be connected with a crystal with a nominal frequency of 30MHz between XI and XO pins.

7. Parameters

7.1. Absolute Maximum Value

Critical value or exceeding the absolute maximum value may cause the chip to work abnormally or even be damaged.

Name	Parameter description		Min.	Max.	Unit
TA	Ambient temperature during operation	VCC33=3.3V VCC18=1.8V	-40	85	℃
TS	Ambient temperature during storage		-55	125	℃
VCC33	Supply voltage (VCC33 connects to power, GND to ground)		-0.4	4.2	V
VCC18	Supply voltage (VCC18 connects to power, GND to ground)		-0.4	2.3	V
VIO	Voltage on the input or output pins		-0.4	VCC33+0.4	V
VIO5	Support voltage on 5V withstand voltage input or output pin		-0.4	5.4	V

7.2. Electrical Parameters

Test Conditions: TA=25℃, VCC33=3.3V, VCC18=1.8V

Name	Parameter description		Min.	Typ.	Max.	Unit
VCCxx	Power voltage	VCC33	2.7	3.3	3.6	V
		VCC18	1.65	1.8	1.95	
ICC	Total supply current during operation	VCC33=3.3V		90(50)	220(195)	mA
ISLP	Supply current at the low power status I/O pin suspended/ internal pull-down	VCC33=3.3V		15		mA
VIL	Low level input voltage		-0.4		0.7	V
VIH	High level input voltage		2.0		VCC33+0.4	V
VOL	Low level output voltage (4mA draw current)				0.4	V
VOH	High level output voltage (4mA output current)		VCC33-0.4			V
IUP	Input current at the input terminal of built-in pull-up resistor		20	40	100	uA
IDN	Input current at the input terminal of built-in pull-down resistor		-20	-40	-100	uA
VR	Voltage threshold of power-on reset		1.4	1.5	2.5	V

Note: The parameters in brackets are in low power mode.

7.3. Timing Parameters

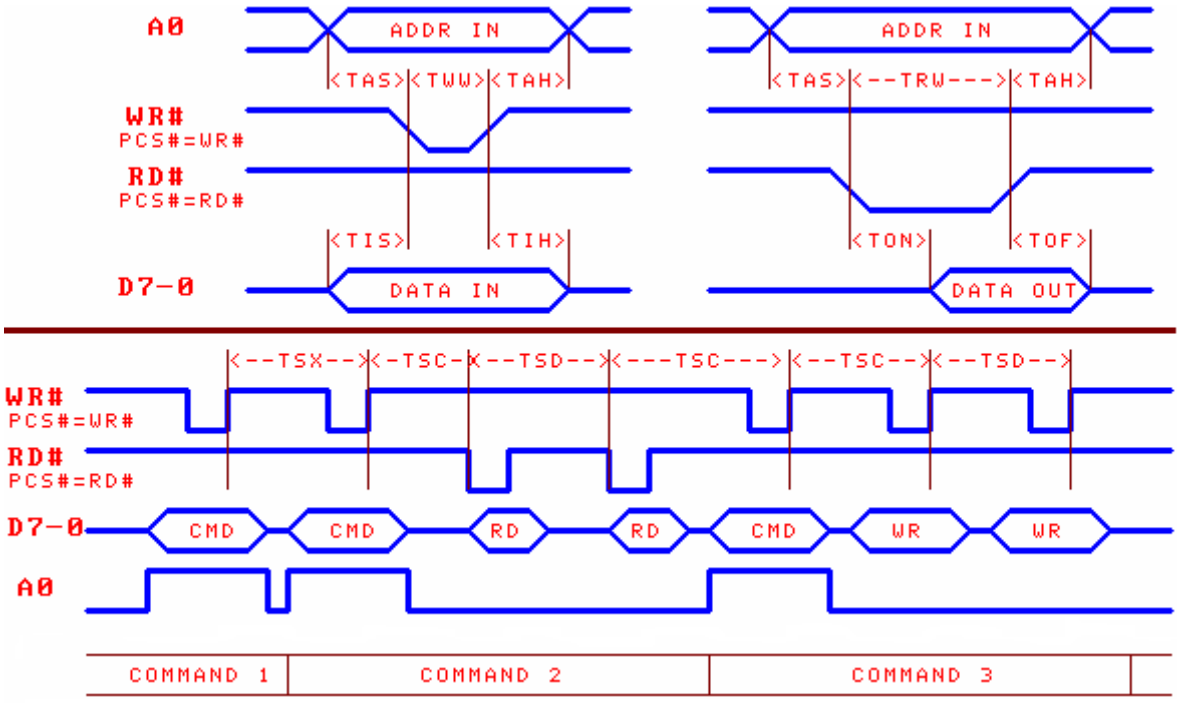
Test Conditions: TA=25℃, VCC33=3.3V, VCC18=1.8V, refer to attached figure

Name	Parameter description	Min.	Typ.	Max.	Unit
FCLK	Input clock frequency of XI pin	29.995	30.00	30.005	MHz
TPR	Internal power-on reset time	15	50	60	mS
TRI	Effective signal width of external reset input	100			nS
TRD	Reset delay after external reset input	15	50	60	mS
TWAK	Wake-up time when exiting from low-power state	3	7	15	mS
TE1	Execution time of command CMD_RESET_ALL	10	50	60	mS
TE2	Execution time of command CMD_INIT_CH395	300	350	400	mS
TE3	Execution time of command with status to be gotten		1	5	mS
TE4	Execution time of command CMD_SET_BAUDRATE	200	1000	2000	uS
TE0	Execution time of other commands		0.8	1.5	uS
TSX	Interval time between command codes	1.5			uS
TSC	Interval time between command code and data	0.6			uS
TSD	Interval time between data and data	0.3			uS
TINT	Receive the command GET_STATUS until INT# pin undoes the interrupt		1	2	uS

7.4. Timing Parameters of Parallel Port

Test Conditions: TA=25℃, VCC33=3.3V, VCC18=1.8V, refer to attached figure.
(RD means that RD# signal is valid and PCS# signal is valid; perform read operation when RD#=PCS#=0)
(WR means that WR# signal is valid and PCS# signal is valid, perform write operation when WR#=PCS#=0)
(In low power mode, all parameters in the table are multiplied by 2)

Name	Parameter description	Min.	Typ.	Max.	Unit
TWW	Write pulse width	30			nS
TRW	Read pulse width	35			nS
TAS	Address input setup time before RD or WR	3			nS
TAH	Address input hold time after RD or WR	3			nS
TIS	Data input setup time before write strobe WR	0			nS
TIH	Data input hold time after strobe writing WR	3			nS
TON	Read active to valid data output	2	7	15	nS
TOF	Read inactive to invalid data output	3	12	25	nS

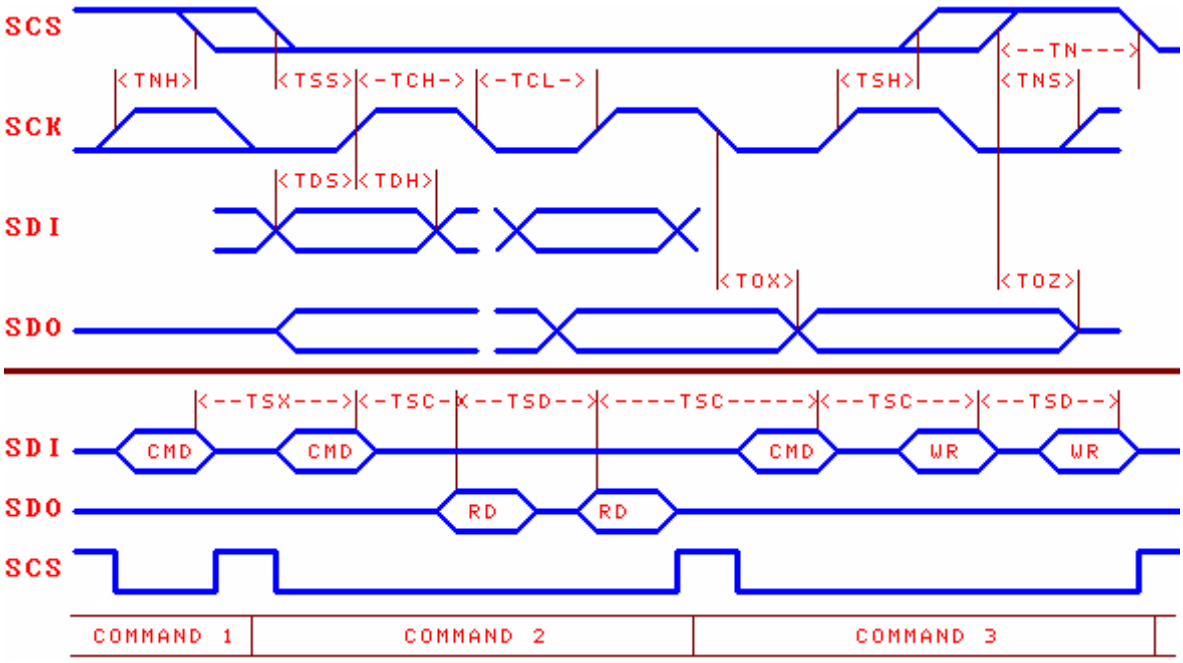


7.5. SPI Timing Parameters

Test Conditions: TA=25℃, VCC33=3.3V, VCC18=1.8V, refer to attached figure.
(In low power mode, all parameters in the table are multiplied by 2)

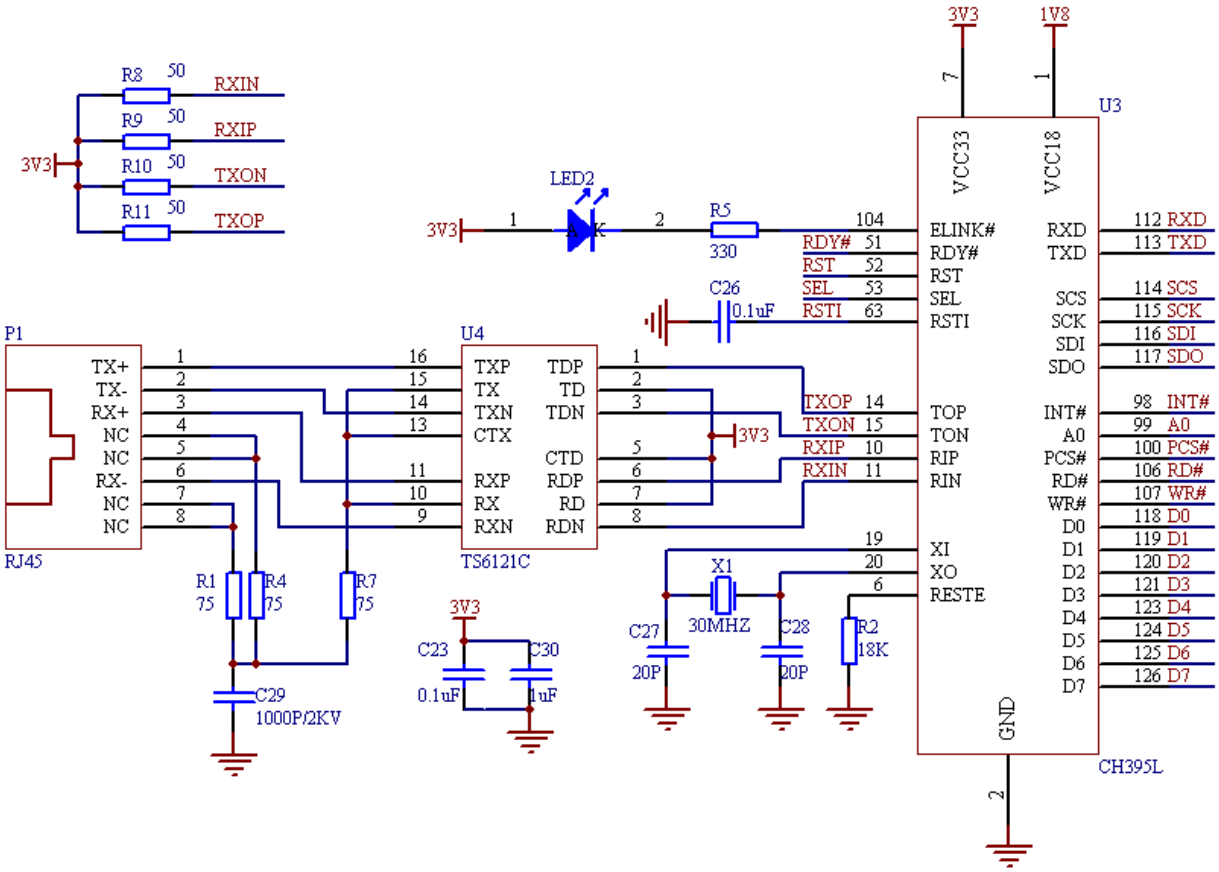
Name	Parameter description	Min.	Typ.	Max.	Unit
TSS	Setup time of valid SCS before SCK rising edge	25			nS
TSH	Hold time of valid SCS after SCK rising edge	12			nS
TNS	Setup time of invalid SCS before SCK rising edge	12			nS
TNH	Hold time of invalid SCS after SCK rising edge	12			nS
TN	Time of invalid SCS (SPI operation interval time)	60			nS
TCH	SCK clock high-level time	15			nS

TCL	SCK clock low-level time	15			nS
TDS	SDI input setup time before SCK rising edge	4			nS
TDH	SDI input hold time after SCK rising edge	2			nS
TOX	Output change from SCK falling edge to SDO	2	6	12	nS
TOZ	SCS invalid to SDO output invalid	3	10	18	nS



8. Application

8.1 Hardware Circuit Design



Note: for space limitation, the figure above is a schematic diagram, omitting power supply related decoupling capacitor circuit, etc. In order to improve the stability of the chip, each power pin is connected with a 0,1uF decoupling capacitor near to the ground. For detailed schematic diagram, please refer to our schematic diagram document CH395SCH.PDF.

If CH395 is required to be configured as 8-bit parallel communication mode PARALLEL, SEL pin shall be connected to GND and TXD pin shall be suspended. Pins used for communication between this interface and peripheral MCU are A0, PCS#, RD#, WR#, D0-D7, INT# (optional) and RSTI (optional).

If CH395 is required to be configured as SPI serial communication mode, TXD pin shall be connected to GND and SEL pins shall be suspended. Pins used for communication between this interface and peripheral MCU are SCS, SCK, SDO, SDI, INT# and RSTI (optional).

If CH395 is required to be configured as asynchronous serial port communication mode UART/SERIAL, SEL pin and TXD pin shall be suspended. Pins used for communication between this interface and peripheral MCU are TXD, RXD, INT# and RSTI (optional). The default serial communication baud rate is set by the three pins of SDO, SDI and SCK. If the communication baud rate of CH395 serial port is required to be dynamically modified, it will be suggested that the I/O pin of MCU control RSTI pin of CH395, so as to reset CH395 to return to the default communication baud rate when necessary.

As INT# pins and TXD pin can only provide weak high level output current during CH395 reset, when a long distance connection is conducted, in order to avoid INT# or TXD interference caused by MCU misoperation during CH395 reset, a pull-up resistor with resistance of 2KΩ-5KΩ can be added on INT# pin or TXD pin to maintain a stable high level. After CH395 chip reset is completed, INT# and TXD pins will be able to provide either a high level output current of 4mA or a low level sinking current of 4mA.

In the 8-bit parallel port mode, the interface status is gotten by inquiring the status port of CH395 (namely, the command port). Bit 7 is an interrupt flag bit, which is active low and equivalent to the query of INT# pin.

If bit 7 is 0, there will be an interrupt request. In SPI and UART modes, interrupts must be gotten through INT#.

P1 is RJ45 port, used to connect network equipment such as switch and router. It includes two pairs of Ethernet differential signals, unused pins of RJ45 shall be connected with a 1000P/2KV capacitor to the ground through 75Ω resistor.

U4 is a network transformer with electrical isolation, impedance matching and other functions. The center tap on the inside (close to CH395) shall be connected to 3.3V, and the unused pins on the outside (close to RJ45) shall be connected to 1000P/2KV capacitor to the ground through 75Ω resistor.

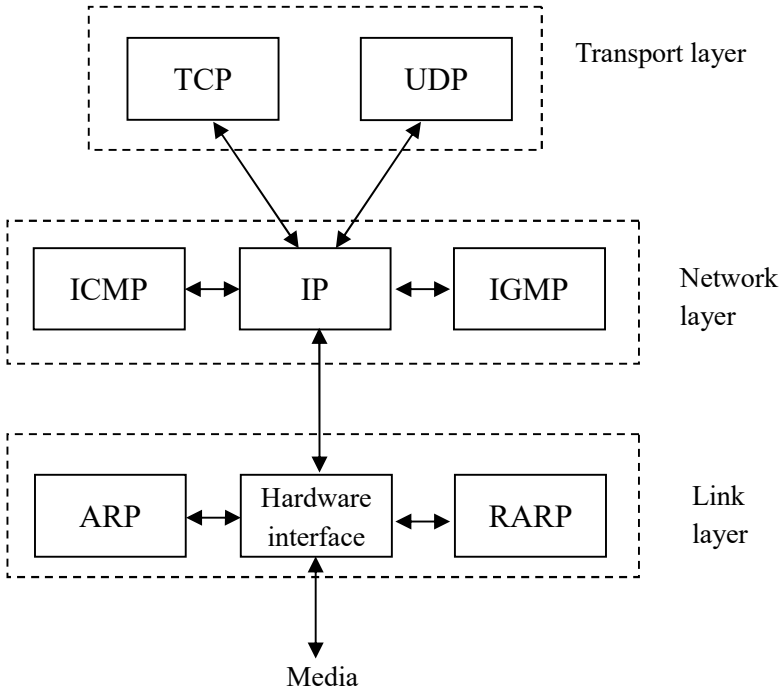
R2 is a regulating resistor for Ethernet signals. 12-18K resistor can be selected. 12K resistor is the best and 18K resistor has the lowest power.

GPIO is not given in this figure. Please refer to the schematic diagram of CH395EVT for details.

When PCB is actually made, R8-R11, C23, C30 shall be as close as possible to the pin 5 of U4. TXOP (RXIP) and TXON(RXIN) are differential signals, which shall be wired close to the parallel line. The ground wire shall be provided or copper shall be clad as far as possible on both sides to reduce interference from the outside. The length of relevant signals of crystals XI and XO shall be shortened as much as possible. In order to reduce the interference of high-frequency clock with the outside, the baselines shall be surrounded or copper shall be clad around relevant components.

8.2. Application Basis

CH395 internally integrates IPv4, ARP, ICMP, IGMP, UDP, TCP and other protocols, and their relationship diagram is as follows:



TCP and UDP are two important transport layer protocols, both of which use IP as the network layer protocol.

TCP is a connection-oriented transport layer protocol, which can provide reliable byte stream transport services.

UDP is a simple datagram-oriented transport layer protocol. Different from TCP, UDP cannot guarantee that the datagram reaches the destination accurately.

TCP provides high reliability communication for network devices, its work includes: delivering the application to its data and dividing it into the suitable blocks, delivering them to the network layer below, confirming the received packets, set the timeout clock, etc. As the transport layer provides high reliability end-to-end communication, the application layer client ignores all the details. UDP provides a very simple service for the application layer, which is faster than TCP. It just sends the datagram from one network terminal to the other network terminal, but it does not guarantee that the datagram can reach the other end. Any necessary reliability must be provided by the application layer.

IP is the protocol on the network layer, which is simultaneously used by both TCP and UDP. Each set of data of TCP and UDP is transmitted in the network through the IP layer.

ICMP is an ancillary protocol of IP protocol, which is used by IP layer to exchange error messages or other important information with other hosts or routers, for example, when CH395 generates inaccessible interrupt, error messages are exchanged through ICMP. PING also uses the ICMP protocol.

IGMP is an Internet group management protocol, which is mainly used to multicast a UDP datagram to multiple hosts.

ARP is an address resolution protocol, which is used to translate the addresses used by the IP layer and the network interface layer.

For basic formats such as Ethernet frame, IP, UDP, TCP packet, etc., please refer to Section 8.3.

8.3. Application Reference Steps

This chapter introduces the common operation steps. Refer to the example program for details.

8.3.1. Initialize CH395 (necessary operation)

① Send the command CMD_SET_MAC_ADDR to set MAC address of CH395;
② Send the command CMD_SET_IP_ADDR to set IP address of CH395;
③ Send the command CMD_SET_GWIP_ADDR to set the gateway IP address of CH395;
④ Send the command CMD_SET_MASK_ADDR sets the subnet mask of CH395;
⑤ Send the command CMD_INIT_CH395 to initialize CH395;
⑥ After delay for more than 2MS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_INIT_CH395. If CH395_ERR_BUSY is returned, it will indicate that CH395 is internally executing the command and needs to execute ⑥ again; If CH395_ERR_SUCCESS is returned, it will indicate that the command is executed successfully. Generally, it takes 350mS to complete CMD_INIT_CH395.

Generally, the step ① is not required. MAC address assigned by IEEE has been burned when CH395 chip is delivered.

If DHCP is required to be started, the steps ②-④ will not be required.

The step ④ is optional. The default subnet mask is 255.255.255.0, which is generally not required to be set.

After CH395 receives the command CMD_INIT_CH395, it initializes the internal TCP/IP stack, MAC and PHY, and then MAC and PHY are initialized to the automated negotiation mode. If a PHY is required to work in other modes, such as 10M full-duplex mode, the command CMD_SET_PHY is required to be sent for setting after successful execution of CMD_INIT_CH395.

8.3.2. Initialize Socket to MACRAW mode

Initialization steps are as follows:
① Send the command CMD_SET_PROTO_TYPE_SN to set Socket to work in MACRAW mode;
② Send the command CMD_OPEN_SOCKET_SN to turn on Socket;

③ After delay for more than 2MS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_OPEN_SOCKET_SN. If CH395_ERR_BUSY is returned, it will indicate that CH395 is internally executing the command and needs to execute ③ again; If CH395_ERR_SUCCESS is returned, it will indicate that the command is executed successfully. Other values indicate the failure to turn on Socket.

IEEE802.3 Ethernet frame format:

Destination MAC	Source MAC	Type	Data	CRC32
6 Byte	6 Byte	2 Byte	46-1500 Byte	4 Byte

In MACRAW mode, CH395 will transparently transmit the data between Ethernet and MCU without TCP/IP data encapsulation. When CH395 receives the data, it will check the Ethernet redundant check CRC32. If the check is wrong, the data packet will not be forwarded to MCU. When CH395 sends data, Ethernet redundant check CRC32 is added at the end of data packet. The length of data written by MCU to CH395 shall not be more than 1514 each time, and CH395 will encapsulate the data written by MCU into a frame for sending. When CH395 receives data from Ethernet, it will notify MCU. At this time, MCU shall immediately read all data from the internal receive buffer of CH395.

Only Socket0 can be set to this mode, and other sockets will not be available.

8.3.3. Initialize Socket to IPRAW mode

Initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set Socket to work in IPRAW mode;
② Send the command CMD_SET_IP_ADDR_SN to set the destination IP address;
③ Send the command CMD_SET_IPRAW_PRO_SN to set the protocol field;
④ Send the command CMD_OPEN_SOCKET_SN to turn on Socket;
⑤ After delay for more than 2MS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_OPEN_SOCKET_SN. If CH395_ERR_BUSY is returned, it will indicate that CH395 is internally executing the command and needs to execute ⑤ again; If CH395_ERR_SUCCESS is returned, it will indicate that the command is executed successfully; if other values are returned, it will indicate the failure to turn on Socket.

IP message structure:

Destination MAC	Source MAC	Type	IP header	IPRAW data	CRC32
6 Byte	6 Byte	2 Byte	20 Byte	Max. 1480 Bytes	4 Byte

After MCU writes several bytes of data stream to CH395, CH395 encapsulates the protocol field of this Socket in the IP header and encapsulates the data stream in the IPRAW data division for sending. The maximum length allowed to be sent per packet in IPRAW mode is 1480 bytes. If the length of the data stream written by MCU is more than 1480 bytes, CH395 will encapsulate the data stream into several IP packets for sending, and the interrupt SINT_STAT_SEND_OK will be generated after each packet is successfully sent. The length of byte written by MCU each time shall not be more than the length of the transmit buffer, and the next write data can only be performed after the interrupt SINT_STAT_SENBUF_FREE is received. If the interrupt SINT_STAT_TIM_OUT is generated, it will indicate the failure to send data. There are usually two reasons for failure to send data:

① If the destination IP address and CH395 are on the same subnet, the network device at the destination IP address may not be online.
② If the destination IP address and CH395 are not on the same subnet, the gateway of CH395 may not be online.

When receiving IP data packets, CH395 first tests whether the protocol field is the same as that set by Socket. If so, CH395 will copy IPRAW data packets to the receive buffer and generate SINT_STAT_RECV interrupt. Upon receipt of this interrupt, MCU can send the command CMD_GET_RECV_LEN_SN to get the length of the receive buffer data, and then send the command CMD_READ_RECV_SN to read the data in the data buffer. MCU can read all the data at a time or read it for several times. As CH395 cannot perform flow control in IPRAW mode, it is suggested that MCU shall read all the data immediately after querying the interrupt port of the received data, so as not to be covered by the subsequent data.

Notes on protocol field settings
CH395 has a higher IPRAW processing priority than UDP and TCP. If the IP protocol field is set to 17 (UDP) or 6 (TCP), there may be the possibility of conflicting with other Sockets, which shall be avoided when in use. The following two situations are listed for explanation:

① Socket0 is set to IPRAW mode, IP protocol field is set to 17, and Socket1 is set to UDP mode. In UDP mode, the IP packet protocol field is also 17, so Socket1 communication data will be intercepted by Socket0, and data cannot be received.
② Socket0 is set to IPRAW mode, IP protocol field is set to 6, and Socket1 is set to TCP mode. In TCP mode, the IP packet protocol field is also 6, so Socket1 communication data will be intercepted by Socket0, and data cannot be received.

8.3.4. Initialize Socket to UDP mode

Initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set Socket to work in UDP mode;
② Send the command CMD_SET_IP_ADDR_SN to set the destination IP address;
③ Send the command CMD_SET_DES_PORT_SN to set the destination port;
④ Send the command CMD_SET_SOUR_PORT_SN to set the source port;
⑤ Send the command CMD_OPEN_SOCKET_SN to turn on Socket;
⑥ After delay for more than 2MS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_OPEN_SOCKET_SN. If CH395_ERR_BUSY is returned, it will indicate that CH395 is internally executing the command and needs to execute ⑥ again; If CH395_ERR_SUCCESS is returned, it will indicate that the command is executed successfully; if other values are returned, it will indicate the failure to turn on Socket.

UDP message structure:

Destination MAC	Source MAC	Type	IP header	UDP header	UDP data	CRC32
6 Byte	6 Byte	2 Byte	20 Byte	8 Byte	Max. 1472 Bytes	4 Byte

UDP is a simple, unreliable, datagram-oriented transport layer protocol with fast transmission speed, which cannot guarantee that the data can reach the destination. Therefore, the application layer must guarantee the reliable and stable transmission.

After MCU writes several bytes of data stream to CH395, CH395 encapsulates the data stream in the UDP data division for sending. The maximum length sent per packet in UDP mode is 1472 bytes. If the length of the data stream written by MCU is more than 1472 bytes, CH395 will encapsulate the data stream into several UDP packets for sending, and the interrupt SINT_STAT_SEND_OK will be generated after each packet is successfully sent. The length of byte written by MCU each time shall not be more than the length of the transmit buffer, and the next write data can only be performed after the interrupt SINT_STAT_SENBUF_FREE is received. The interrupt SINT_STAT_TIM_OUT will be generated if data sending fails. There are usually two reasons for failure to send data:

① If the destination IP address and CH395 are on the same subnet, the network device at the destination IP address may not be online.

② If the destination IP address and CH395 are not on the same subnet, the gateway of CH395 may not be online.

When receiving UDP message, CH395 copies UDP data to the Socket receive buffer and generates SINT_STAT_RECV interrupt. Upon receipt of this interrupt, MCU can send the command CMD_GET_RECV_LEN_SN to get the length of the receive buffer data, and then send the command CMD_READ_RECV_SN to read the data in the data buffer. As CH395 cannot perform flow control in UDP mode, it is suggested that the received data shall be read quickly in time, so as not to be covered by the subsequent data.

CH395 supports two UDP modes: UDP client and UDP server. UDP client can only communicate with the specified IP and port, and UDP server can communicate with any remote IP and port.

There are some differences in usage:

① Initialization step ②. If the destination IP address is 0XFFFFFFFF, this Socket will be in UDP server mode, otherwise it will be in UDP client mode.

② MCU reads data from CH395. CH395 directly sends the received data stream to MCU in the client mode, and MCU can read all the data at a time or read only part of the data. In the server mode, CH395 will add an 8-byte information table to the header of the data. MCU can get the source information of the data packet according to the information table, and must read all the data at a time.

Data packet length	Port	IP	Data
2 Byte	2 Byte	4 Byte	N Byte

③ MCU sends data, and CH395 directly sends data to the destination IP and port specified during initialization in the client mode. In the server mode, CH395 can send data to any IP and port. MCU sets the destination IP and port before sending.

8.3.5. Initialize Socket to TCP client mode

Initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set Socket to work in TCP mode;

② Send the command CMD_SET_IP_ADDR_SN to set the destination IP address;

③ Send the command CMD_SET_DES_PORT_SN to set the destination port;

④ Send the command CMD_SET_SOUR_PORT_SN to set the source port;

⑤ Send the command CMD_OPEN_SOCKET_SN to turn on Socket;

⑥ After delay for more than 2MS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_OPEN_SOCKET_SN. If CH395_ERR_BUSY is returned, it will indicate that CH395 is internally executing the command and needs to execute ⑥ again; If CH395_ERR_SUCCESS is returned, it will indicate that the command is executed successfully; if other values are returned, it will indicate the failure to turn on Socket;

⑦ Send the command CMD_TCP_CONNECT_SN to perform TCP connection;

⑧ After delay for more than 2MS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_TCP_CONNECT_SN. If CH395_ERR_BUSY is returned, it will indicate that CH395 is internally executing the command and needs to execute ⑧ again; If CH395_ERR_SUCCESS is returned, it will indicate that the command is executed successfully. Other values indicate the failure to execute the command. Returning CH395_ERR_SUCCESS only indicates that the command is executed successfully and does not indicate a successful TCP connection. If TCP connection is successful, CH395 will generate the interrupt SINT_STAT_CONNECT. If the connection fails, CH395 will generate the interrupt SINT_STAT_TIM_OUT. If it is necessary to connect again, execute again from ⑤.

TCP message structure:

Destination MAC	Source MAC	Type	IP header	TCP header	TCP data	CRC32
6 Byte	6 Byte	2 Byte	20 Byte	20 Byte	Max. 1460 Bytes	4 Byte

TCP provides a connection-oriented, reliable byte stream service.

CH395 generates SINT_STAT_CONNECT, indicating that TCP connection is established and data can be sent and received. Data sending operation shall not be performed before the connection is not established.

After MCU writes several bytes of data stream to CH395, CH395 encapsulates the data stream in the TCP data division for sending. The maximum length sent per packet in TCP mode is TCP MSS bytes. If the length of the data stream written by MCU is more than TCP MSS bytes, CH395 will encapsulate the data stream into several TCP packets for sending, and the interrupt SINT_STAT_SEND_OK will be generated after each packet is successfully sent. The length of byte written by MCU each time shall not be more than the length of the transmit buffer, and the next write data operation can only be performed after the interrupt SINT_STAT_SENBUF_FREE is received. In TCP mode, the interrupt SINT_STAT_TIM_OUT will be generated if the data transmission fails. CH395 will automatically turn off the Socket (in case FUN_PARA_FLAG_SOCKET_CLOSE is 0, see 5.55).

When receiving TCP message, CH395 copies TCP data to the Socket receive buffer and generates SINT_STAT_RECV interrupt. Upon receipt of this interrupt, MCU can send the command CMD_GET_RECV_LEN_SN to get the length of the receive buffer data, and then send the command CMD_READ_RECV_SN to read the data in the data buffer. MCU can read all the data at one time or only read part of the data. The free space of the receive buffer is TCP window. After the MCU reads the data each time, CH395 will check the free space of the receive buffer and inform the TCP server of the size of the current window.

8.3.6. Initialize Socket to TCP server mode

Initialization steps are as follows:

① Send the command CMD_SET_PROTO_TYPE_SN to set Socket to work in TCP mode;
② Send the command CMD_SET_SOUR_PORT_SN to set the source port Sport;
③ Send the command CMD_OPEN_SOCKET_SN to turn on Socket;
④ After delay for more than 2MS, send the command CMD_GET_CMD_STATUS to get the execution status of CMD_OPEN_SOCKET_SN. If CH395_ERR_BUSY is returned, it will indicate that CH395 is internally executing the command and needs to execute ④ again; If CH395_ERR_SUCCESS is returned, it will indicate that the command is executed successfully; if other values are returned, it will indicate the failure to turn on Socket.

In TCP server mode, if the client is connected and always be in the monitoring state on the Socket, the timeout interrupt will not be generated. If TCP connection is successful, CH395 will generate the interrupt SINT_STAT_CONNECT. At this time, MCU can send the command CMD_GET_REMOT_IPP_SN to get the IP address and port number of the client.

By default, the multi-connection function of the server is turned off. In TCP server mode, only one TCP connection can be established for each Socket.

In case the multiple connection mode is enabled, TCP server can be connected to multiple TCP connections, MCU shall set the source port of the Socket to be consistent with the source port of the server. If TCP server detects the connection, CH395 will detect whether all current source ports of the Socket are consistent with the current server, whether the protocol type is TCP and whether they are in OFF state. If found, CH395 will immediately turn on this Socket, assign the connection to this Socket and notify MCU of this connection event; if not found, CH395 will reset the connection. In this mode, the Socket of the server is only used for

monitoring, and MCU needs to allocate other Sockets for the connection of the server. For example, Socket0 is set to the server mode, Socket1 and Socket2 are used for the connection of this server, and the steps are as follows:

Socket0 executes ①-④;

⑤ Send the command CMD_SET_SOUR_PORT_SN to Socket1 to set the source port Sport;

⑥ Send the command CMD_SET_PROTO_TYPE_SN to Socket1 to set Socket to work in TCP mode;

⑦ Send the command CMD_SET_SOUR_PORT_SN to Socket2 to set the source port Sport;

⑧ Send the command CMD_SET_PROTO_TYPE_SN to Socket2 to set Socket to work in TCP mode;

For data structures and the process for sending and receiving data, refer to the TCP client mode.

8.3.7. DHCP

Before beginning the following steps, initialize CH395.

DHCP steps are as follows:

① Send the command CMD_DHCP_ENABLE, and start DHCP if the parameter is 1.

② Wait for CH395 to generate the interrupt GINT_STAT_DHCP;

③ After waiting for the interrupt GINT_STAT_DHCP, send the command CMD_GET_DHCP_STATUS to get DHCP status. If the status code is 0, it will indicate success, and MCU can send the command CMD_GET_IP_INF to get IP, MASK and other information. If the status code is 1, the connection between CH395 and DHCP Server may fail, for example, DHCP Server is not online. Although CH395 informs MCU of DHCP error by the interrupt, it will still try again internally to find DHCP Server. MCU can send the command CMD_DHCP_ENABLE. Stop DHCP if the parameter is 0.

8.3.8. About TCP MSS and Buffers

CH395 supports modification of TCP MSS. The default size of TCP MSS is 800. Generally speaking, if TCP MSS is larger, the communication speed and efficiency will be higher. Some principles shall be followed when TCP MSS and the receive buffer are modified.

① (Suggested) The length of the receive buffer shall not be less than 2 times of TCP MSS;

② (Necessary) The length of the receive buffer shall not be less than TCP MSS;

③ (Suggested) The length of the receive buffer shall not be more than 6 times of TCP MSS;

④ (Necessary) The size of the transmit buffer shall not be more than 8KB.

When modifying TCP MSS, try to be as large as possible. If it is too small, it may lead to the increase of small packet data on Ethernet, affecting the communication efficiency.