MICROCONTROLLER BASED ETHERNET EMBEDDED SYSTEMS

R. DEMETER[1] R. CÂMPEANU[1]

Abstract: *Ethernet has traditionally been a quite complex interface. Until today, the Ethernet chips were difficult to use with a small microcontroller with little memory. The new ENC28J60 Ethernet chip, which is a small chip with only 28 pins, was developed by Microchip to be used as an Ethernet network interface for any microcontroller equipped with SPI. So, Microchip opens a whole world of completely new applications. It is easy to build small devices which can be spread all over the house and are simply connected via Ethernet, so there is no need anymore for a separate serial connection or other bus. Everything can be easily connected via Ethernet. Distance is no longer a limiting factor. Even WIFI connectivity is possible because the devices can be connected to a wireless bridge.*

Key words: *embedded system, microcontroller, SPI programming.*

1. Introduction

In this paper we will build a hardware having an 8 bit AVR microcontroller with lots of IO interfaces, analog to digital converter inputs and Ethernet interface.

The main purpose is to show here the schematic block diagram and explain the software application. For testing we use a UDP application to communicate with the ATmega168 microcontroller (MCU). This application receives the measured temperature, humidity and calculated dew point from the embedded system.

All Ethernet chips until today had 100 pins or more (for example RTL8019, RTL8139 or SMSC LAN91C111), were difficult to find in small quantities and difficult to use with a small microcontroller with little memory [2].

As shown in Table 1, most of the Ethernet controllers have ISA, PCI or SNI interfaces, and cannot connect directly to a general purpose microcontroller.

Comparison of Ethernet chips Table 1

Ethernet chip	Number of pins	Speed [mbps]	BUS
ENC28J60	28	10	SPI
RTL8201	48	10/100	SNI
RTL8019	100	10	ISA
RTL8139	100	10/100	PCI
LAN91C111	128	10/100	ISA

2. The ENC28J60 Ethernet Controller

Microchip's ENC28J60 controller is a 28-pin, 10BASE-T standalone Ethernet Controller, with on board MAC & PHY, 8 Kbytes of Buffer RAM and an SPI (Serial Peripheral Interface) serial interface used as an Ethernet network interface for any microcontroller equipped with SPI interface. Microchip offers also a free licensed TCP/IP stack optimized for the PIC18, PIC24, dsPIC and PIC32 microcontroller families. The stack is divided into multiple layers, where each layer accesses services from one or more layers directly below it

[1] Dept. of Automatics, *Transilvania* University of Braşov.

and includes the following key features:

- Supported protocols: ARP, IP, ICMP, UDP, TCP, DHCP, SNMP, HTTP, FTP, TFTP;
- Socket support for TCP and UDP;
- Secure Sockets Layer (SSL);
- NetBIOS Name Service;
- DNS - Domain Name System;
- Support for MPLAB C18, C30, and C32 compilers.

The ENC28J60 meets all of the IEEE 802.3 specifications. It provides an internal DMA module for fast data throughput and hardware assisted checksum calculation, which is used in network protocols [5].

The ENC28J60 consists of seven important functional blocks:

- An SPI interface (for the communication channel between the host controller and the ENC28J60);
- Control registers (to control and monitor the ENC28J60);
- A dual port RAM buffer (to receive and transmit data packets);
- An arbiter to control the access to the RAM buffer when requests are made from DMA, transmit and receive data blocks;
- The bus interface (interprets data and commands received via SPI interface);
- The Medium Access Control (MAC) module that implements IEEE 802.3 compliant MAC logic;
- The Physical Layer (PHY) module which encodes and decodes the analogue data that is present on the interface.

The ENC28J60 also contains: on-chip voltage regulator, oscillator, level translators to provide 5 V tolerant I/Os and system control logic.

The ENC28J60 controller from Microchip is a very useful chip. It has Tx/Rx buffer, MAC and PHY in one small chip, as in the Figure 1 [5]. There are very few external parts: like an Ethernet transformer. All this comes in a convenient 28-pin DIP package and it is easy to solder and it is perfect for hobby applications. The MCU can then control remotely any hardware, like: sensors (light, temperature), switch on an off something, LCD display etc.

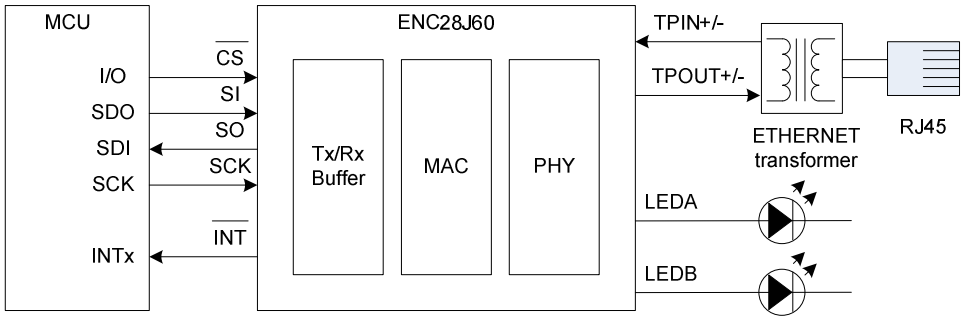

Fig. 1. *Schematic block diagram*

As presented above, the ENC28J60 uses SPI interface for communications. SPI requires 4 signals for bidirectional communication:

1. Clock (SCK);
2. Data In (SI);
3. Data Out (SO);
4. Chip Select (CS).

The clock signal is controlled by the master device i.e. the ATmega168. All data is clocked in and out using this pin. These lines need to be connected to the relevant pins on the ATmega168. Any unused GIO pin can be used for CS, instead pull this pin high.

In Table 2 is an overview of SPI instruction set implemented for the ENC28J60 [5]:

SPI instruction set Table 2

Instruction	Opcode
Read Control Register (RCR)	000
Read Buffer Memory (RBM)	001
Write Control Register (WCR)	010
Write Buffer Memory (WBM)	011
Bit Field Set (BFS)	100
Bit Field Clear (BFC)	101
Soft Reset	111

2.1. Initialization

During initialization the following operations are executed: setting the buffer memory and registers, putting the MAC-address on the right location, setting PHY register to half/full-duplex communication, setting the LED configuration, and enabling automatic padding, enabling CRC operations and the interrupts.

The Ethernet buffer contains transmit and receive memory used by the Ethernet controller. The entire buffer is 8 Kbytes, divided into separate receive and transmit buffer spaces. The sizes and locations of transmit and receive memory are fully programmable by the host controller using the SPI interface. Any space within the 8-Kbyte memory, which is not programmed as part of the receive FIFO buffer, is considered to be the transmit buffer.

2.2. Transmitting Packets

A MAC packet consists of a 7 byte preamble, 1 byte start of frame delimiter, a 6 byte MAC source address, 6 byte MAC destination address, 2 byte type/length packet, 46-1500 data bytes and a 4 byte checksum.

The MAC module inside the ENC28J60 will automatically generate the preamble and Start-Of-Frame delimiter fields when transmitting. Additionally, the MAC can generate any padding (if needed) and the CRC if configured to do so. The host controller must generate and write all other frame fields into the buffer memory for transmission.

Additionally, the ENC28J60 requires a single per packet control byte to precede the packet for transmission. Before transmitting packets, the MAC registers which alter the transmission characteristics should be initialized and then, the host can generate all the other sections and place them in the correct place within the ENC28J60's memory map.

To transmit a packet, the host controller should:

1. Program the ETXST Pointer to point to an unused location in memory. It will point to the per packet control byte
2. Use the WBM SPI command (Table 1) to write a per packet control byte, the destination address, the source MAC address, the type/length and the data payload.
3. Program the ETXND Pointer. It should point to the last byte in the data payload.
4. Clear EIR.TXIF, set EIE.TXIE and EIE.INTIE to enable an interrupt.
5. Start the transmission process by setting ECON1.TXRTS.

2.3. Receiving Packets

Assuming that the receive buffer has been initialized, the MAC has been properly configured and the receive filters have been configured to receive Ethernet packets; the host controller should enable reception by setting ECON1.RXEN.

After reception is enabled, packets which are not filtered out will be written into the circular receive buffer. Any packet which does not meet the necessary filter criteria will be discarded and the host controller will not have any means of identifying that a packet was thrown away. When a packet is accepted it is completely written into the buffer.

To process the packet, the host controller will normally use the RBM (Read Buffer Memory) SPI command (Table 2) and start reading from the beginning of the next Packet Pointer. The host controller will save the next Packet Pointer, any necessary bytes from the receive status vector and then proceed to read the packet contents.

To minimize the processing requirements of the host controller, the ENC28J60 incorporates several different receive filters which can automatically reject packets which are not needed. Six different types of packet filters are implemented: unicast, pattern match, magic packet, hash table, multicast and broadcast.

The individual filters are all configured by the ERXFCON register. More than one filter can be active at any given time. Additionally, the filters can be configured by the ANDOR bit to either logically AND, or logically OR, the tests of several filters. In other words, the filters may be set so that only packets accepted by all active filters are accepted, or a packet accepted by any one filter is accepted.

The device can enter in promiscuous mode and receive all packets by clearing the ERXFCON register. The proper setting of the register will depend on the application requirements.

A driver for the ENC28J60 will contain the following accessible functions:

1. Initialize the ENC28J60 controller;
2. Write a packet on to the Ethernet;
3. Try to Read a packet from the MAC.

The driver functions have the following prototypes:

```
// initialize the Ethernet interface for transmit/receive
void enc28j60Init(uint8_t* macaddr);

// do a ENC28J60 read operation
uint8_t enc28j60ReadOp(uint8_t op, uint8_t address);

// do a ENC28J60 write operation
void enc28j60WriteOp(uint8_t op, uint8_t address, uint8_t data);

// copy the packet from the receive buffer
void enc28j60ReadBuffer(uint16_t len, uint8_t* data);

// copy the packet into the transmit buffer
void enc28j60WriteBuffer(uint16_t len, uint8_t* data);

// copy the packet into the transmit buffer.
// send the contents of the transmit buffer onto the network
void enc28j60PacketSend(uint16_t len, uint8_t* packet);

// gets a packet from the network receive buffer, if one is available.
// the packet will be headed by an ethernet header.
// returns the packet length in bytes if a packet was retrieved, zero otherwise.
uint16_t enc28j60PacketReceive(uint16_t maxlen, uint8_t* packet);
```

The ENC28J60 driver should be portable across a number of platforms, so ideally any platform specific code should be kept out of the driver module. In this case the platform specific details will be in the SPI.

3. Ethernet Based Temperature, Humidity and Dew Point Monitoring

The monitoring system is built using a few embedded modules, described above. Each embedded module is equipped with

an ATmega168 [1], [4] and an ENC28J60 Ethernet controller.

Instead of the ATmega168 any MCU that has at least 16 KB of non-volatile memory and an SPI interface can be used from the AVR or PIC families. The AVR and PIC MCUs have the same performances and similar pricing. Also, for picking the MCU we have to consider the programming language and the SDK. For the AVR family there is *avr-gcc*, a targeted version of *gcc*, which is completely free and works on Windows, Linux and MacOS. The C language used for AVR is standard C, with the standard libraries, includes, linking, unions, structs and pointers, which means that porting code is really easy because everyone uses it. All of PIC compilers aren't 100% compatible and the SDK, MPLAB IDE, student version has only level 2 optimizations.

So, we chose ATmega168 from AVR family to show that ENC28J60 driver is platform independent and can be used with another microcontroller instead of PIC family MCUs.

To upload the firmware in ATmega168 MCU *PonyProg* [3] was used, which is serial device programmer software, available for Windows and Linux operating systems. *SI-Prog* programmer hardware interface [3], is a board that offers support for most of the PIC and AVR MCUs. Using *PonyProg* and *SI-Prog* Wafercard for SAT, eeprom within GSM, TV or CAR-RADIO can be programmed. Furthermore it can be used as a low cost starter kit for PIC and AVR microcontrollers.

The monitoring system, besides the ATmega168 and Ethernet controller, also has a subsystem with SHT11 sensors [6], for measuring temperature and humidity connected to the MCU using the I2C serial bus. SHT11 sensors are powered by Sensirion Inc., which has ±3.0 humidity accuracy and ±0.4 temperature accuracy. For the I2C communication any one of the MCU's digital IO ports can be used.

To initiate a transmission, a "Transmission Start" sequence has to be issued. The subsequent command consists of tree address bits and five command bits.

SHT11 list of commands Table 3

Instruction	**Opcode**
Reserved	0000x
Measure Temperature	**00011**
Measure Humidity	**00101**
Read Status Register	00111
Write Status Register	00110
Reserved	0101x-1110x
Soft reset, resets the interface, clears the status register to default values and wait 11ms	**11110**

After issuing the measurement command (Table 3) (00000101 for RH and 00000011 for Temperature) the MCU has to wait for measurement to complete (data ready). After that, two bytes of measurement data and one byte of CRC-8 checksum will then be received.

With measured temperature T of air and relative humidity RH, the dew point T_d can be calculated, using a well-known approximation:

$$T_d = \frac{b\varphi(T,RH)}{a-\varphi(T,RH)}, \tag{1}$$

where:

$$\varphi(T,RH) = \frac{aT}{b+T} + \ln\frac{RH}{100}, \tag{2}$$

and the temperatures are in degrees Celsius.

The a and b constants are:

a = 17.271,

b = 237.7 °C.

This expression (1) is based on the August-Roche-Magnus approximation for the saturation vapor pressure of water in air as a function of temperature.

From the entire network traffic we only need the UDP traffic with the destination of our MAC address. On top of that we have to process ping (IP/ICMP packets for our MAC address) and ARP packets (*content-type=ARP*) to our MAC or to broadcast, because no IP networking is possible without ARP. All other packets, especially broadcast IP packets can be ignored.

Each embedded system transmits at one minute intervals a MAC broadcast frame, packaged with the module's identifier and with the measured values of temperature, humidity and calculated value of dew point. The transmitted data by embedded systems are received by a monitoring system and saved into a database.

4. Conclusion

The temperature, humidity and dew point monitoring application is just a simple example to show the possibilities of this method. It can be used for any kinds of applications (including monitoring and control) where different sensors and devices need to be interconnected even over large distances.

With a small foot print package size, the ENC28J60 minimizes complexity, board space and cost. Target applications with Ethernet chip include Voice over IP, Industrial Automation, Building Automation, Home Control, Security and Instrumentation.

References

1. Dhananjay, V.G.: *Programming and Customizing the AVR Microcontroller*, McGraw-Hill, 2001.
2. Jan, Axelson: *Embedded ETHERNET and Internet complete*. Lakeview Research LLC, 2003.
3. Lanconelli Open Systems (LancOS). Available at: *http://www.lancos.com/prog.html.* Accessed: 16-06-2009.
4. *** *Atmel ATmega168 AVR Microcontroller Data Sheet*. Atmel Co, 2004.
5. *** *ENC28J60 Data Sheet - Stand-Alone Ethernet Controller with SPI Interface*. Microchip Technology Inc., 2008.
6. *** *SHT11 Humidity and Temperature Sensor Data Sheet*. Sensirion Inc., 2007.