STAND-ALONE ETHERNET CHAT CLIENT

By :

Anurag Dwivedi

Avinash Bhattarmakki

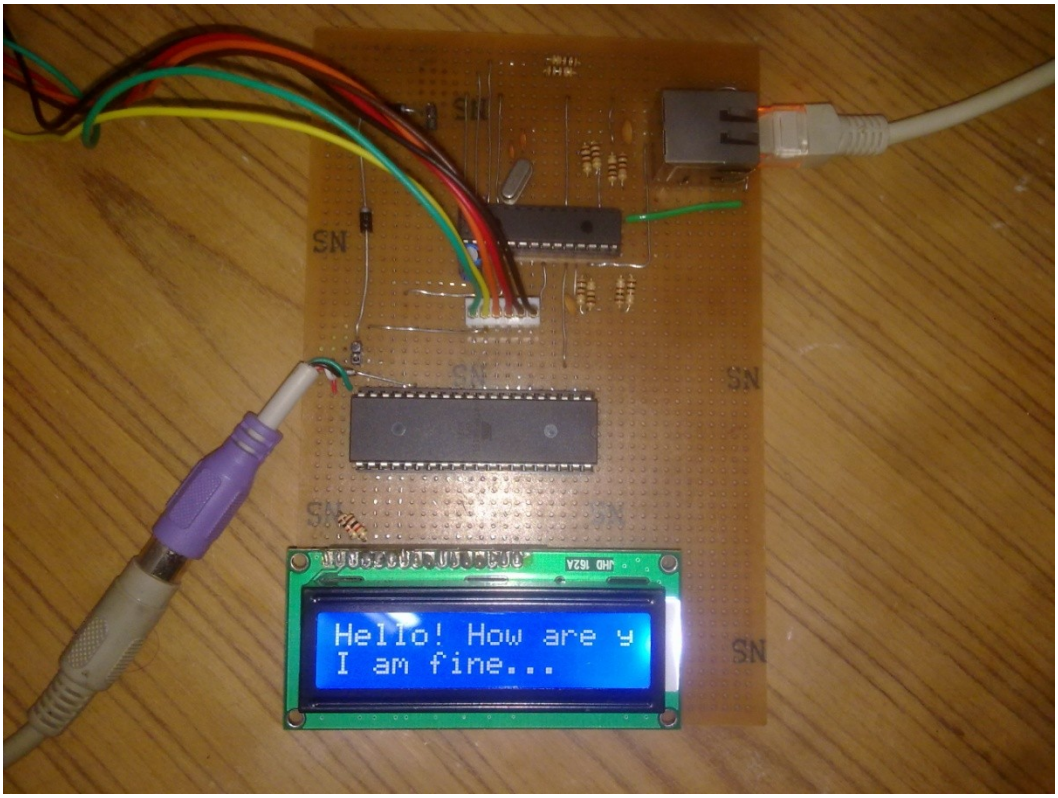
Nitish Kumar Srivastava

PROJECT AIM

To develop a module that can communicate with a computer over ethernet and can perform various tasks according to the instructions given by the computer…

THEORY

The communication over Ethernet is done via 802.33 ethernet standards. The IC ENC28J60 is capable of transmitting and receiving data from Ethernet and storing it in its buffer. The IC can also communicate with any host controller via SPI protocol. In Our project we have controlled the IC through an Atmega to send and receive data packets over Ethernet. The host controller also distinguishes the simple text messages from commands and processes them accordingly, displaying chat messages on a lcd and performing the commands…



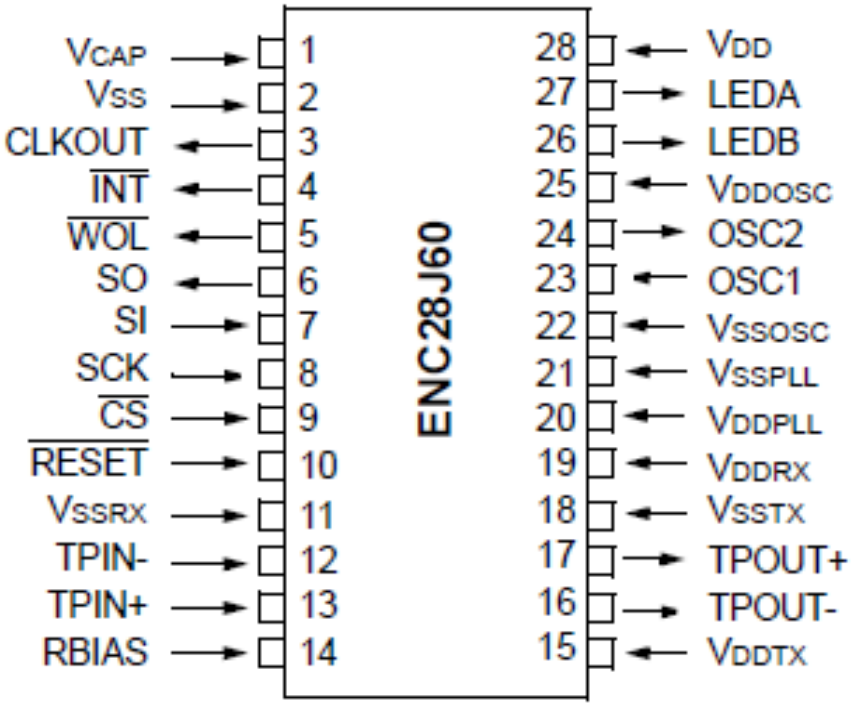
Given below is a brief description of the IC , the various functions available in enc8j60.h header file to control the ENC28J60 IC , the connections to be made between the ic and Atmega and the actual implementation of the circuit.

ABOUT ENC28J60

ENC28J60 is a standalone ethernet controller ic which is designed to serve as an Ethernet network interface for any controller equipped with SPI. It can be used to transmit and receive packets over ethernet. Its main features are:

- IEEE 802.3 compatible Ethernet controller
- Operating voltage of 3.1V to 3.6V (3.3V typical)
- 25 MHz clock input requirement
- Clock out pin with programmable prescaler
- Supports Full and Half-Duplex modes
- Two programmable LED outputs for LINK, TX, RX, collision and full/half-duplex status
- Programmable automatic retransmit on collision
- Programmable automatic rejection of erroneous packets
- Configurable transmit/receive buffer size
- SPI interface to communicate with host controller
- Supports Unicast, Multicast and Broadcast packets

PIN CONFIGURATION



PIN DESCRIPTIONS

Pin Name	Pin Number		Pin Type	Buffer Type	Description
	SPDIP, SOIC, SSOP	QFN			
VCAP	1	25	P	—	2.5V output from internal regulator. A low Equivalent Series Resistance (ESR) capacitor, with a typical value of 10 μF and a minimum value of 1 μF to ground, must be placed on this pin.
VSS	2	26	P	—	Ground reference.
CLKOUT	3	27	O	—	Programmable clock output pin.(1)
$\overline{INT}$	4	28	O	—	$\overline{INT}$ interrupt output pin.(2)
NC	5	1	O	—	Reserved function; always leave unconnected.
SO	6	2	O	—	Data out pin for SPI interface.(2)
SI	7	3	I	ST	Data in pin for SPI interface.(3)
SCK	8	4	I	ST	Clock in pin for SPI interface.(3)
$\overline{CS}$	9	5	I	ST	Chip select input pin for SPI interface.(3,4)
$\overline{RESET}$	10	6	I	ST	Active-low device Reset input.(3, 4)
VSSRX	11	7	P	—	Ground reference for PHY RX.
TPIN-	12	8	I	ANA	Differential signal input.
TPIN+	13	9	I	ANA	Differential signal input.
RBIAS	14	10	I	ANA	Bias current pin for PHY. Must be tied to ground via a resistor (refer to **Section 2.4 "Magnetics, Termination and Other External Components"** for details).
VDDTX	15	11	P	—	Positive supply for PHY TX.
TPOUT-	16	12	O	—	Differential signal output.
TPOUT+	17	13	O	—	Differential signal output.
VSSTX	18	14	P	—	Ground reference for PHY TX.
VDDRX	19	15	P	—	Positive 3.3V supply for PHY RX.
VDDPLL	20	16	P	—	Positive 3.3V supply for PHY PLL.
VSSPLL	21	17	P	—	Ground reference for PHY PLL.
VSSOSC	22	18	P	—	Ground reference for oscillator.
OSC1	23	19	I	ANA	Oscillator input.
OSC2	24	20	O	—	Oscillator output.
VDDOSC	25	21	P	—	Positive 3.3V supply for oscillator.
LEDB	26	22	O	—	LEDB driver pin.(5)
LEDA	27	23	O	—	LEDA driver pin.(5)
VDD	28	24	P	—	Positive 3.3V supply.

Legend: I = Input, O = Output, P = Power, DIG = Digital input, ANA = Analog signal input, ST = Schmitt Trigger

Note 1: Pins have a maximum current capacity of 8 mA.

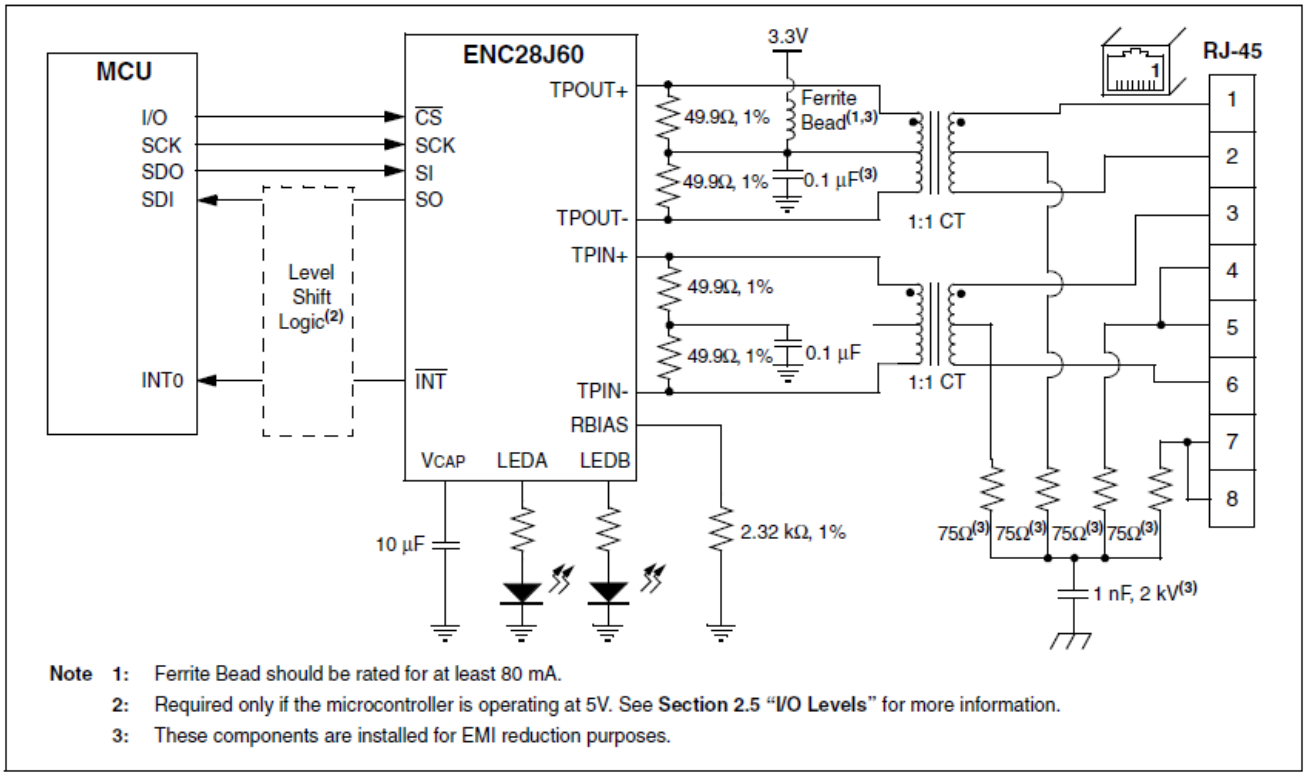
2: Pins have a maximum current capacity of 4 mA.

3: Pins are 5V tolerant.

4: Pins have an internal weak pull-up to VDD.

5: Pins have a maximum current capacity of 12 mA.

EXTERNAL CONNECTIONS



Note 1: Ferrite Bead should be rated for at least 80 mA.
2: Required only if the microcontroller is operating at 5V. See **Section 2.5 "I/O Levels"** for more information.
3: These components are installed for EMI reduction purposes.

1. **Oscillator**

Enc28j60 is designed to operate at 25MHz, so an external crystal must be placed between the OSC1 and OSC2 pins and grounded by 18pF Capacitors for noise removal. An external clock source of 25MHz, 3.3v can also be connected to OSC1 in which case OSC1 is left open.

2. **Magnetics**

In order for the IC to receive and send data over ethernet, On the differential receive pins (TPIN+/TPIN-), a 1:1 pulse transformer rated for 10BASE-T operation is required. And on the differential transmit pins (TPOUT+/TPOUT-), a 1:1 pulse transformer with a center tap is required before connecting it to RJ-45. Luckily, we have all these magnetics integrated in Magjack. So ,we will use Magjack in our project.

3. **I/O levels**

The IC is designed to work at 3.3V but it achieve SPI at 5V also. But to feed data to mcu we need to convert the 3.3v logic to 5v logic

4. **LEDs**

Two LEDs are connected to the IC which can be configured to shoe link status or receive/transmit activity. The polarity of LEDB is crucial as is decides the mode of transmission. If +ve is attached to the IC, it corresponds to half-duplex and vice versa for full-duplex.

MEMORY ORGANISATION

The memory of enc28j60 is divided in three parts namely; Control Registers , Ethernet Buffer and Phy Registers.

Control Registers

It is a part of memory which contains registers to control (by writing to it) and monitor (by reading values of the register) the working of the ic. The Control registers are directly read and written to by the SPI interface.

The control registor memory is dived into four banks. Each bank has 32 control registers addressed by a 5 bit value. The last five locations (1B to 1F) of all banks point to a common set of registers: EIE, EIR, ESTAT, ECON2 and ECON1. These are key registers used in controlling and monitoring the operation of the device. If you alter any of these registers from any of the bank the corresponding register of all the four banks get updated automatically. To access a particular register we need to first select its bank by changing the BSEL1:BSEL0 bits of ECON2 and then we can access it through its address.
Control registers for the ENC28J60 are generically grouped as ETH, MAC and MII registers.

TABLE 3-1: ENC28J60 CONTROL REGISTER MAP

Bank 0 Address	Name	Bank 1 Address	Name	Bank 2 Address	Name	Bank 3 Address	Name
00h	ERDPTL	00h	EHT0	00h	MACON1	00h	MAADR5
01h	ERDPTH	01h	EHT1	01h	Reserved	01h	MAADR6
02h	EWRPTL	02h	EHT2	02h	MACON3	02h	MAADR3
03h	EWRPTH	03h	EHT3	03h	MACON4	03h	MAADR4
04h	ETXSTL	04h	EHT4	04h	MABBIPG	04h	MAADR1
05h	ETXSTH	05h	EHT5	05h	—	05h	MAADR2
06h	ETXNDL	06h	EHT6	06h	MAIPGL	06h	EBSTSD
07h	ETXNDH	07h	EHT7	07h	MAIPGH	07h	EBSTCON
08h	ERXSTL	08h	EPMM0	08h	MACLCON1	08h	EBSTCSL
09h	ERXSTH	09h	EPMM1	09h	MACLCON2	09h	EBSTCSH
0Ah	ERXNDL	0Ah	EPMM2	0Ah	MAMXFLL	0Ah	MISTAT
0Bh	ERXNDH	0Bh	EPMM3	0Bh	MAMXFLH	0Bh	—
0Ch	ERXRDPTL	0Ch	EPMM4	0Ch	Reserved	0Ch	—
0Dh	ERXRDPTH	0Dh	EPMM5	0Dh	Reserved	0Dh	—
0Eh	ERXWRPTL	0Eh	EPMM6	0Eh	Reserved	0Eh	—
0Fh	ERXWRPTH	0Fh	EPMM7	0Fh	—	0Fh	—
10h	EDMASTL	10h	EPMCSL	10h	Reserved	10h	—
11h	EDMASTH	11h	EPMCSH	11h	Reserved	11h	—
12h	EDMANDL	12h	—	12h	MICMD	12h	EREVID
13h	EDMANDH	13h	—	13h	—	13h	—
14h	EDMADSTL	14h	EPMOL	14h	MIREGADR	14h	—
15h	EDMADSTH	15h	EPMOH	15h	Reserved	15h	ECOCON
16h	EDMACSL	16h	Reserved	16h	MIWRL	16h	Reserved
17h	EDMACSH	17h	Reserved	17h	MIWRH	17h	EFLOCON
18h	—	18h	ERXFCON	18h	MIRDL	18h	EPAUSL
19h	—	19h	EPKTCNT	19h	MIRDH	19h	EPAUSH
1Ah	Reserved	1Ah	Reserved	1Ah	Reserved	1Ah	Reserved
1Bh	EIE	1Bh	EIE	1Bh	EIE	1Bh	EIE
1Ch	EIR	1Ch	EIR	1Ch	EIR	1Ch	EIR
1Dh	ESTAT	1Dh	ESTAT	1Dh	ESTAT	1Dh	ESTAT
1Eh	ECON2	1Eh	ECON2	1Eh	ECON2	1Eh	ECON2
1Fh	ECON1	1Fh	ECON1	1Fh	ECON1	1Fh	ECON1

Ethernet Buffer

ENC28J60 has 8Kb buffer memory which is divided into separate transmit and receive buffer, the sizes and locations of which can be programmed by the user .

Receive buffer :

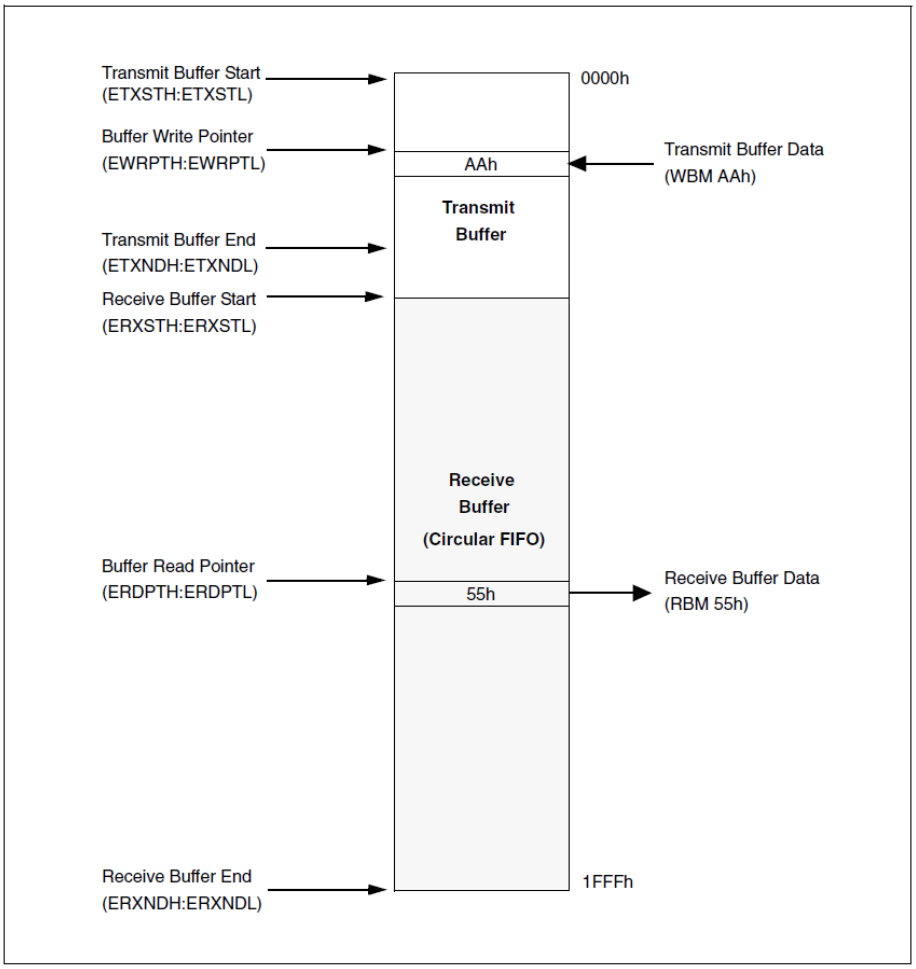
The data received through the ethernet are written sequentially in the receive buffer in FIFO(circular queue) pattern. The IC has pointers(registers) that point and the start and end of the receive buffer as well as pointers that points at the memory location upto which data has been received an read. If the receive buffer fills up with data and new data continues to arrive, the hardware will not overwrite the previously received data. Instead, the new data will be thrown away and the old data

will be preserved. So, in order to continuously receive new data, the host controller must periodically advance these pointers whenever it finishes processing some, or all, of the old received data.

Transmit buffer :

Any space other than FIFO receive buffer is used to temporarily store the data to be transmitted. When we write in the transmit buffer the IC doesn't check whether this region overlaps with the receive buffer memory, so care should be taken while writing.

FIGURE 3-2: ETHERNET BUFFER ORGANIZATION



PHY register

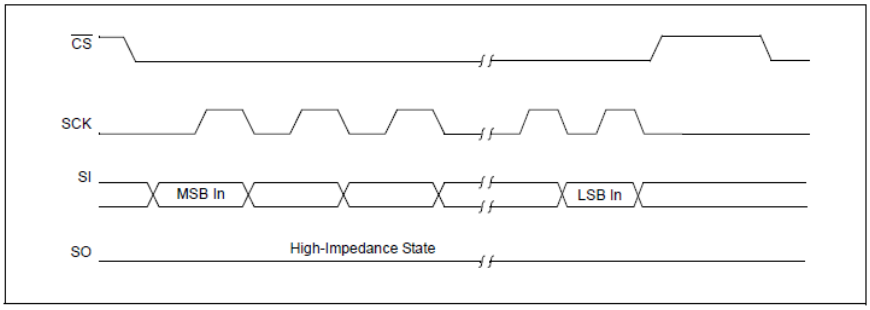
This register is used to configure the Phy module. The PHY (Physical Layer) module encodes and decodes the analog data that is present on the twisted pair interface.All PHY registers are 16 bits in width. There are a total of 32 PHY addresses; however, only 9 locations are implemented.Unlike the ETH, MAC and MII control registers, or the buffer memory, the PHY registers are not directly accessible through the SPI control interface. Instead, access is accomplished through a special set of MII control registers.

SPI INTERFACING

As said earlier, the Ic interfaces with the host controller through SPI. The IC supports SPI (0,0) mode only and that SCK must be low in idle state. During the SPI communication Csbar must be held low.

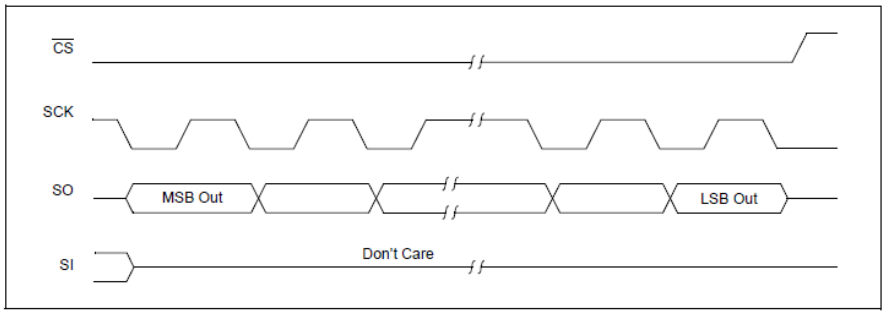
Data is read at each rising edge of enc28j60.

FIGURE 4-1: SPI™ INPUT TIMING



Data is transmitted at falling edges by enc28j60.

FIGURE 4-2: SPI™ OUTPUT TIMING



The instructions are given to the IC by sending special codes called Opcodes commands the IC to perform instructions such as reading or writing of control registers or buffer memory.

There are SEVEN Opcodes :

Instruction	Opcode	Argument	Data
Read Control Register	**000**	**aaaaa**	**N/A**
Read Buffer Memory	**001**	**11010**	**N/A**
Write Control Register	**010**	**aaaaa**	**dddddddd**
Write Buffer Memory	**011**	**11010**	**dddddddd**
Bit Field Set	**100**	**aaaaa**	**dddddddd**
Bit Field Clear	**101**	**aaaaa**	**dddddddd**
System Reset Command (soft reset)	**111**	**11111**	**N/A**

a=address d=data

1. ## Read Control Register (RCR) Instruction

   The mcu sends the opp code(ooo) followed by the 5 bit address of the register to be read. In return during next cycle of SPI communication ,the value of the register is send by the IC. If the register is MAC or MII then first a dummy byte is send and then in the next cycle the data

is send.

FIGURE 4-3: READ CONTROL REGISTER COMMAND SEQUENCE (ETH REGISTERS)

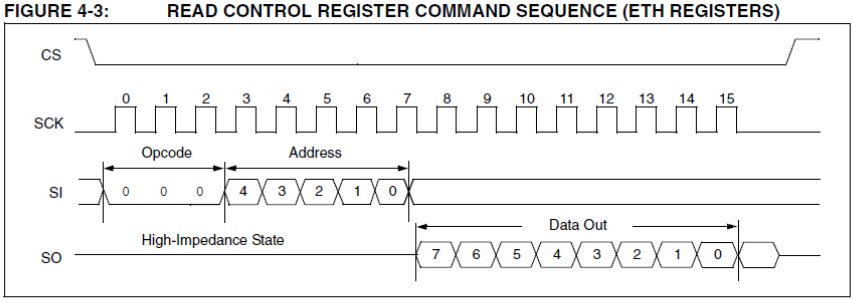
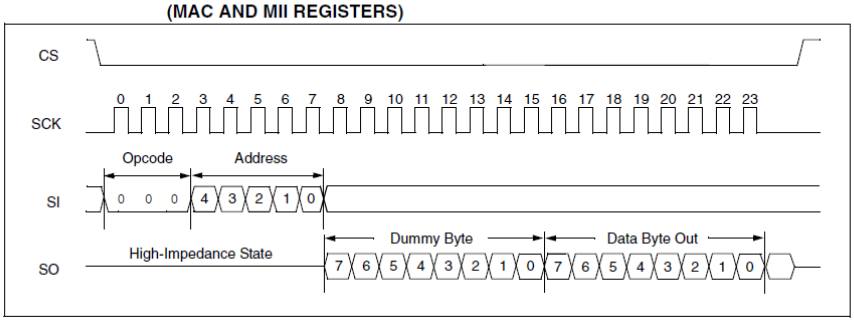


FIGURE 4-4: READ CONTROL REGISTER COMMAND SEQUENCE (MAC AND MII REGISTERS)



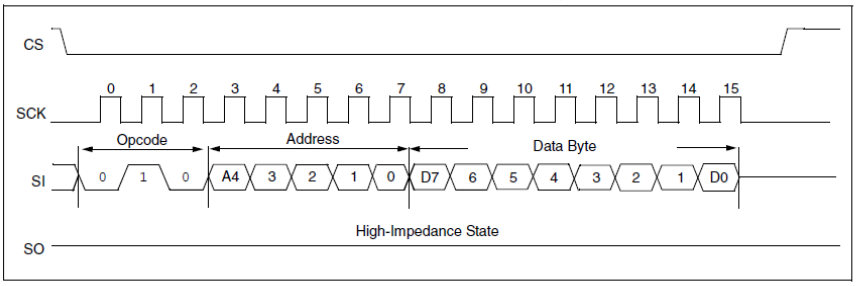
2. Read Buffer Memory (RBM) Instruction

The mcu sends the opcode(001) followed by 1A(11010). The IC shifts out data from the receive buffer memory as long as CSbar is kept low(and of course, master sends some dummy data). The receive buffer is a circular queue, so whenever the last byte is read, wrapping occurs.

3. Write Control Register (RCR) Instruction

The mcu sends the opcode(010) followed by the address of the control register to be written to. The next byte to be send corresponds to the data to be written.

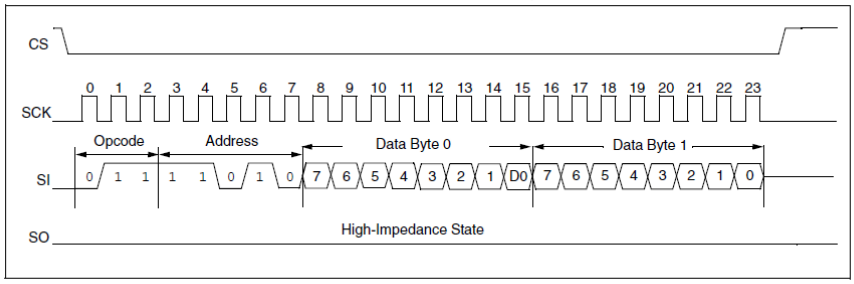
FIGURE 4-5: WRITE CONTROL REGISTER COMMAND SEQUENCE



4. Write Buffer Memory (WBM) Instruction

The mcu sends opcode(011) followed by 1A. Then the data o be written in the buffer memory is send byte-per-byte , keeping CSbar low.

FIGURE 4-6: WRITE BUFFER MEMORY COMMAND SEQUENCE



5. Bit Field Clear (BFT) Instruction

It is used to clear any or all of the 8 bits of a control register(only ETH , not MAC or MII). The opcode(100) is send followed by register's address. Then the bits to be cleared. (Eg: if bit 0 and 6 are to be cleared we will send 01000001)

6. Bit Field Set (BFS) Instruction

It is used to set any or all of the 8 bits of a control register(only ETH , not MAC or MII). The opcode(100) is send followed by register's address. Then the bits to be cleared. (Eg: if bit 1 and 5 are to be cleared we will send 00100010)

FUNCTION DESCRIPTION

By including the header file enc28j60.h in our project file, we can use various functions to control the various registers of the IC and to read and write from the buffer memory of the IC. Given below is a brief description of the useful functions. unint_8 in the examples below is nothing but an 8-bit number ,i.e., unsigned char.

a. **uint8_t enc28j60Read(uint8_t address);**

address is the address of the control register to be read

Note: The address of control registers is actually of 5 bits but for simplicity we covert it into a 8 bit address out of which B7 bit is 1 if the register is MAC or MII and 0 if it is ETH.The bits B6 – B5 is the bank of the control register .The last 5 bits are address of control register in that bank.

- It returns the value of the control register
- It sets the bank by its own.

b. **void enc28j60ReadBuffer(uint16_t len,uint8_t *data);**

- It stores len bytes of data in the memory pointed by the data pointer.
- It also converts it into a string by placing '\0' at the end.

c. **void enc28j60Write(uint8_t address,uint8_t data);**

- It writes 8 bit data to the control register.

d. **void enc28j60WriteBuffer(uint16_t len,uint8_t *data);**

- It writes len bytes of data to the transmit buffer.

e. **void enc28j60PhyWrite(uint8_t address,uint16_t data);**

- Writes 16 bit data to the PHY register whose address is given.

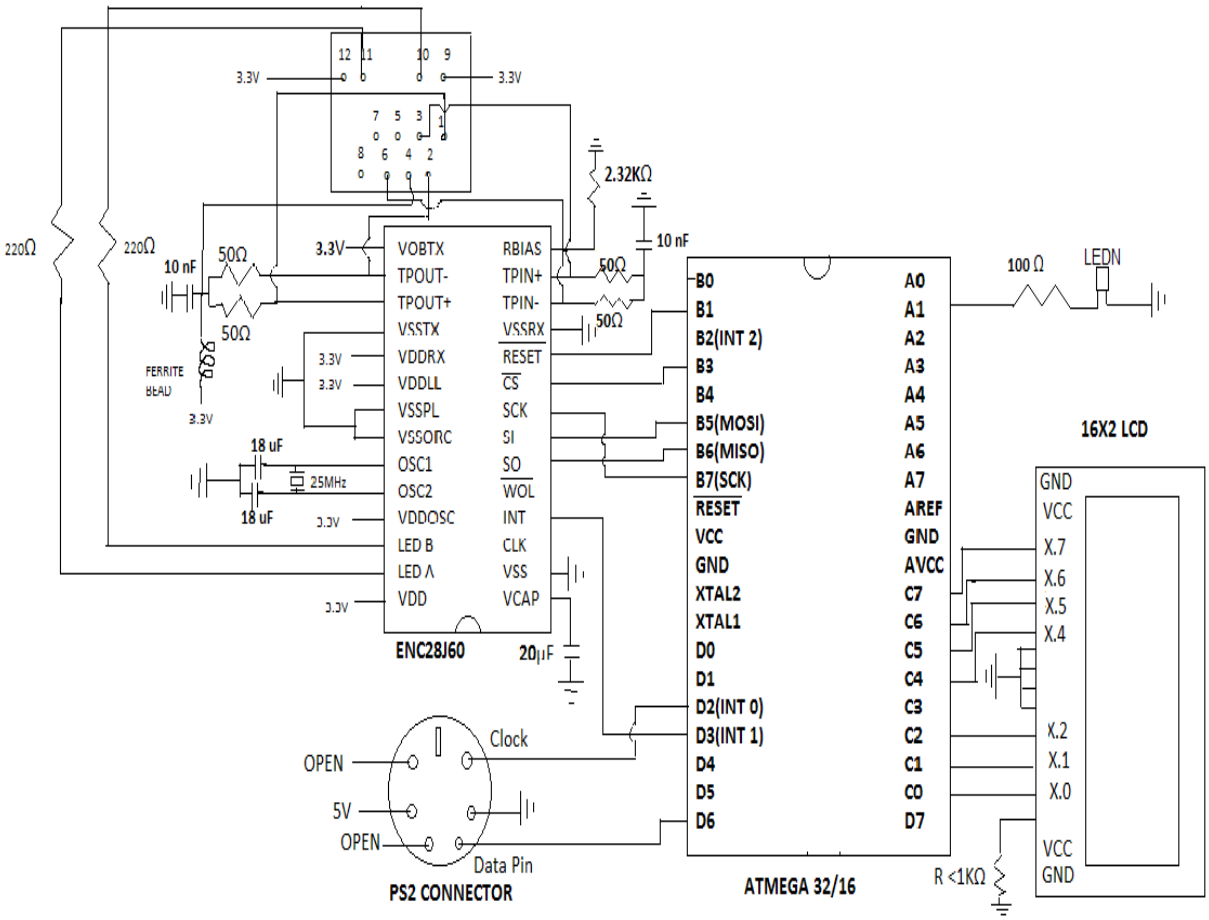
f. **void enc28j60WriteOp(ENC28J60_BIT_FIELD_CLR,uint8_t address,uint8_t data);**

- **data** field will be : if bit 0 and 6 are to be cleared we will send 01000001

g. **void enc28j60WriteOp(ENC28J60_BIT_FIELD_SET, uint8_t address,uint8_t data);**

- **data** field will be : if bit 1 and 7 are to be cleared we will send 10000010

CIRCUIT DIAGRAM



IMPLEMENTATION

INITIALISATION

Before the Enc28j60 IC can be used, certain Control and Phy registers need to be properly initialised. This is done by calling the function

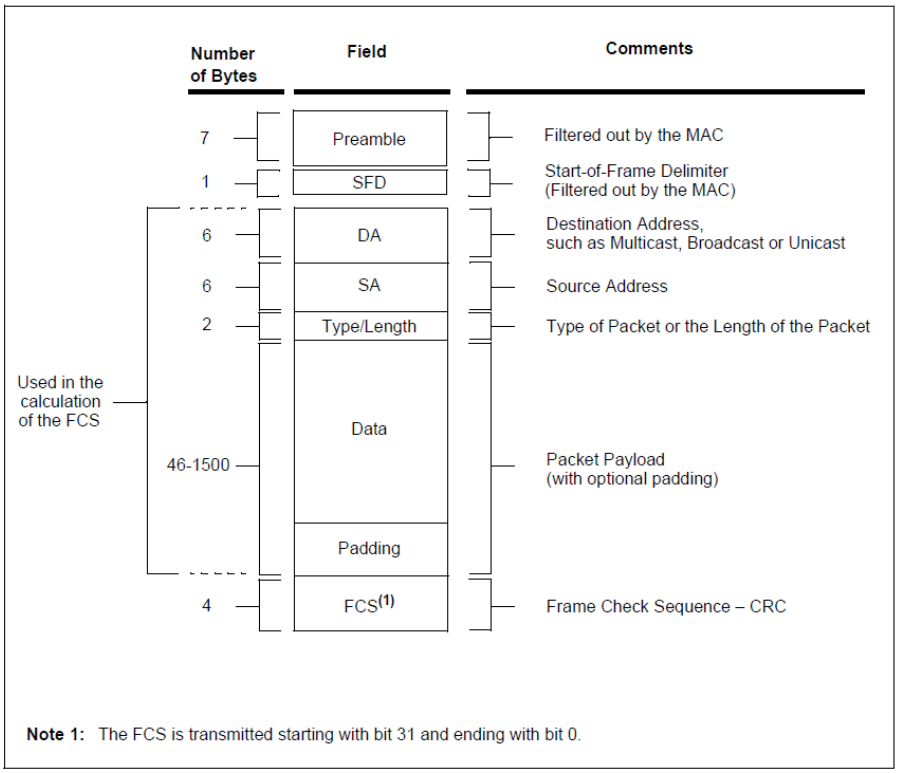
```
void enc28j60Init(char *mac);
```

in the main before using any of enc functions. Here,mac is an array which contains the mac address of the IC.(for the list of registers that need to be initialised,refer to datasheet, page no 33)

TRANSMITING PACKETS

To transmit a packet first the data to be transmitted is written in the buffer memory in the appropriate format using write buffer memory command. Then the ECON1.TXRTS bit is set and the data is send over the LAN.

FIGURE 5-1: ETHERNET PACKET FORMAT



This is the basic layout of an Ethernet data packet .Out of all these fields the destination address ,source address ,type of data the data should be wriiten in an array.The size of the array and pointer to the array should be passed to the packet send function. All the remaining fields will be filled by enc if configured to do so.

void enc28j60 PacketSend (int len, char* data);

If we need to receive interrupts when the transmission is over we need clear EIR.TXIF, set EIE.TXIE and set EIE.INTIE And this can simply be done by writing the following lines in main() before while(1)

```
enc28j60WriteOp(ENC28J60_BIT_FIELD_CLR,EIR,EIR_TXIF);
enc28j60WriteOp(ENC28J60_BIT_FIELD_SET,EIE,EIE_TXIE);
enc28j60WriteOp(ENC28J60_BIT_FIELD_SET,EIE,EIE_INTIE);
```

Connect the INTbar pin of enc to the external interrupt pin of the controller IC and write the following commands in the interrupt function

```
enc28j60WriteOp(ENC28J60_BIT_FIELD_CLR,EIE,EIE_INTIE);
enc28j60WriteOp(ENC28J60_BIT_FIELD_CLR,EIR,EIR_TXIF);
enc28j60WriteOp(ENC28J60_BIT_FIELD_SET,EIE,EIE_INTIE);
```

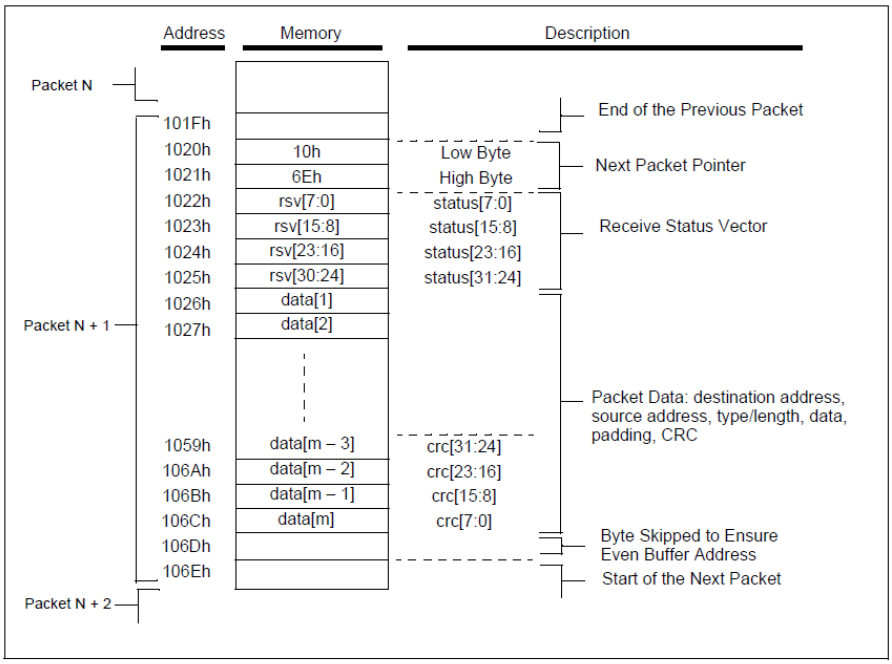
RECEIVING PACKETS

The settings for receiving the data packet has already been done in the initialization function.So by just calling the function void enc28j60Init(uint8_t* macaddr); all the settings will be done.

The receive buffer is also a circular buffer.So if data is continuously received but not read then the data will be lost.

Whenever a packet is received an interrupt will be generated .We can read the data in the interrupt and display it.

FIGURE 7-3: SAMPLE RECEIVE PACKET LAYOUT



READING DATA

As shown in the receive packet format the first two bytes indicate the pointer to the next packet.The next 4 bytes is a receive vector .The next two bytes (included in receive vector) after next packet pointer indicate the length of the packet.The next two bytes of receive vector store various other informations.Then there is destination address, source address, length, data , padding, CRC and then 1 byte is skipped.

The data can easily be read by calling read packet function.

```
uint16_t enc28j60PacketReceive(uint16_t maxlen, uint8_t* packet);
```

where **maxlen** is the maximum length of data user wants to read.If the data packet is larger than the maxlen then the maxlen bytes of data else the complete data will be written to the memory location pointed by the packet pointer.This function also frees the buffer for further usage.

This function returns the length of the data so that user can determine how long was the received data packet.

The received data packet is of the format given below. The format of the data packet depends upon the type of protocol used. The required fields can be read from the buffer memory and the data can be interpreted.

An overview of the first 56 bytes of an ethernet-packet:

	ARP	IP	ICMP	TCP	UDP
1	Destination MAC-addr. 1	Destination MAC-addr. 1	Destination MAC-addr. 1	Destination MAC-addr. 1	Destination MAC-addr. 1
2	Destination MAC-addr. 2	Destination MAC-addr. 2	Destination MAC-addr. 2	Destination MAC-addr. 2	Destination MAC-addr. 2
3	Destination MAC-addr. 3	Destination MAC-addr. 3	Destination MAC-addr. 3	Destination MAC-addr. 3	Destination MAC-addr. 3
4	Destination MAC-addr. 4	Destination MAC-addr. 4	Destination MAC-addr. 4	Destination MAC-addr. 4	Destination MAC-addr. 4
5	Destination MAC-addr. 5	Destination MAC-addr. 5	Destination MAC-addr. 5	Destination MAC-addr. 5	Destination MAC-addr. 5
6	Destination MAC-addr. 6	Destination MAC-addr. 6	Destination MAC-addr. 6	Destination MAC-addr. 6	Destination MAC-addr. 6
7	Source MAC-address 1	Source MAC-address 1	Source MAC-address 1	Source MAC-address 1	Source MAC-address 1
8	Source MAC-address 2	Source MAC-address 2	Source MAC-address 2	Source MAC-address 2	Source MAC-address 2
9	Source MAC-address 3	Source MAC-address 3	Source MAC-address 3	Source MAC-address 3	Source MAC-address 3
10	Source MAC-address 4	Source MAC-address 4	Source MAC-address 4	Source MAC-address 4	Source MAC-address 4
11	Source MAC-address 5	Source MAC-address 5	Source MAC-address 5	Source MAC-address 5	Source MAC-address 5

12	Source MAC-address 6	Source MAC-address 6	Source MAC-address 6	Source MAC-address 6	Source MAC-address 6
13	Packet type 0	Packet type 0	Packet type 0	Packet type 0	Packet type 0
14	Packet type 1	Packet type 1	Packet type 1	Packet type 1	Packet type 1
15	hwtype 0	version and length	version and length	version and length	version and length
16	hwtype 1	tos	tos	tos	tos
17	prtype 0	packet-length h	packet-length h	packet-length h	packet-length h
18	prtype 1	packet-length l	packet-length l	packet-length l	packet-length l
19	hwlen	id 0	id 0	id 0	id 0
20	prlen	id 1	id 1	id 1	id 1
21	op 0	frag-offset 0	frag-offset 0	frag-offset 0	frag-offset 0
22	op 1	frag-offset 1	frag-offset 1	frag-offset 1	frag-offset 1
23	shaddr	Time to live	Time to live	Time to live	Time to live
24	sipaddr	Protocol	Protocol	Protocol	Protocol
25	thaddr	Header checksum h	Header checksum h	Header checksum h	Header checksum h
26	tipaddr	Header checksum l	Header checksum l	Header checksum l	Header checksum l
27		Source address 0	Source address 0	Source address 0	Source address 0
28		Source address 1	Source address 1	Source address 1	Source address 1
29		Source address 2	Source address 2	Source address 2	Source address 2
30		Source address 3	Source address 3	Source address 3	Source address 3
31		Destination address 0	Destination address 0	Destination address 0	Destination address 0
32		Destination address 1	Destination address 1	Destination address 1	Destination address 1
33		Destination address 2	Destination address 2	Destination address 2	Destination address 2
34		Destination address 3	Destination address 3	Destination address 3	Destination address 3
35		IP-data	Type	Source port h	Source port h
36			Code	Source port l	Source port l
37			Checksum h	Destination port h	Destination port h
38			Checksum l	Destination port l	Destination port l
39			Id h	Sequence-number 3	Length h
40			Id l	Sequence number 2	Length l
41			Sequence	Sequence	Checksum h

			number h	number 1	
42			Sequence number l	Sequence number 0	Checksum l
43			ICMP-data	Acknowledge number 3	UDP-data
44			ICMP-data	Acknowledge number 2	UDP-data
45			ICMP-data	Acknowledge number 1	UDP-data
46			ICMP-data	Acknowledge number 0	UDP-data
47			ICMP-data	Header	UDP-data
48			ICMP-data	Flags	UDP-data
49			ICMP-data	Window h	UDP-data
50			ICMP-data	Window l	UDP-data
51			ICMP-data	Checksum h	UDP-data
52			ICMP-data	Checksum l	UDP-data
53			ICMP-data	Urgent-pointer h	UDP-data
54			ICMP-data	Urgent-pointer l	UDP-data
55			ICMP-data	TCP-data	UDP-data
56			etc. etc.	etc. etc.	etc. etc

CONCLUSION

- The device can chat with any computer connected to ethernet within the IITK campus (LAN) if we know the local IP address of the computer
- The module can also easily chat with another similar module connected to LAN.

FUTURE SCOPE

The Standalone Chat Client has tremendous scope for use in future.

It may be used to implement:

- Home Surveillance System : We can connect various devices to atmega and then can sends instructions over lan to control these devices

- Mini Server : The IC can also act as a mini server, where we can log in from our computer and view the website data stored in the IC .

ACKNOWLEDGEMENT

We would like to thank E-club for providing us this opportunity and resources to think and implement this idea. Special thanks to our mentors Ganesh Pitchiach and Rishabh Maheswari for guiding us and motivating us throughout this project. It was a memorable,fun and useful experience..