Utveckling och analys av en PIC-mikrokontrollers möjligheter att kommunicera via TCP/IP

Development and analysis of a PIC-microcontrollers possibilities to communicate over TCP/IP

Daniel Snel, Stefan Mattsson

EXAMENSARBETE
Elektroteknik
Nr: E 3444 E

2006

EXAMENSARBETE, C-nivå Elektroteknik

DALARNA
University College

Program	Reg nr	Omfattning
Elektroteknik, 120 p	E 3444 E	10 p
Namn Daniel Snel Stefan Mattsson	Datum 2006-05-21	
Handledare Mats Isberg	Examinator Examinators namn	
Företag/Institution Sterners Specialfabrik AB	Kontaktperson vid företaget/institutionen Håkan Sterner	
Titel Utveckling och analys av en PIC-mikrokontrollers möjligheter att kommunicera via TCP/IP		
Nyckelord PIC 18F8722, TCP/IP, ENC28J60		

Sammanfattning

Sterners Specialfabrik AB tillverkar bland annat mynt- och biljettautomater. För att underlätta felsökning, underhåll, avläsning av statistik med mera på en automat är det lämpligt att detta utförs på en PC över Internet. Det innebär att automaten inte behöver besökas då dessa uppgifter skall utföras. Examensarbetet går ut på att få en PIC mikrokontroller att kommunicera med en användare vid en PC över internet. För detta krävs att en TCP/IP stack implementeras på PIC-mikrokontrollern.

Arbetet ledde till en fungerande demoapplikation som klarar av att kommunicera över TCP/IP. Demoapplikationen innehåller diverse olika funktioner för att demonstrera hur en automat skulle kunna styras och övervakas.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

DALARNA
University College

DEGREE PROJECT
Electrical Engineering

Programme		Reg number	Extent
Electrical Engineering		E 3444 E	15 ECTS
Name of student		Year-Month-Day	
Daniel Snel	Stefan Mattsson	2006-05-21	
Supervisor		Examiner	
Mats Isberg		Examiners name	
Company/Department		Supervisor at the Company/Department	
Sterners Specialfabrik AB		Håkan Sterner	
Title			
Development and analysis of a PIC- microcontrollers possibilities to communicate over TCP/IP			
Keywords			
PIC 18F8722, TCP/IP, ENC28J60			

Summary

Sterners Specialfabrik AB manufactures among other things coin- and ticket vending machines. To facilitate fault-detecting, maintenance, follow-up statistics etc. on a machine it would be convenient if this is done on a PC over the Internet. This means that there is no need to visit the machine to perform these tasks. The objective of the degree project is to get a PIC-microcontroller to communicate with a user by a PC over the Internet. To accomplish this a TCP/IP stack needs to be implemented on the PIC- microcontroller.

The project resulted in a working demoapplication which can communicate over TCP/IP. The demoapplication contains different functions to demonstrate how a vending machine could be controlled and supervised.

Högskolan Dalarna
Röda vägen 3
781 88 BORLÄNGE

Telefon: 023-77 80 00
Telefax: 023-77 80 50
URL: http://www.du.se/

1. Förord

Genom detta examensarbete har vi fått lära oss mycket som vi inte var särskilt insatta i sedan förut. Vi hade aldrig tidigare jobbat med en PIC och mycket lite med C-programmering, TCP/IP och HTML och vi är tacksamma för att ha fått chansen att arbeta med detta.

Vi har försökt skriva rapporten på ett lättläst sätt och hoppas att allt går att förstå. Siffror inom hakparentes (ex. [20]) indikerar från vilken källa vi hämtat information till delar av det som står skrivet i det avsnittet.

Vi vill tacka:

- Sterner Specialfabrik AB och speciellt Håkan Sterner, för att ha gett oss chansen att göra detta examensarbete.
- Mats Isberg, för den handledning vi fått av honom och för att ha lånat ut utrustning som vi haft god nytta av.
- Johan Ekholm, för att ha varit tillgänglig när vi behövt hjälp.

2. Introduktion

2.1 Bakgrund

Sterners Specialfabrik AB har sedan 1958 tillverkat och marknadsfört myntautomater. De tillverkar även andra sorters automater, t.ex. biljettautomater och har uttryckt en önskan om att kunna kommunicera med dessa via Internet genom att koppla automaten till ett nätverk. Man har idag inget sätt att titta på t.ex. statistik vad gäller användning av automaterna utan att behöva åka till själva platsen där automaten finns och tänkte att detta var ett bra sätt att kunna göra det.

2.2 Syfte

Syftet med arbetet har varit att få TCP/IP-kommunikation mellan dator och automat att fungera och sedan ge en bra beskrivning så att företaget kan tillämpa det hela. Exakt hur det ska fungera har inte specificerats utan arbetet har setts som forskande eftersom Sterners vet väldigt lite om möjligheterna med detta själva.

Syftet har också varit att göra ett projekt som visar att studenten kan klara av att tillämpa de kunskaper som samlats under studietiden och använda dom ute i arbetslivet. Arbetet ska också vara utvecklande och tvinga studenten att lära sig nya saker för att klara av att göra det samt även vara till nytta för företaget uppgiften utförs i uppdrag av.

Arbetet är på C-nivå och ska därför ses som forskningsförberedande, vilket gör utförandet och redovisningen av arbetet viktigt.

2.3 Mål

Målet med arbetet har varit att testa så många funktioner som det går med hjälp av utvecklingskorten, för att se vad som är möjligt.

Ett mål har också varit att använda en PIC-mikrokontroller och programvaran MPLAB IDE och programmerings-/debugutrustningen MPLAB ICD 2 för utvecklandet av applikationen, eftersom det här är utrustning som Sterners själva använder sig av dagligen.

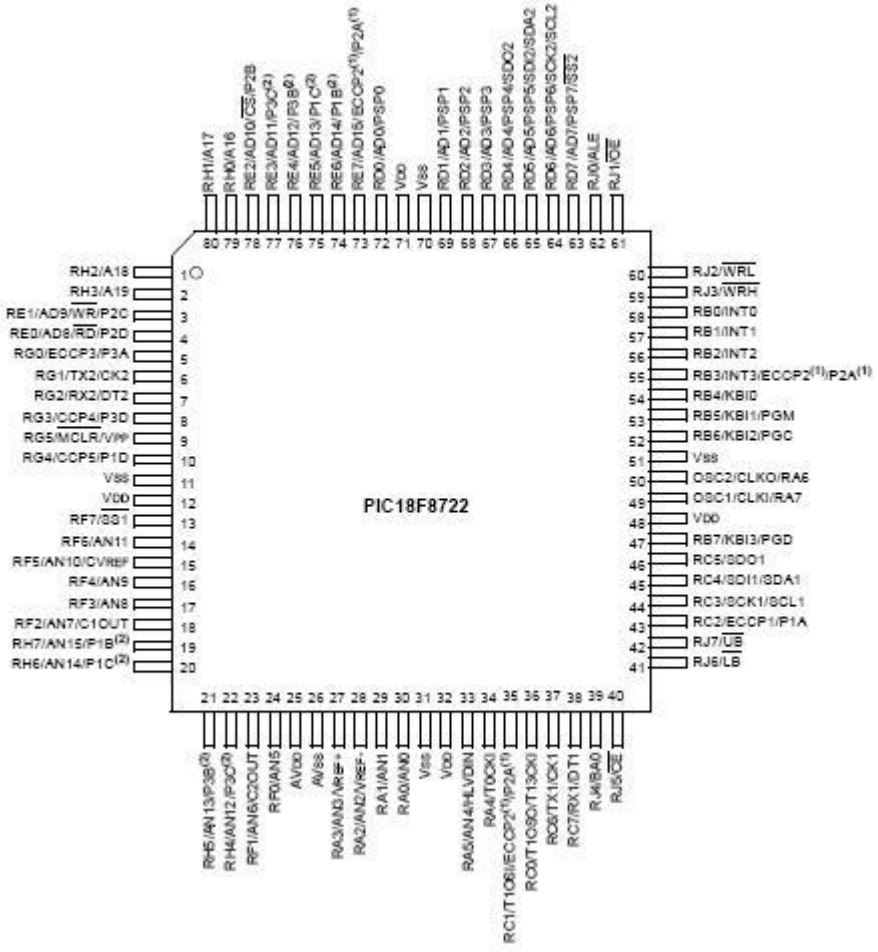
3 Hårdvarubeskrivning

3.1 Mikrokontrollern PIC18F8722

[23]

3.1.1 Översikt

Mikrokontrollern som används i detta arbete är tillverkad av Microchip och ingår i deras PIC18F-serie. Den har en 8-bitars arkitektur som kan arbeta i upp till 40MHz. Programminne och dataminne finns internt i mikrokontrollern men externa minnen kan användas med en upp till 20 bitar bred adressbuss och en 8 eller 16 bitar bred databuss. Det finns också 4st seriella portar vilket ger möjligheten att kommunicera med andra enheter. Mikrokontrollern sitter monterad på ett utvecklingskort som heter PICDEM HPC Explorer Board vilket beskrivs utförligare nedan. Mikrokontrollern har 80st anslutningspinnar varav 70 är I/O pinnar, bilden nedan visar dessa.

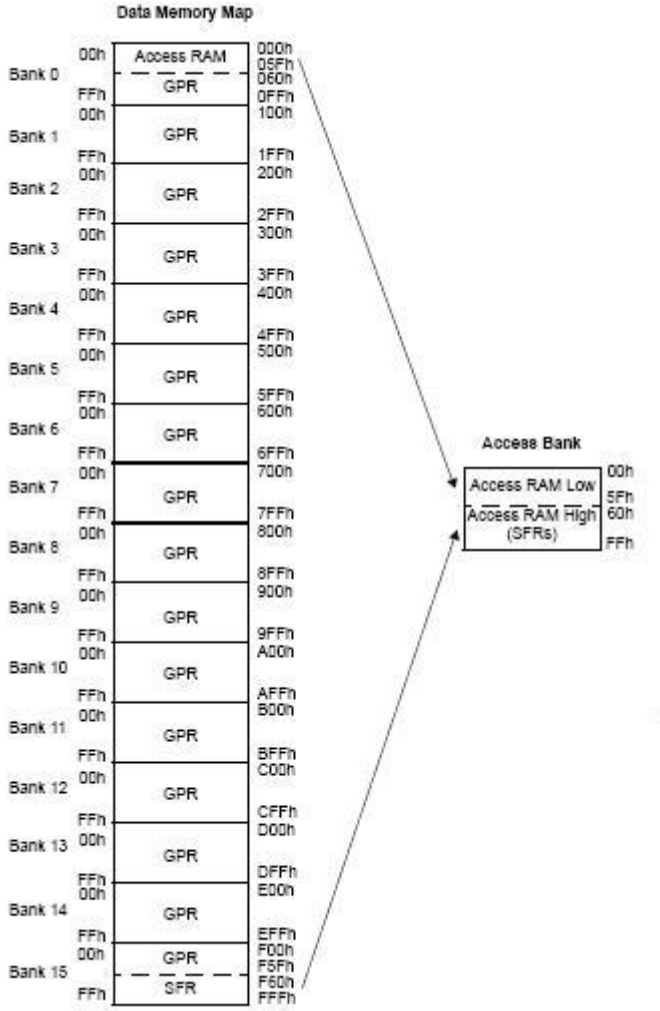


Figur 3.1: Anslutningar för PIC18F8722

3.1.2 Interna minnen

PIC18F8722 har 3st interna minnen. Dessa är program-, RAM- och EEPROM-minne. Programminnet är av typen Enhanced Flash och rymmer 128 Kbytes. I detta minne ligger programmet lagrat som instruktioner om vardera 2 eller 4 byte. Programminnet adresseras av den interna adressbussen som är 21 bitar bred vilket ger möjligheten att adressera 2 Mbyte. Att man kan adressera mer än de 128 Kbyte som finns internt beror på att man ska ha möjligheten att även adressera externt minne. För att kunna hantera hopp i programkoden så finns det en stack med plats för 31st återhoppsadresser.

RAM-minnet (se Figur 3.2) är också internt och rymmer 4096 byte data som är fördelat på 16st banker med 256 byte vardera. För att kunna göra snabba minnes-accesser finns även en Access Bank om 256 byte som innehåller SFR (Special Function Register) och 96 byte av bank 0. SFR-registret tar upp 160 byte vilket gör att det finns 3936 byte kvar till datalagring. Minnet adresseras med en 12 bitars adress där de 4 mest signifikanta bitarna anger vilken bank som används och de lägre 8 bitarna anger adressen i banken.



Figur 3.2: RAM minnets uppdelning i olika banker

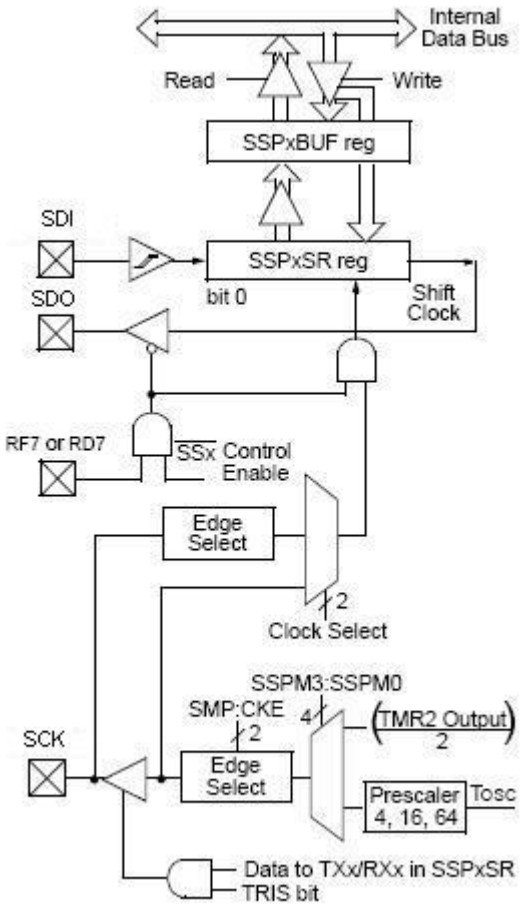
Det finns även ett dataminne av typen EEPROM som rymmer 1024 byte vilket används för lagring av värden en längre tid. EEPROM minnet adresseras via register i SFR och läsning/skrivning av data sker även det via register i SFR.

3.1.3 Seriella portar

Mikrokontrollern har tillgång till 4st seriella portmoduler varav två är s.k. Master Synchronous Serial Port- (MSSP) moduler och de andra två är Enhanced Universal Synchronous Asynchronous Receiver Transmitter- (EUSART) moduler. MSSP-modulerna kan konfigureras på två sätt, antingen som Inter-Integrated Circuit (I²C™) eller som Serial Peripheral Interface (SPI™). Det senare alternativet har använts i detta arbete.

SPI™ används för att kommunicera med perifer utrustning eller andra mikrokontrollers. Flera enheter kan vara anslutna till samma SPI och kommunicerar då på samma ledning, där en av enheterna agerar som en s.k. Master och de andra är s.k. Slave. Vilken enhet som man kommunicerar med väljs av mastern med en s.k. chip select-signal.

För SPI kommunikationen används 3st I/O pinnar: Serial Clock (SCK), Serial Data In (SDI) och Serial Data Out (SDO). Sedan behövs en pinne för varje enhet som ansluts som man kan skicka en chip select-signal på. Figur 3.3 visar hur SPI modulen ser ut.



Figur 3.3: Blockdiagram över SPI-modulen

Den andra typen av seriella portar som finns på mikrokontrollern är av typen Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART). Dessa kan konfigureras till att vara asynkrona för kommunikation med t.ex. en dator eller också kan de konfigureras till att vara synkrona för kommunikation med olika typer av perifer utrustning.

3.1.4 Avbrottshantering

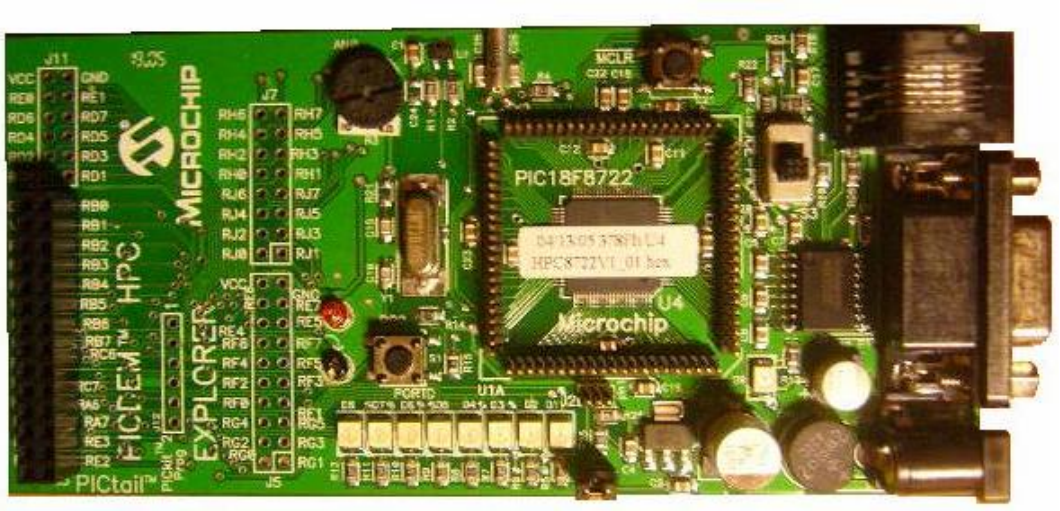
Det finns totalt 29st avbrottskällor som kan användas för att avbryta pågående programexekvering. Vilka avbrottskällor som ska generera avbrott bestäms genom att ställa in olika register. Det finns även en möjlighet att ge olika typer av avbrott olika prioritet – hög och låg prioritet. Beroende på om avbrottet hade låg eller hög prioritet hoppar programmet till respektive adress. För avbrott med låg prioritet används adressen 0018h och för hög prioritet 0008h. Ett avbrott med hög prioritet kan avbryta en pågående avbrottsrutin med låg prioritet. Möjligheten till olika prioritet på avbrotten kan stängas av så att alla avbrott hanteras lika, då görs alltid ett hopp till adress 0008h vid ett avbrott. För avbrott från externa enheter så finns det 3st pinnar som kan användas som avbrottskällor.

3.1.5 Övriga egenskaper

Det finns 70st I/O pinnar fördelade på 9 olika portar – port A, B, C, D, E, F, G, H och J. Portarna kan programmeras till att vara ingång eller utgång, vilket bestäms av portens kontrollregister som kallas TRISX där X är namnet på porten. Alla portar har 8st pinnar utom port G som har 6st. Ofta påverkas portarna av att annan utrustning behöver använda deras pinnar. T.ex. så använder SPI kommunikationen 3 pinnar på port C. Mikrokontrollern har en 10 bitars analog till digital omvandlare med möjlighet till 16st ingångskanaler. Vilken kanal som skall användas väljs genom interna register i SFR. Insignalen på den valda kanalen omvandlas till ett digitalt värde med 10 bitar. Det finns även 5st timer-moduler, timer0-timer4. Timer0 är en 8 eller 16 bitars räknare eller timer som kan generera avbrott vid ett overflow. Timer1 och timer3 är 16 bitars timers eller räknare med möjlighet till avbrott vid overflow. Timer2 och timer4 är 8 bitars timers som räknas upp på varje klockcykel och sedan jämförs med ett värde. Om de är lika genereras ett avbrott.

3.1.6 PIC18F8722 på PICDEM HPC Explorer Board

Mikrokontrollern som avänds är monterad på ett utvecklingskort som heter PICDEM HPC Explorer Board och är tillverkat av Microchip. På kortet finns det en 10MHz kristall vilket gör att den maximala frekvensen 40MHz kan användas om frekvensen omvandlas internt av en så kallad PLL- (Phase Locked Loop) krets. Kortet har en anslutning för MPLAB ICD2 (In-Circuit Debugger) som används för programmering och debugging. Vidare finns en RS232-port med 9 pinnars D-typ anslutning för att ansluta till t.ex. en PC. För att kunna göra diverse experiment på kortet finns det 8st lysdioder anslutna till port D, en temperatursensor, en potentiometer samt två tryckknappar. Det finns även en kristall som svänger med frekvensen 32.768kHz monterad vilket är mycket användbart om man vill implementera en realtidsklocka. En anslutning för strömkälla finns också, till vilken en 9V likspänningskälla ansluts för att ge kortet strömförsörjning.



Figur 3.4: PICDEM HPC Explorer Board

3.2 Ethernetkontrollern ENC28J60

[25]

3.2.1 Egenskaper

Ethernetkontrollern ENC28J60 är även den tillverkad av Microchip och den arbetar på de två lägsta lagren enligt OSI-modellen – det fysiska lagret och datalänklagret. Ethernetkontrollern sköter en 10BASE-T ethernetanslutning och följer IEEE 802.3-standarden. Den har en sändar-/mottagarbuffert som rymmer 8Kbytes data. För att kunna kommunicera med mikrokontrollern finns ett SPI interface genom vilket instruktioner och data kan skickas till och från ethernetkontrollern. Hastigheten på SPI-överföringen kan vara upp till 10Mb/s. ENC28J60 är konstruerad för att arbeta med en frekvens på 25MHz.

Ethernetkontrollern har inbyggda funktioner för automatisk återsändning vid kollision, automatisk utfyllnad s.k. padding, generering och kontroll av CRC samt automatisk borttagning av ej önskvärda paket. Ethernetkontrollern har en DMA- (Direct Memory Access) kontroller som används för att dels kopiera data mellan olika adresser i bufferten samt för att beräkna en 16 bitars checksum.

ENC28J60 har ett flertal avbrottskällor som kan användas för att generera avbrott till mikrokontrollern via 2st utgångar. Avbrottskällorna delas in i två grupper; control events (INT) och wake-up on LAN (WOL), och är sedan kopplade till varsin utgång.

3.2.2 ENC28J60 på PICtail Ethernet Board

Ethernetkontrollern ENC28J60 är monterad på ett dotterkort som lätt kan kopplas samman med moderkortet HPC Explorer Board. Dotterkortet heter PICtail Ethernet Board och tillverkas av Microchip. Det är konstruerat för att passa ihop med några av deras demonstrations- och utvecklingskort som t.ex. HPC Explorer board som vi använder oss av. På kortet finns de komponenter som är nödvändiga för att ethernetkontrollern ska kunna kommunicera på ethernet. Kortet har även ett EEPROM på 32 Kbytes som är försett med SPI-interface. EEPROMet och ethernetkontrollern använder samma SPI-buss för kommunikation med mikrokontrollern, vilket beskrivs utförligare i nästa avsnitt.

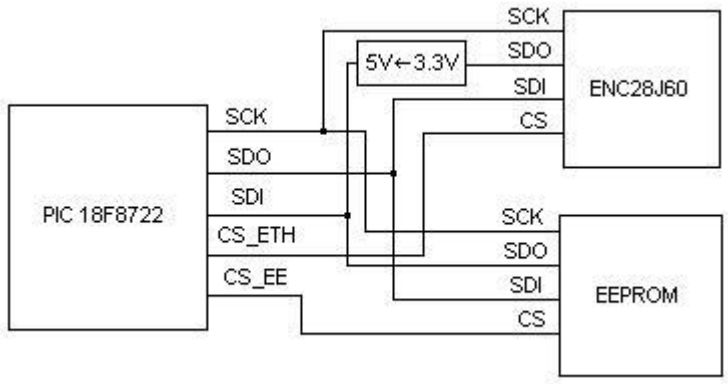
För att kunna ansluta kortet till ethernet finns det en RJ-45 anslutning vilken även har 2st lysdioder (grön och gul) för att indikera länk- och trafikstatus. Kortet har egen strömförsörjning och arbetar på två olika spänningsnivåer: 3.3V och 5V. Strömförsörjningen kan även fås från moderkortet om så önskas eller också kan moderkortet strömförsörjas av dotterkortet. Ethernetkontrollern ENC28J60 arbetar med spänningen 3.3V medan EEPROMet arbetar med 5V. Signalerna från mikrokontrollern ligger på 5V, vilket går att koppla direkt till ethernetkontrollern då dom ingångar som används för kommunikation med mikrokontrollern tål 5V. När en signal ska skickas från ethernetkontrollern måste den däremot först omvandlas till 5V vilket görs i en spänningsomvandlare på kortet. Det finns även en kristall som ger en frekvens på 25 MHz till ENC28J60.

Figur 3.5: PICtail Ethernet Board

3.3 Kommunikationen mellan PIC18F8722 och PICtail Ethernet Board

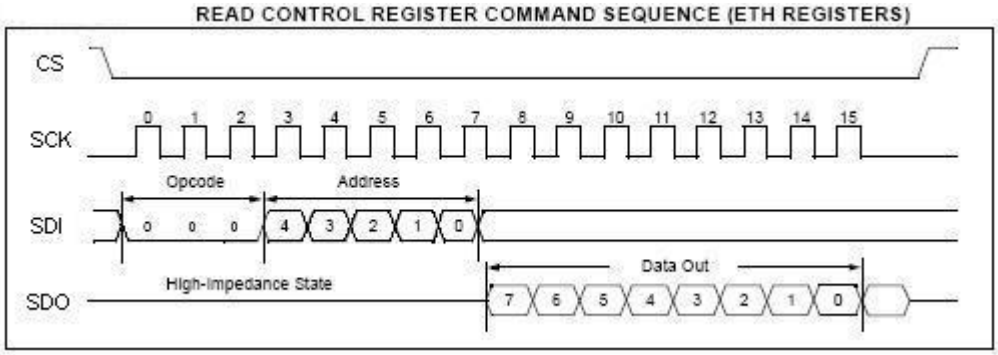
[23][25]

För kommunikation mellan PICen, ethernetkontrollern och EEPROMet används SPI-kommunikation, vilket beskrevs i avsnitt 3.1.3. I figur 3.6 visas hur SPI-bussen är kopplad mellan de olika enheterna. Mikrokontrollern agerar master och har därför 2st chip select-utgångar för att välja vilken enhet den vill kommunicera med. Det är också masterns uppgift att skicka en klocksignal på SCK-ledningen. Av figuren framgår också att SCK, SDO och SDI-ledningarna är samma för ENC28J60 och EEPROMet. Därför krävs att utgången SDO från ENC28J60 och EEPROMet är hög impedans då de inte används, detta för att undvika att två källor driver samma signal. För ENC28J60 krävs även en omvandling på utgången, så att den logiska nivån kan tolkas rätt av PICen eftersom ENC28J60 arbetar med spänningen 3.3V.



Figur 3.6: SPI anslutningar mellan PIC, EEPROM och ENC28J60

Chip select-signalerna är aktivt låga, dvs. när man vill kommunicera med en enhet blir signalen låg och när kommunikationen är klar återgår signalen till att vara hög. I figur 3.7 nedan visas ett exempel på en läsning av ett kontrollregister i ENC28J60. Chip select-signalen CS blir först låg, vilket startar kommunikationen, och sedan klockas data in på stigande klockflank. Man kan också se att utgången SDO är en hög impedans när den inte skickar ut någon data.



Figur 3.7: Läsning av kontrollregister på ENC28J60

För att styra ethernetkontrollern ENC28J60 finns det 7st instruktioner som kan användas. I figur 3.8 finns en lista över dessa. De instruktioner som finns är läsning och skrivning till kontrollregistren, läsning och skrivning av buffert minnet, en funktion för att bestämma värdet på en specifik bit i ett register samt en reset-funktion.

TABLE 4-1: SPI™ INSTRUCTION SET FOR THE ENC28J60

Instruction Name and Mnemonic	Byte 0		Byte 1 and Following
	Opcode	Argument	Data
Read Control Register (RCR)	0 0 0	a a a a a	N/A
Read Buffer Memory (RBM)	0 0 1	1 1 0 1 0	N/A
Write Control Register (WCR)	0 1 0	a a a a a	d d d d d d d d
Write Buffer Memory (WBM)	0 1 1	1 1 0 1 0	d d d d d d d d
Bit Field Set (BFS)	1 0 0	a a a a a	d d d d d d d d
Bit Field Clear (BFC)	1 0 1	a a a a a	d d d d d d d d
System Command (Soft Reset) (SC)	1 1 1	1 1 1 1 1	N/A

Legend: a = control register address, d = data payload.

Figur 3.8 SPI instruktioner för ENC28J60

EEPROMet styrs också av instruktioner som den får via SPI-kommunikationen. För EEPROMet finns det 6st instruktioner som kan användas. Dessa är läsning och skrivning av minnet, läsning och skrivning av statusregistret samt möjlighet att välja om skrivning av minnet ska vara tillåtet eller inte. Figur 3.9 nedan visar de olika instruktionerna.

TABLE 2-1: INSTRUCTION SET

Instruction Name	Instruction Format	Description
READ	0000 0011	Read data from memory array beginning at selected address
WRITE	0000 0010	Write data to memory array beginning at selected address
WRDI	0000 0100	Reset the write enable latch (disable write operations)
WREN	0000 0110	Set the write enable latch (enable write operations)
RDSR	0000 0101	Read STATUS register
WRSR	0000 0001	Write STATUS register

Figur 3.9: SPI instruktioner för EEPROM [28]

4 Nätverksprotokoll

[30][27]

4.1 Allmänt om TCP/IP

TCP/IP-modellen skapades av det amerikanska försvaret för att de ville ha ett nätverk som överlevde oavsett förhållanden, till och med kärnvapenkrig. TCP/IP skapades dock som en öppen standard, fri för vem som helst att använda. Detta gjorde att TCP/IP snabbt blev en standard i nätverk och används idag för Internet. TCP/IP indelas i fyra lager: Application, Transport, Internet och Network Access lagret.

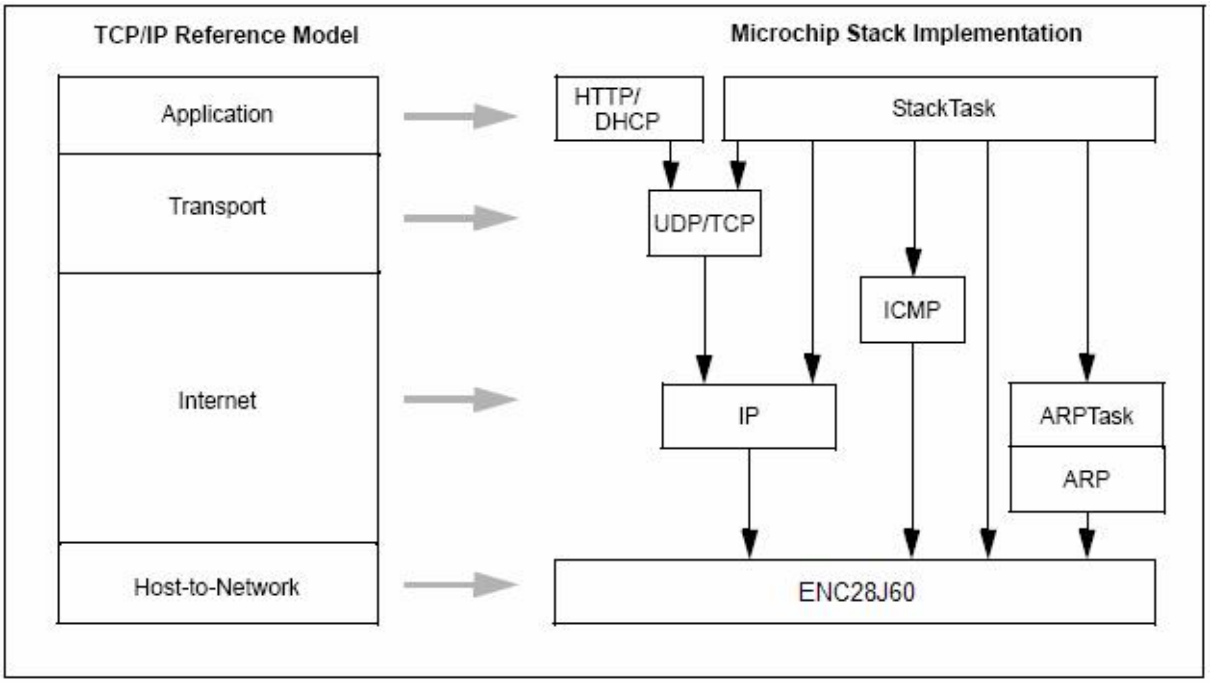
- Application lagret hanterar saker som representation och dialogkontroll. Vanliga protokoll här är FTP (File Transfer Protocol), HTTP (Hypertext Transfer Prot.), SMTP (Simple Mail Transfer Prot.) och DHCP (Dynamic Host Configuration Prot.).
- Transport lagret har hand om tillförlitlighet, felkorrigering, flödeskontroll och kvalitetskontroll hos nätverket. TCP (Transmission Control Protocol) är ett protokoll som utför detta, UDP (User Datagram Protocol) ett annat.
- Internet lagret har som uppgift att dela upp TCP-segmenten i paket och skicka iväg dom. IP (Internet Protocol) sköter detta. Beräkning av bästa väg samt packet switching sker i detta lager.
- Network Access lagret hanterar alla fysiska och logiska komponenter som behövs för att upprätta en fysisk länk.

TCP ger alltså en tillförlitlig transport. Det arbetar genom att det hela tiden håller en dialog mellan källa och destination medan informationen som skapas i Application lagret paketeras i segment. IP delar sedan segmenten i paket och agerar som vägvisare till destinationen.

4.2 Microchips TCP/IP-stack

Vid en jämförelse (se Figur 4.1) mellan Microchips TCP/IP-stack och TCP/IP-modellen ser man att, till skillnad från modellen, har många av lagren i Microchips stack kontakt med lager som inte ligger direkt under dem, detta för att slippa onödiga processer när de inte behövs.

En annan skillnad är två extra moduler; ”StackTask” och ”ARPTask”, i stacken. StackTask sköter hela stackens och dess modulers operationer, medan ARPTask har hand om ARP-lagrets uppgifter. Microchips TCP/IP-stack använder sig av en teknik kallad Cooperative multitasking. Med detta menas att det finns mer än en ”task”, och när en (Stacktask) utfört sitt jobb kan nästa (ARPTask) utföra sitt. Applikationer som ska använda sig av stacken måste också använda sig av Cooperative multitasking. HTTP-server-applikationen beskriven i denna rapport gör detta genom att dela upp stora uppgifter i flera små uppgifter med en så kallad Finite state machine.



Figur 4.1: Microchips stack jämförd med TCP/IP-modellen

4.3 Protokollen

I Microchips TCP/IP-stack är varje protokoll representerat av en modul.

4.3.1 ENC28J60

ENJC28J60 representerar Network Access-lagret (eller MAC-lagret) i stacken och är speciellt anpassat för ethernetkortet, ENCJ28J60.

Bland de funktioner som ingår i denna modul finns *MACInit,* som ställer in PICens SPI-modul och alla ethernetkortets register som behövs så att normal operation kan påbörjas. Andra saker som konfigureras i denna modul:

- Inställningar vad gäller mottagning och sändning av paket.
- Kontroll av den fysiska länken, andras MAC-adresser.
- Kontroll om en sändning/mottagning är igång, när den är klar etc.
- Checksum beräkning.
- Duplex Mode.
- Ethernetkortets Sleep Mode.

4.3.2 ARP

Eftersom MAC-protokollet (ENC28J60) inte alls förstår vad en IP-adress är måste det finnas ett protokoll som kan översätta den till något det förstår, annars kommer data inte kunna passera till de övre lagren. ARP är ett protokoll som automatiskt mappar en IP-adress till rätt MAC-adress. Detta görs genom att en *ARP request* skickas ut som ett broadcast från den som har ett IP-paket att skicka. Detta broadcast innehåller destinationens IP-adress. När det nått fram till destinationen skickas ett *ARP reply* tillbaka, vilket innehåller destinationens korrekta MAC-adress. Om IP-adressen inte ligger på samma nät fås istället Default Gateways MAC-adress.

I stacken innehåller modulen ARP alla ARP-funktioner medan ARPTask utnyttjar dessa funktioner för att ge en komplett ARP-funktionalitet. ARPTask är implementerad som en kooperativ tillståndsmaskin, som svarar på ARP requests utifrån. ARPTask kan operera i två lägen: Server mode eller Server/Client mode. Eftersom stacken används som en HTTP-server blir koden mindre, och därmed sparas minne, genom att kompilera ARPTask i Server mode.

4.3.3 IP

IP är den vanligast implementerade hierarkiska nätverksadresserings-metoden. Man säger att IP är otillförlitligt och ett ”best effort protocol”. Med detta menas att IP inte verifierar att data som sänts verkligen når fram till sin destination.

IP-paket

0		8	16	24
VERS(4)	HLEN(4)	Service Type(8)	16-bit Total Length of Packet	
16-bit Identification			Flags(3)	13-bit Fragment Offset
Time to Live (8)		Protocol (8)	16-bit IP Header Checksum	
32-bit Source IP Address				
32-bit Destination IP Adress				
IP Options (if any)				Padding
Data				
…				

I Microchips TCP/IP-stack agerar IP bara som ett passivt protokoll – det svarar inte på datapaket, utan de övre lagren använder IP-funktioner, som ligger i modulen IP.c, för att hämta IP-paket, tyda dem och utföra det som ska göras.

4.3.4 ICMP

IP har ingen funktion som visar om en datasändning har fallerat – detta har man istället ICMP till. ICMP rapporterar tillbaka till källan för datasändningen när det blivit ett fel.

Dock gör ICMP inget för att rätta till felet utan felkorrigering sker i de övre lagren. ICMP.c är den modul som har hand om ICMP.

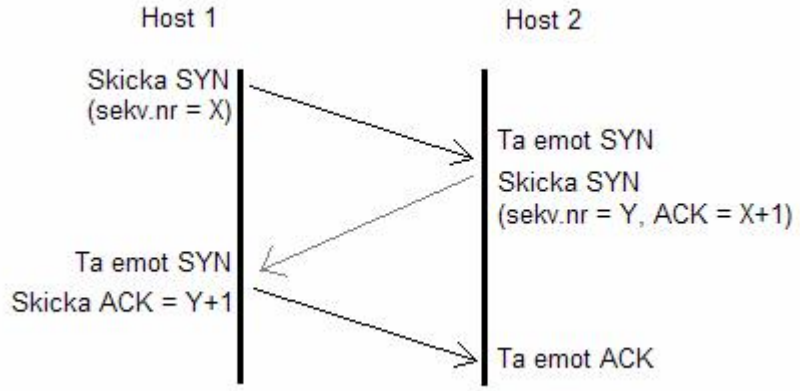
ICMP-meddelande

0	8	16	24
8-bit Type	8-bit Code	16-bit ICMP Header Checksum	
Data			
…			

Även ICMP är implementerat i stacken som ett passivt protokoll. Precis som IP arbetar det inte självt utan de övre lagren kan använda ICMP-funktioner (i modulen ICMP.c) för att generera ICMP-meddelanden. Det används t.ex. för att svara på ping-paket.

4.3.5 TCP

TCP ser alltså till att transporten av data blir tillförlitlig och det kräver att en anslutning är upprättad innan data börjar skickas. De två ”hostar” som ska kommunicera måste göra en synkronisering av sina första sekvensnummer (sequence number). Detta görs genom en mekanism (se Figur 4.2) som väljer de första sekvensnumren och en handskakning mellan ”hostarna” för utbyte av dem.



Figur 4.2: TCP:s handskakningsmekanism

Denna mekanism, som kallas tree-way handshake, fungerar så här:

1. Host 1 börjar med att skicka ett SYN-paket, innehållandes sitt första sekvensnummer (X), till Host 2.

2. Host 2 tar emot paketet och vet nu om Host 1:s sekvensnummer. Host 2 skickar tillbaka ett eget SYN, innehållandes sitt eget sekvensnummer (Y) samt en verifiering (ACK = X+1) som berättar att den härnäst förväntar sig X+1 från Host 1.

3. Host 1 tar emot paketet och vet nu om Host 1:s sekvensnummer och skickar en verifiering (ACK = Y+1), vilket avslutar processen.

TCP tillför följande till anslutningen:

- Flow Control – Mottagaren av en sändning kanske inte kan ta emot ett helt paket. TCP hjälper till med detta genom att dela upp datan i segment. TCP använder sedan något som kallas Windowing för att se till att segmenten kommer fram. Varje TCP-segment numreras så att de kan sätts ihop i rätt ordning när de kommit fram.
- Reliability – TCP förväntar sig alltid ett ACK från destinationen som bekräftar att data nått fram, vilket ger pålitlighet. Window-storleken, som talar om hur många segment som kan skickas innan vi förväntar oss ett ACK, varierar dynamiskt och sätts genom en dialog mellan källa och destination.

TCP-segment

0		8	16	24
16-bit Source Port			16-bit Destination Port	
32-bit Sequence Number				
32-bit Acknowledgment Number				
Hlen (4)	Reserved (6)	Code Bits (6)	16-bit Window size	
16-bit TCP Checksum			16-bit Urgent Pointer	
Options (if any)				Padding
Data				
…				

TCP är implementerat i stacken som ett aktivt lager. Det finns stöd för upp till 253 TCP-sockets, vilket gör det möjligt att upprätta flera TCP-anslutningar samtidigt. Detta är användbart i applikationen om fler än en användare skulle vilja ansluta till servern. Dock gör fler sockets att den genomsnittliga processeringstiden för TCP ökar. Olikt andra TCP/IP-applikationer delar stackens sockets på alla tillgängliga transmit buffers. Detta gör att man måste se till att det finns tillräckligt med transmit buffers för alla sockets, så att inte vissa blir utan om alla skulle reserveras av andra sockets – vilket skulle leda till att det inte går att ansluta genom de sockets som blivit utan. På mottagarsidan finns bara en enda receive buffer, vilket innebär att när en socket tar emot data måste den hämta allt på en gång. Den kan inte hämta en del och hoppas på att ta resten senare eftersom buffern ska kunna användas av andra sockets och måste rensas.

I Microchips TCP/IP-stack finns automatisk retry implementerat. Detta innebär att en transmit buffer är reserverad tills det kommit ett ACK från mottagarsidan. Genom kommandot #define TCP_NO_WAIT_FOR_ACK kan man slå av detta för att få bättre genomströmning av data, vilket utnyttjas i denna version.

4.3.6 UDP

UDP arbetar på samma nivå i TCP/IP-modellen som TCP. Till skillnad från TCP tillför UDP otillförlitlig sändning av data, där en upprättad anslutning ej krävs innan data skickas. Inte heller någon verifiering på att data verkligen når sin destination krävs. UDP är ett enklare protokoll än TCP och kan därför med fördel användas när en helt tillförlitlig datatransport inte behöver garanteras. DHCP är ett protokoll som använder sig av UDP.

UDP-segment

0	8	16	24
16-bit Source Port		16-bit Destination Port	
16-bit Message Length		16-bit UDP Checksum	
Data			
…			

UDP är, precis som TCP, implementerat som ett aktivt lager i stacken. Det finns stöd för upp till 254 UDP-sockets och det fungerar likadant som för TCP vad gäller Receive och Transmit buffers. I stacken krävs ingen UDP-checksum-beräkning utan checksum-fältet sätts till noll för att spara minne. Detta innebär att alla moduler som använder sig av UDP själva måste kontrollera att datan hålls riktig.

4.3.7 HTTP

HTTP står för Hypertext Transfer Protocol och är det protokoll som används för att överföra text, bilder mm över World Wide Web. Protokollet ligger på applikationslagret i TCP/IP-modellen och det använder sig av det underliggande TCP-protokollet för att skapa anslutningar. En HTTP-kommunikation kräver en server och en klient. Klienten kan vara en webbläsare på en dator som kommunicerar med en webbserver. Klienten skickar en request till servern på TCP-port 80 och servern svarar då med ett response som skickas tillbaka till klienten. Webbläsaren åskådliggör data som skickats från webbservern genom att skapa en webbsida utifrån den HTML- (Hypertext Markup Language) fil som skickats. HTML-filen beskriver hur webbsidan ska se ut då den visas i webbläsaren.

Ett HTTP-request består främst av tre delar: metod, URI (Uniform Resource Identifier) och version. Metoden är oftast GET och betyder att man vill att servern ska skicka det som finns på den sökväg man anger i URI. Version anger vilken version av HTTP man använder.

Servern svarar på ett request genom att skicka ett response som också främst innehåller tre delar: version, statuskod och ett meddelande. Statuskoden talar om för klienten om requestet lyckades eller inte och ifall det inte lyckades anges orsaken i statuskoden. Meddelandet är samma sak som statuskoden men i textformat, så att det blir läsligt för användaren.

Efter de tre första delarna i ett request och ett response följer oftast ett antal headers som ger information om responset eller requestet. En header kan även tala om vad som skickas med i HTTP-överföringen, t.ex. en HTML-fil eller en bild.

Efter headerdelen följer själva innehållet som skickats med vid ett response. Det kan t.ex. vara en HTML-fil som man begärt. Nedan visas ett exempel på en HTTP-kommunikation.

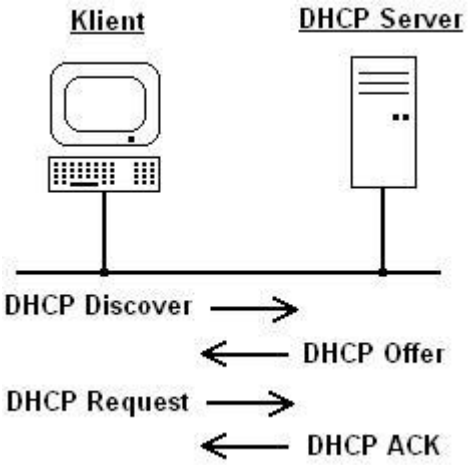
Request:
GET /folder/index.html HTTP/1.1
Host: www.microchip.com
User-agent: Mozilla/5.0
.....(ev. Fler headers)

Response:
HTTP/1.1 200 OK
Date: Tue, 02 May 2006 10:46:13 GMT
Server: Microsoft-IIS/6.0
Content-type: text/html
Content-length: 6425
.....(ev. Fler headers)

<html>
<body>
(HTML text som beskriver sidans innehåll)
</body>
</html>

4.3.8 DHCP

DHCP står för Dynamic Host Configuration Protocol och används för att automatiskt göra nödvändiga inställningar för TCP/IP hos en klient som skall anslutas till ett ethernetnät. Klienten kan t.ex. vara en dator som saknar inställningar för TCP/IP, såsom IP adress, gateway-adress mm. Dessa parametrar kan erhållas från en DHCP-server som är ansluten till samma nät som klienten. Klienten gör ett s.k. DHCP discover som ett broadcast på nätet och DHCP-servern kommer då att svara med ett DHCP offer, som innehåller IP adress, Default Gateway adress och DHCP-server-adress. Svaret skickas som ett broadcast eftersom klienten inte har någon IP-adress. Om klienten accepterar den adress som erbjöds av DHCP-servern skickas ett DHCP request tillbaka till servern som då svarar med ett DHCP acknowledge, vilket också sänds som ett broadcast från DHCP-servern. När klienten tagit emot DHCP acknowledge från servern börjar den använda de nya inställningarna.



Figur 4.3: DHCP-kommunikation

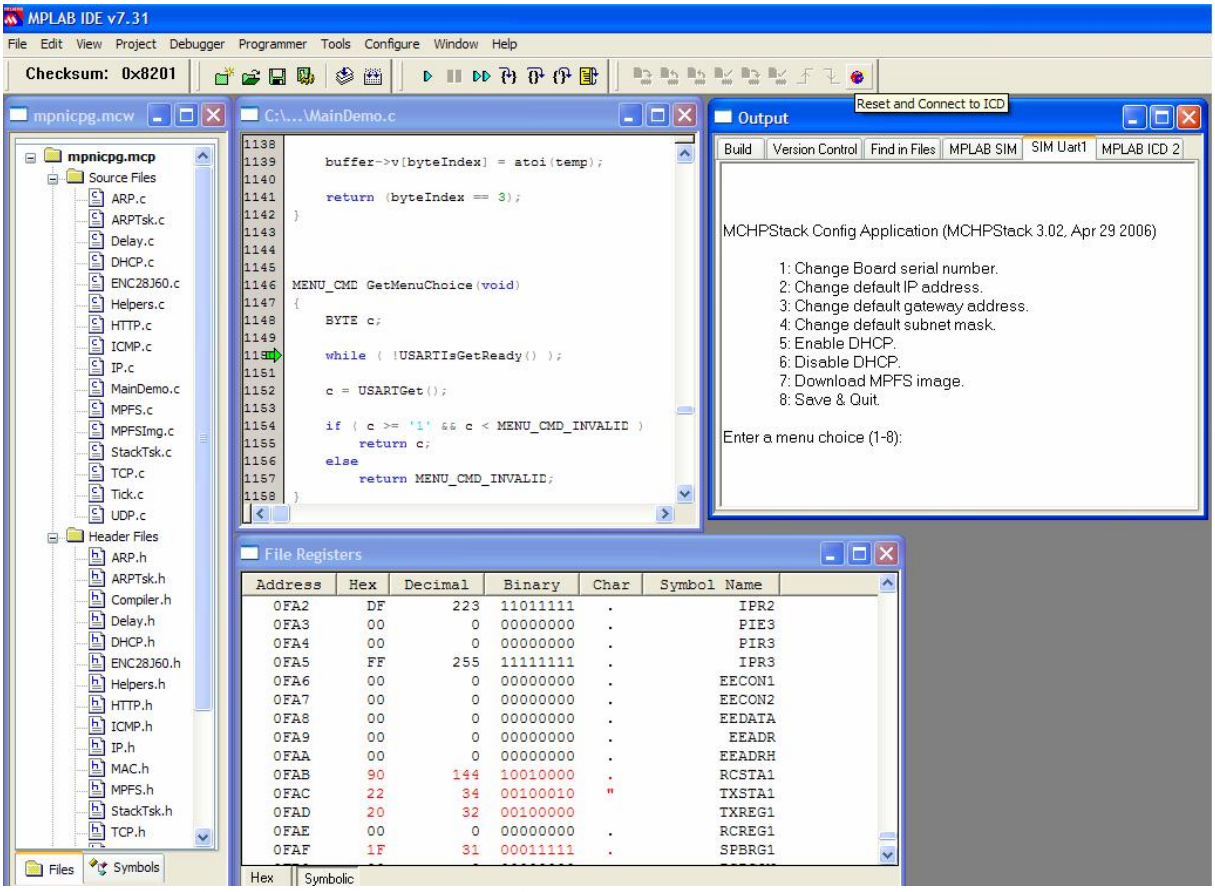
5. Genomförande

5.1 Programmeringsutrustning

Vi valde i vårt projekt att primärt hålla oss till Microchips utrustning och programvaror för att slippa tänka på kompatibilitetsproblem med PIC, ethernetkort och stack. Dessutom är de flesta programvaror vi använt oss av gratis.

5.1.1 MPLAB IDE

Utvecklingsmiljön heter MPLAB IDE och är en gratis programvara [12] från Microchip. MPLAB är i grunden avsedd för Assemblerprogrammering, men med tillägget C18 har man möjlighet att istället programmera i C. Förutom kompilering finns också möjlighet att simulera i MPLAB, vilket gör att man inte alltid behöver programmera PICen för att testa programmet.



Figur 5.1: MPLAB IDE arbetsfönster

5.1.2 MPLAB C18

MPLAB C18 är Microchips egna C-kompilator för användning med MPLAB IDE. Eftersom TCP/IP-stacken var skriven i C var det ganska självklart att utnyttja C18 för att kunna använda och modifiera stacken för applikation. Det är dessutom betydligt enklare att programmera i C än i Assembler, som hade varit alternativet. Vi använde oss av en gratis studentversion [11] som innehöll alla funktioner som fullversionen innehar, i 60 dagar.

5.1.3 MPLAB ICD 2

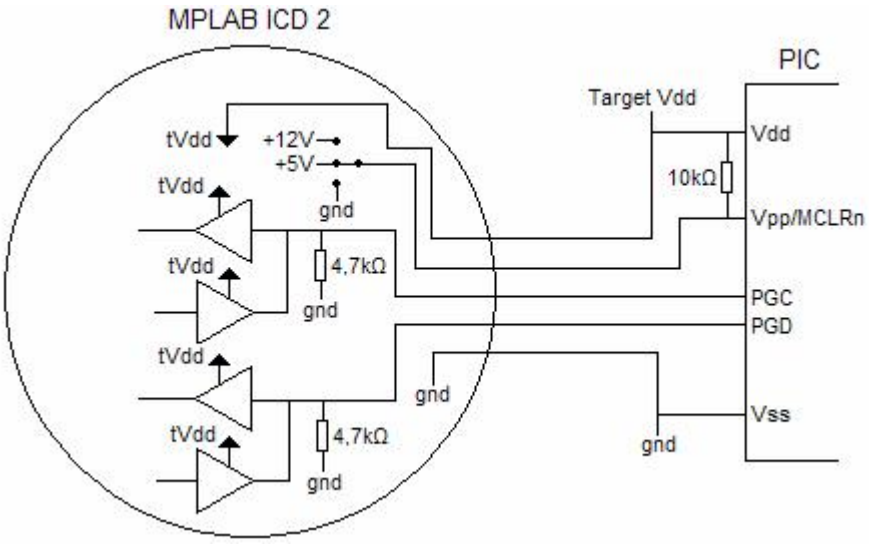
MPLAB ICD 2 [21] (se Figur 5.2) är en debugger och programmerare i ett, avsedd för PIC mikrokontrollers. Den ansluts till en PC med antingen en RS-232 (seriell) kabel eller en USB-kabel (vilket vi använt oss av). När en USB-kabel används behöver inte någon spänningskälla anslutas till ICD 2 eftersom den då får ström via datorns USB-port. ICD 2 ansluts sedan till Explorerkortet med en 6-pins modular interface-kabel.

Figur 5.2: MPLAB ICE 2 programmer/debugger

Med ICD 2 kan debugging utföras i realtid genom att programmera görs i Debug mode – PICarna har inbyggd support för detta. Man kan sedan sätta breakpoints, stega sig igenom programmet etc. Tyvärr hade vi problem med Debug mode, (det fungerade inte alltid att köra programmet after att ha programmerat i Debug mode) så vi använde oftast Program mode för att programmera PICen istället.

I figur 5.3 kan man se hur ICD 2 kopplas till en PIC. Vid programmering sätter ICD 2 nödvändig spänningsnivå på Vpp. Klockpulser skickas till PGC och seriell data till PGD. För att verifiera att programmeringen gjorts korrekt skickas klockpulser till PGC samtidigt som data läses tillbaka från PGD.

Genom Vdd kan ICD 2 ge PICen en begränsad spänning under särskilda förhållanden. Vdd är ej nödvändig för grundläggande ICD 2 operation. Vss är jordanslutning.



Figur 5.3: Förenklad bild över kopplingarna mellan ICD 2 och PIC

5.1.4 Ethereal network protocol analyzer

”Analyzer” är ett gratisprogram [14] som har varit användbart för att titta på trafik när ethernetkortet varit anslutet till en PC. Det här programmet användes framför allt i början av projektet innan vi förstod hur allting fungerade och för förståelse över hur protokollen arbetar.

5.1.5 Hyperterminal

Hyperterminal är ett program inbyggt i Windows som använts att titta på information skickad genom en seriell anslutning från utvecklingskortet till en PC. Demoprogrammet som följde med Microchips TCP/IP stack använder sig av Hyperterminal för att ställa in IP-adress, Mask, Gateway och för att slå på/av DHCP. Det används nu i applikationen för att göra de initiala inställningarna.

5.1.6 MPFS.exe

MPFS är ett program som följer med Microchips TCP/IP-stack och som gör om HTTP-filer till en MPFS-image-fil, dvs. antingen kompilerbar C-kod som kan lägga in i projektet som en Source-fil eller en binär fil, som kan läggas i ett externt EEPROM. De filtyper som stöds är: TXT-, HTML-, GIF-, CGI-, JPG-, JAVA- samt WAV-filer. I projektet ingår enbart HTML-, CGI- och JPG-filer. HTML-filer är textfiler som kan läsas av en webbläsare och vad alla internetsidor består av. CGI-filer behövs när man vill kunna utföra något på dessa sidor, t.ex. skriva in något, trycka på en knapp etc. JPG-filer är bilder som ska visas på sidorna.

Exempel: *MPFS c:\Webbsidor MPFSimage.c /c*
Genom kommandot ovan görs en C-fil med namnet MPFSimage.c av de filer som ligger i mappen Webbsidor. C-filen placeras i den mapp där MPFS.exe-programmet ligger. Följande kommando ger en binär fil:
MPFS c:\Webbsidor MPFSimage.bin

5.1.7 Övrigt

Vi använde oss av våra egna bärbara datorer för att utföra examensarbetet. Detta gjorde att vi kunde sitta och arbeta var vi ville och att vi alltid hade allt med oss hem och kunde fortsätta utan onödigt krångel. Det gjorde det också enkelt att installera de program vi behövde använda.

5.2 Tillvägagångssätt

Denna del beskriver hur arbetet har lagts upp och en del av de val vi gjort under arbetets gång.

5.2.1 Förberedelser (Vecka 1-2)

Innan vi hade fått utrustningen valde vi att ta hem och studera datablad över PIC och ethernetkontroller samt ritningar över korten vi skulle använda. Tyvärr var ritningarna väldigt enkla och förklarade inte särkilt mycket, vilket hade blivit ett problem om vi hade varit tvungna att skriva en egen stack.

Vi kom alltså snabbt fram till att det skulle vara klart smart att använda sig av Microchips färdiga TCP/IP-stack, anpassad för användning av den utrustning vi skulle använda. Dock betydde det här att vi blev tvungna att studera stacken och dess kod väldigt noga för att kunna anpassa den till vårt ändamål. Det visade sig att med stacken följde en enkel HTTP-server-applikation, som vi kunde utgå ifrån för att skriva vår applikation. Vi började även läsa på en del generellt om TCP/IP, vilket var bra för att öka förståelsen för det vi höll på med, även om det hade behövts studeras betydligt mer om vi hade skrivit vår egen stack.

En annan sak vi behövde göra under förberedelserna var att bekanta oss med de program vi skulle använda, så att vi kunde sätta igång utan problem när vi fick utrustningen. Vi började med att ladda hem utvecklingsmiljön, MPLAB IDE, och titta igenom dess ”user’s guide”. Vi hittade också en onlinekurs för programmering av en PIC i IDE, som vi tittade en del på för att lära oss det grundläggande. C-kompilatorn, MPLAB C18, hade också en bra ”getting started” guide som vi gick igenom.

5.2.2 Test av utrustning (Vecka 2-3)

Andra veckan av arbetet fick vi vår utrustning och kunde börja testa den. Vi började med att löda det som behövde lödas: en plugg skulle lödas på Explorerkortet så att ethernetkortet kunde anslutas och så behövdes trådar lödas på för att kunna ansluta en spänningskälla, eftersom vi först inte hade någon adapter att plugga i.

Vi valde att först skriva några små testprogram för att kontrollera att allt fungerade och att vi förstod hur vi skulle gå tillväga. Det första vi testade var ett enkelt program som tände några lysdioder på Explorerkortet, vilket fungerade fint. Efter ytterligare lite småtester programmerade vi stacken med tillhörande demofil för att se om den fungerade.

5.2.3 Kodning och deltestning (Vecka 4-8)

När demokoden verkade fungera som den skulle började vi modifiera koden för att få de funktioner vi ville ha i programmet. Deltestningen har skett kontinuerligt för att se till så att alla funktioner fungerat innan vi lagt till någonting nytt.

Vi tyckte att det skulle vara smart att kunna ställa in adresser via webbsidan (HTML-sidan) istället för att behöva ansluta genom Hyperterminal varje gång. Därför började vi med att lägga till kod så att man kunde ställa in IP-adress, Nätmask och Default Gateway samt slå på och av DHCP genom webbsidan. Vi har sedan sett till så att funktionerna fungerar genom att ansluta till webbsidan och testa dem.

För att kunna implementera nya funktioner som ska visas på webbsidan har vi hela tiden varit tvungna skapa en ny MPFS-image-fil (en ny webbsida) och byta ut den mot den gamla, vilket betyder att vi också behövt lära oss en del HTML-kodning. Vi har under hela arbetet med kodningen lagt till och ändrat CGI-filer för att de funktioner vi lagt till i programmet ska kunna användas via webbsidan. Vi valde att lära oss HTML- och CGI-kodningen efter hand (då vi behövde det) och inte avsätta tid till att göra det.

Vi fortsatte sedan att lägga till funktioner som vi ville testa och se så de fungerade, bl a att kunna spara och hämta värden ur det interna och externa EEPROMet för att sedan visa dem på webbsidan. Testningen av detta utfördes genom att läsa EEPROMen för att se vad som ligger/sparas där samtidigt som vi tittat vad som kommer upp på webbsidan.

De sista veckorna handlade också mycket om finputsning av koden och utseendet på webbsidan samt rapportskrivning. För att förenkla användningen av programmet valde vi också att skriva en användarmanual till applikationen.

6. Beskrivning av applikationen

Den demoapplikation som vi har utvecklat använder sig av Microchips TCP/IP-stack för att upprätta kommunikation över ethernet. Stacken beskrivs utförligare i avsnitt 4. Stacken har en HTTP-server som vår applikation använder sig av för att läsa in kommandon från och skicka information till klienten. HTTP-servern följer protokollet HTTP 1.0 och stödjer filformaten txt, htm, gif, cgi, jpg, cla och wav.
Applikation är alltså en webbserver som ska svara på requests från en klients webbläsare. Applikationen lyssnar på TCP-port 80 efter ett request och svarar med att skicka en webbsida. Webbsidan kan vara dynamisk, dvs. den kan innehålla olika information beroende på vad PICen har för status. Möjligheten till dynamiska sidor fås genom att använda CGI- (Common Gateway Interface) script, vilket förklaras utförligare i avsnitt 6.4.1. Genom CGI-scriptet får applikationen ett kommando eller en begäran om information.

6.1 Gränssnittet mellan main-applikationen och HTTP-servern

Om vi utgår ifrån ett enkelt exempel där användaren har öppnat en sida med CGI script i sin webläsare, som då visar en tryckknapp som ska tända en lysdiod på HPC Explorer Board.


```
<html>
<body bgcolor="#FFFFFF">
<FORM METHOD=GET action=0>
    <input type=submit name=1
      value="Toggle LED1">
 </body>
</html>
```

Figur 6.1: CGI script i webbläsare

Användaren trycker på knappen vilket får till följd att webbläsaren skickar ett request till HTTP-servern enligt nedan.

GET /0?1=Toggle+LED1 HTTP/1.1
(+ olika headers)

Metoden är alltså GET, URI= /0?1=Toggle+LED1 och versionen är HTTP 1.1. Servern tar hand om requestet och lagrar värdena i URI i en tvådimensionell array som kallas argv. De olika värdena skiljs åt genom ’?’,’=’ och ’&’ tecken. ’+’ tecken tolkas som mellanrum.

Värdena lagras enligt följande:

argv[0]= 0
argv[1]= 1
argv[2]= Toggle LED1

Det första värdet i arrayen används för att urskilja vilken typ av kommando det är frågan om. Det kan t.ex. vara ett kommando för att påverka lysdioderna, konfigurera TCP/IP eller ställa in värden för automaten. Det andra värdet talar om vilket kommando som ska utföras, t.ex. vilken av lysdioderna som ska tändas, vilken inställning som ska göras o.s.v. Det sista värdet har i just det här exemplet ingen betydelse utan är bara det värde som stod på knappen. Om det istället hade rört sig om en TCP/IP-inställning skulle den kunna innehålla t.ex. en IP-adress.

Servern anropar nu funktionen HTTPExecCmd(BYTE** argv, BYTE argc), som finns i mainapplikationen. Tillsammans med anropet skickas arrayen argv och dessutom ett värde argc som talar om hur lång argv är. Mainapplikationen genomför då lämplig åtgärd beroende på vilket kommando den fick och lagrar sedan filnamnet på den sida som ska visas för användaren som ett svar på dennes knapptryckning i argv[0]. Servern svarar nu användaren med ett HTTP response följt av html-/cgi-texten från den fil som ska visas som svar på användarens kommado.

6.2 Användning av PICtailkortets EEPROM

Det externa EEPROMet på PICtailkortet kan användas som lagrings-utrymme för den webbsida som skall användas för kommunikationen över Internet mellan automaten och användaren. Alternativt kan webbsidan lagras i programminnet, men eftersom den kan kräva en hel del utrymme så är det en fördel om den kan sparas på EEPROMet. Detta frigör en stor del av programminnet, som då kan användas till annat. EEPROMet kan även användas för lagring av inställningar för TCP/IP-kommunikationen som t.ex. IP-adress, gateway-adress etc. En stor fördel med att spara sådana värden på EEPROMet är att det då blir lätt att återupprätta en anslutning efter en omstart eller ett strömavbrott. På EEPROMet går det även att spara andra värden som inte bör gå förlorade som t.ex. biljettpris, statistik mm.

Webbsidan som ska sparas på EEPROMet måste konverteras till en binär fil innnan den kan lagras. Detta görs med programmet MPFS, som beskrivs utförligare i avsnitt 5.1.6. När den binära filen skapas måste även det utrymme på EEPROMet som ska användas till lagring av inställningar med mera reserveras. Hur mycket utrymme som ska reserveras beror på vad det ska användas till och desto mer utrymme som reserveras desto mindre blir kvar för lagring av webbsidan.

6.3 Funktioner i applikationen

Via webbsidan kan flera olika processer utföras på mikrokontrollern. Det går att styra utgångar, läsa av värden på ingångar mm. De funktioner som implementerats använder sig av de komponenter som finns på HPC Explorer Board, t.ex. potentiometer, tryckknapp och lysdioder. Via webbsidan kan potentiometerns värde avläsas som ett heltalsvärde 0-1024. Tryckknappens värde kan också avläsas som 0 eller 1. Dessutom kan lysdiodernas status visas, d.v.s. om de är på eller ej. Det går även slå på och stänga av lysdioderna via sidan.

6.3.1 Klockan

Applikationen använder en extern oscillator på 32.768kHz som är monterad på HPC Explorer Board för att skapa en realtidsklocka. Oscillatorns frekvens gör den lämplig att användas som klocka eftersom den kan räkna upp ett 16-bitars register på två sekunder. Klockan använder sig av Timer1 på PICen för att räkna upp till en sekund. Timer1 har ett 16-bitars register som den räknar upp med ett på varje klockflank till dess att registret är fullt, då ett overflow genereras och registret börjar om på noll. Genom att låta registret börja räkna med värdet 8000h tar det en sekund innan ett overflow sker, vilket då skapar ett avbrott. Vid ett avbrott kallas avbrottsrutinen för klockan som räknar upp klockans värde.

Klockfunktionen håller reda på år, månad, dag, timmar, minuter och sekunder. Värdena lagras i RAM-minnet och tar upp 7 byte av minnet – två byte för år och en byte vardera för övriga. När applikationen startas upp blir klockan inställd med startvärden för år, månad och dag medan de övriga värdena börjar på noll. På webbsidan finns en funktion för att ställa in datum och tid vilket sker genom att det värde som skall användas skrivs in i ett textfält. Värdet skickas till applikationen, som omvandlar texten med siffror till ett värde och sedan kontrollerar om värdet är giltigt.

6.3.2 TCP/IP konfigurering

För att TCP/IP ska fungera krävs MAC-adress, IP-adress, gateway-adress och subnetmask. Dessa värden måste ställas in innan applikationen kan kommunicera över Internet. När applikationen startas första gången kommer förbestämda värden att användas. MAC-adressens värde kan inte ändras (bara ställas in i Hyperterminal) utan kommer hela tiden att vara detsamma. IP-adress, gateway-adress och subnetmask kan däremot ändras antingen manuellt eller också automatiskt med DHCP. Värdena sparas på det externa EEPROMet så att det vid en eventuell omstart går att hämta de inställningar man använde senast.

Inställningarna för TCP/IP kan göras på två sätt, antingen via webbsidan eller också med hjälp av en dators Hyperterminal. På webbsidan kan man se vilka värden applikationen använder sig av för tillfället samt se om DHCP är aktiverat eller inte. Vill man ändra IP-adress, gateway-adress eller subnetmask fyller man i det nya värdet i textfältet och skickar till applikationen. Värdet kommer som en textsträng som omvandlas till ett värde. Det nya värdet kommer att börja användas om det gick att omvandla till ett korrekt värde och sedan sparas till EEPROMet. Det finns också en funktion för att aktivera och avaktivera DHCP från webbsidan.

Inställningar för TCP/IP kan även göras från en dator med en ledig COM-port och Hyperterminal installerat. Datorns COM-port ansluts då till RS-232 anslutningen på HPC Explorer Board med en rak kabel. Hyperterminalen måste ställas in så att den använder 19200 bitar/s, 8 databitar, ingen paritet, en stoppbit och ingen flödeskontroll. Via hyperterminalen kan IP-adress, gateway-adress, subnet-mask och DHCP på/av ställas in. Om EEPROMet på ethernetkortet skall användas för lagring av webbsidan kan den laddas ned via Hyperterminalen med hjälp av Xmodem-protokollet som beskrivs i avsnitt 6.4.2.

6.3.3 Funktioner för automaten

Via webbsidan ska man även kunna göra olika (simulerade) inställningar för den automat som ska kontrolleras. I applikationen har användaren möjlighet att ställa in ett biljettpris och för att underlätta eventuell felsökning finns en loggfil som loggar olika händelser samt tidpunkten för respektive händelse. Om t.ex. IP-adressen skulle ändras så kommer det att synas i loggfilen. Nedan visas ett exempel på hur det kan se ut.

2006:05:12:14:00:00 Device was reset
2006:05:13:10:30:00 IP has changed

Automatens biljettpris kan ges ett värde mellan 0 och 65535. Vid start sätts värdet till 0 eller till det värde som finns sparat i EEPROMet. Skulle ett felaktigt värde matas in kommer ingen ändring av priset att ske.
Loggfilen som skapas är från början tom men kan innehålla tidigare loggar om det finns sparat i EEPROMet. Loggfilen är i denna demoversion begränsad till maximalt 50st loggar eftersom det kräver en hel del utrymme. En sparad händelse kräver ett datum, en tid och en kod som talar om vilken typ av händelse som inträffade. Detta innebär att det behövs 8 byte för en loggning och totalt 400 byte för hela loggfilen. Nedan visas hur en loggning sparas och hur stor del respektive värde tar i anspråk.

Datum, År
Datum, År
Datum, Månad
Datum, Dag
Klockslag, Timmar
Klockslag, Minuter
Klockslag, Sekunder
Kod för typ av händelse

Eftersom koden för en händelse är en byte, kan man använda sig av 256 olika typer av händelser. När loggfilen presenteras för användaren omvandlas koden till en text som förklarar vad koden betyder. Det går också att tömma loggfilen genom att trycka på ”reset log”, vilket tar bort alla gamla loggar och lägger till en ny logg som talar om att loggen rensades. De händelser som skapar en logg är om IP-adressen ändras, DHCP slås av eller på, applikationen startas om, loggfilen rensades eller om testknappen på webbsidan trycktes.

6.3.4 Mynträknare

I applikationen finns en funktion som simulerar antal mynt stoppad i en automat. Funktionen räknar antal enkronor, femkronor och tiokronor som stoppas i. Eftersom det bara finns en knapp på utvecklingskortet som kan användas har en lösning gjorts som använder den aktuella inställningen på Explorerkortets potentiometer. Beroende på vad potentiometern står inställd på så räknas enkronor, femkronor och tiokronor upp för varje tryck på knapp RB0.

Den analoga potentiometerns inställning har gjorts om till ett digitalt värde mellan 0 och 1024. Vid en inställning på 100 eller mindre räknas antalet enkronor upp vid knapptryck. En inställning på mellan 100 och 900 ger en uppräkning av antalet femkronor och en inställning på 900 eller mer ger en uppräkning av antalet tiokronor. På det här sättet hittas rätt inställning på potentiometern enkelt, även när man inte ser det aktuella digitala värdet, genom att man skruvar den till min, max eller låter den stå någonstans i mitten.

Denna funktion fungerar också som ett test av hur PICens interna EEPROM kan användas. De aktuella värdena för respektive myntsort lagras i det interna EEPROMet, som annars är oanvänt. Det innebär att om en reset av programmet av någon anledning skulle inträffa, så läses de aktuella värdena in från EEPROMet.

När det interna EEPROMet skrivs till måste en speciell ordning av kommandon följas. Följande kodavsnitt visar ett exempel på detta:

```
EEADRH = 0x00;              //
EEADR = 0x07;               // - minnesplats 0007h
EEDATA = PressRB0Low10;     //Data som ska skrivas in
EECON1 = 0b00000100;        //Aktivera write to EEPROM
INTCON_GIEH = 0;            //Inaktivera avbrott under skrivning
EECON2 = 0x55;              //Initiera write
EECON2 = 0x0AA;             //...
EECON1bits.WR = 1;          //...
INTCON_GIEH = 1;            //aktivera avbrott
EECON1bits.WREN = 0;        //inaktivera write enable
```

Vilken minnesplats i EEPROMet som data ska skrivas till anges genom att lägga den höga byten av adressen i EEADRH och den låga byten i EEADR. Det värde som ska skrivas till EEPROMet läggs i EEDATA-registret.

För att aktivera skrivning till EEPROM måste sedan vissa bitar i EECON1-registret ändras. EECON1-registret (se Figur 6.2) är kontrollregistret för minnesaccesser.

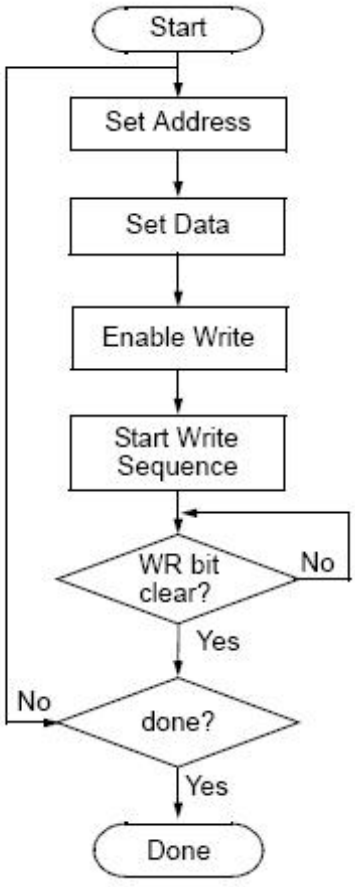
EECON1: EEPROM CONTROL REGISTER 1

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

Figur 6.2: Registret EECON1 [23]

De bitar i EECON1-registret som måste ändras från default är bit 7 (EEPGD) som sätts till ’0’, vilket anger EEPROM-access och bit 2 (WREN = Write Enable) som sätts till ’1’, vilket enablar skrivning.

INTCON är ett register som styr avbrott. Under skrivning till EEPROMet bör inga avbrott tillåtas. Genom att sätta GIEH-biten i INTCON-registret till ’0’ förhindras alla avbrott. EECON2 är inget fysiskt register. Det används endast vid skrivning och rensning av minnen. För att initiera skrivning till EEPROMet läggs först 0x55 och sedan 0x0AA i EECON2. Nu kan skrivningen startas. Detta görs genom att sätta bit 1 (WR) i EECON1-registret till ’1’. Skrivningen utförs på en instruktion. Sedan kan avbrott aktiveras igen och Write Enable slås av. Figur 6.3 visar ett flödesdiagram för skrivning till det interna EEPROMet.



Figur 6.3: EEPROM-programmeringsflöde

För att slippa skriva alla rader kod varje gång en skrivning till det interna EEPROMet görs ligger det i en egen funktion, WriteInternalEEPROM( ).

Läsning av det interna EEPROMet är betydligt enklare. Adressen till minnesplatsen anges på samma sätt som vid en skrivning. Sedan sätts EECON1-registret likadant, med den skillnaden att bit 0 (RD = Read) sätts till ’1’ istället för bit 1. Det lästa värdet ligger sedan i EEDATA-registret. Här följer ett kodexempel för läsning av det interna EEPROMet:

```
EEADRH = 0x00;                  //
EEADR = 0x07;                   // - minnesposition 0007h
EECON1 = 0b00000001;            //Starta läsning av EEPROM
PressRB0Low10 = EEDATA;         //Spara data i variabel
```

6.4 Protokoll som används av applikationen

6.4.1 CGI script

[45]

CGI är ett protokoll som används för att kommunicera mellan ett webbformulär och ett program. Webbformuläret visas i användarens webbläsare och programmet körs på webbservern. I vårt fall är programmet den demoapplikation som vi har skapat. Webbformulären är som vanliga HTML-sidor där användaren kan göra olika val, skriva in text, trycka på olika knappar med mera. Användarens val eller text skickas som en sträng till programmet som körs på servern. Flera olika val och texter kan skickas samtidigt och hamnar då i samma sträng. De olika värdena skiljs i så fall åt genom att de har olika namn.

```
<FORM METHOD=GET action=2>
<input type=text name=08 value=”Put text here”><input type=submit value=Apply>
</FORM>
```

Dessa tre rader skapar ett textfält i vilket valfri text kan skrivas och sedan skickas genom att trycka på retur eller genom att klicka på den knapp som ligger efter textfältet med texten ”Apply”. Nedan visas hur det ser ut i webbläsaren.

Om användaren nu skriver in sin text i rutan, t.ex. ”Mytext” och trycker på ”Apply” kommer följande sträng att skickas till servern: ”/2?08=Mytext”. I strängen är de olika parametrarna åtskiljda med '?'- och '=' -tecken. Första värdet anger vilken ”action” det är fråga om. Detta kan vara olika beroende på vad formuläret innehåller. Nästa värde är ”name” och används för att skilja på olika värden i samma formulär. Efter ett ”name” följer det värde som användaren ville skicka, vilket i det här fallet är en sträng med text. Programmet som körs på servern tar emot och tolkar strängen som skickats och ser till att rätt funktion utförs.

6.4.2 Xmodem-protokollet

[46]

Xmodem-protokollet används för nedladdning av image-filer till EEPROMet. En nedladdning initieras av användaren genom att via hyperterminalen välja alternativet ”Download MPFS Image”, vilket resulterar i att PICen skickar NAK till hyperterminalen till dess att den svarar med att skicka ett paket enligt figur nedan. Datadelen är alltid 128 Byte och skulle den vara mindre fylls den ut. När ett paket skickats till PICen kontrolleras checksumman. Om allt stämmer skickas ett ACK tillbaka till hyperterminalen och om något är fel skickas ett NAK. Om hyperterminalen fick ett ACK tillbaka fortsätter den med att skicka ut nästa block med data om det finns fler. Om däremot ett NAK togs emot skickas samma block igen. När alla block är skickade sänds ett EOT av hyperterminalen vilket besvaras med att ACK från PICen som därmed avslutar överföringen.(5)

SOH	01H	Start of Header
EOT	04H	End of transmission
ACK	06H	Acknowledge
NAK	15H	Not Acknowledge
CAN	18H	Cancel

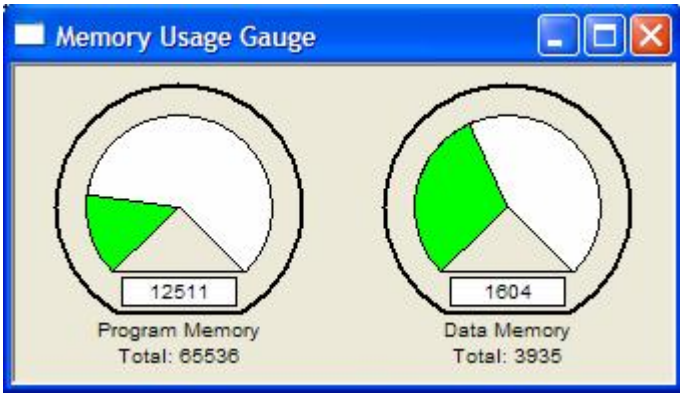
SOH	Block nr	255 – Block nr	128 Bytes Data	Checksum

7. Resultat

Arbetet ledde fram till en fungerande demoapplikation som klarar av att kommunicera över TCP/IP. Kommunikationen har genom tester visat sig fungera i såväl mindre lokala nätverk som över internet. Med hjälp av en webbläsare kan man ansluta sig till demokortet och utföra diverse funktioner samt få information om status mm. Demoapplikationen är skapad för demokortet och kommer att behöva anpassas för att kunna integreras i en automat. De delar av applikationen som används för själva kommunikationen är dock färdiga att använda och kommer inte att behöva någon förändring. Den TCP/IP stack från Microchip som vi utgick från har underlättat arbetet betydligt och lett till att vi kunnat koncentrera oss på att utveckla själva applikationen.

7.1 Krav för funktion

Hela stacken och applikationen kräver (Se Figur 7.1) 1604 Bytes dataminne (RAM) och ca 24,4 kB (12511 instruktioner = 25022 bytes = 24,4 kB) programminne under förutsättning att MPFS-imagen (webbsidan) lagras externt i EEPROM. Då programminnet är på 128 kB så finns det gott om utrymme kvar för att lägga till det som behövs när applikationen skall användas i automaten. Valet finns också att lägga MPFS-imagen i programminnet. Det skulle då kräva totalt ca 55,1 kB utrymme i programminnet.



Figur 7.1: Default minnesanvändning (Programminne i instruktioner (2 bytes), dataminne i bytes)

Default-imagen (MPFSImg.c) innehåller en JPG-bild som kräver mycket minne. För att spara minne kan MPFSImg2.c användas istället. För att göra det måste MPFSImg.c tas bort från projektet och MPFSImg2.c läggas till. Detta sparar ungefär 20 kB minne, vilket är en hel del.

7.2 Kända problem

Demoapplikationen är inte helt fri från problem. Det har under tester visat sig att stacken ibland kan låsa sig och en omstart krävs för att få igång den igen. Efter en titt på Microchips forum så har vi sett att många andra har detta problem också – där kommunikationen helt plötsligt slutar fungera och en omstart är det enda som kan åtgärda det – vilket betyder att det med största sannolikhet beror på Microchips TCP/IP-stack. Förhoppningsvis åtgärdas detta till nästa version av stacken.

En sak som man bör veta är att RB0-knappen på Explorerkortet inte verkar ligga helt stabil. Ibland när den trycks in registreras mer än ett tryck. Detta är inget som påverkar funktionaliteten men är bra att känna till.

8. Slutsats

Möjligheterna till kommunikation över TCP/IP för en PIC-mikrokontroller kan anses som mycket goda, det som utgör en begränsning är dels storlek på programminne och dataminne samt vilka möjligheter PICen har för att kommunicera med ethernetkontrollern. Denna demoapplikation klarar av 10Mbps ethernet, för att kunna erhålla högre hastigheter krävs snabbare mikrokontroller och ethernetkontroller. För de uppgifter applikationen skall klara kommer dock inte hastigheten att orsaka något större hinder.

Vi tycker att vi har uppnått de mål vi ställde i början av projektet och hoppas att Sterners kommer att kunna dra nytta av vårat arbete.

9. Problem under arbetet

Detta beskriver problem som funnits under arbetet. De beskrivs för att de ska kunna undvikas om arbetet fortsätts av Sterners eller om någon annan gör något liknande och använder detta arbete som hjälp.

- Trots att Microchips TCP/IP-stack var anpassad till att använda vår utrustning hade inte beskrivningen av stacken uppdaterats, utan vi fick gå efter den [27] som är skriven för PICDEM.net-kortet. Detta medförde att flera saker tog betydligt längre tid än det behövde. Bland annat hade vi svårt att få fram något i Hyperterminalen, så vi kunde slå av DHCP för att få kommunikationen mellan PC och ethernetkort att fungera. Detta berodde på att man var tvungen att hålla inne en knapp på Explorerkortet när demoprogrammet i PICen körde igång, vilket inte stod någonstans – all beskrivning var ju för ett helt annat kort.
- Det fungerade inte alltid att programmera och sedan köra programmet när vi körde MPLAB ICD 2 i Debug mode. Vad detta berodde på vet vi inte. Det gav dock inte några större problem. När det inträffade så programmerade vi med ICD 2 i Program mode istället.
- Ofta när vi laddade ned en MPFS-image i det externa EEPROMet ville inte Setupmenyn i Hyperterminal komma upp igen – så vi kunde spara inställningarna – efter att nedladdningen var klar. Detta avhjälptes genom att bara koppla ifrån anslutningen och ansluta igen, och sedan spara.
- Vi testade implementera en sleepfunktion för ethernetkortet för att spara ström, vilket inte gick eftersom kortet då helt slutade ta emot paket via nätverksporten.

10. Vidareutveckling

Vidareutveckling tar upp några saker som skulle förbättra applikationen och som verkligen är möjliga att lägga till.

- Eftersom vi inte vill att vår applikation ska vara åtkomlig för utomstående skulle det behövas någon form av inloggningsmetod för att få tillgång till sidan. Detta skulle kunna lösas genom att servern vid början av en kommunikation begär ett användarnamn och ett lösenord.
- En bra funktion för att underlätta vid en oförutsedd omstart, skulle vara om realtidsklockan kunde ställa in sig själv då den får kontakt med Internet.
- En spännande möjlighet skulle vara att implementera en SMTP-modul. Den skulle kunna göra så applikationen skickar ett mail till användaren om något har blivit fel inne i maskinen.
- En FTP-modul skulle kunna läggas till, så att det vore möjligt att hämta hela filer från PICen till webbsidan.

11. Källförteckning

[10] Programvaror

[11]. MPLAB C18 Student Edition
http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB-C18-Student-Edition-no-doc-v3_02.exe

[12]. MPLAB IDE
http://ww1.microchip.com/downloads/en/DeviceDoc/MP731_Full.zip

[13]. TCP/IP Stack Software (ENC28J60 driver)
http://ww1.microchip.com/downloads/en/DeviceDoc/MCHPStackENC8722v302.zip

[14]. Network analyzer
http://www.ethereal.com/distribution/win32/ethereal-setup-0.99.0.exe

[20] Datablad, scheman och manualer

[21]. Microchip Technology Inc. (2005),
MPLAB ICD 2 User's Guide, DS51331B

[22]. Microchip Technology Inc. (2005), *PIC18F8722 Family, Flash Microcontroller Programming Specification*
http://ww1.microchip.com/downloads/en/DeviceDoc/39643B.pdf

[23]. Microchip Technology Inc. (2004),
PIC 18F8722 Family Data Sheet 64/80-Pin, 1-Mbit, Enhanced Flash Microcontrollers with 10-bit A/D and nanoWatt Technology
http://ww1.microchip.com/downloads/en/DeviceDoc/39646b.pdf

[24]. Microchip Technology Inc. (2004), *MPLAB® C18 C Compiler Getting Started*
http://ww1.microchip.com/downloads/en/DeviceDoc/MPLAB_C18_Getting_Started_51295f.pdf

[25]. Microchip Technology Inc. (2004), *ENC28J60 Data Sheet*
http://ww1.microchip.com/downloads/en/DeviceDoc/39662a.pdf

[26]. Humberd S (2005), *PICDEM HPC Board Schematics,*
Microchip Technology Inc.
http://ww1.microchip.com/downloads/en/DeviceDoc/PICDEM_HPC_SCHEM_03-01822_REV5.pdf

[27]. Rajbharti, Nilesh (2002), *AN833 The Microchip TCP/IP Stack*
Microchip Technology Inc.
http://ww1.microchip.com/downloads/en/AppNotes/00833b.pdf

[28]. Microchip Technology Inc. (2005), *25AA256/25LC256 256K SPI™ Bus Serial EEPROM Data Sheet*
http://ww1.microchip.com/downloads/en/DeviceDoc/21822E.pdf

[30] TCP/IP information

[31]. Stevens, W. Richard (1994), *TCP/IP Illustrated Volume 1: The Protocols*
Addison Wesley proffessional computing series, ISBN 0-201-63346-9

[32]. CISCO Networking Academy Programs (2003), *CCNA 1: Networking Basics*
http://cisco.netacad.net (kräver inloggning)

[33]. CISCO Networking Academy Programs (2003), *CCNA 2: Routers and Routing Basics*
http://cisco.netacad.net (kräver inloggning)

[40] Övrig information

[41]. Examensarbete vid Högskolan Dalarna (2000-02-08), *Information till studenter*
http://www2.du.se/projekt/144/dokument/student04.doc?iProjektId=144

[42]. Examensarbete vid Högskolan Dalarna (2000-01-23), *Anvisningar för rapporter i examensarbeten*
http://www2.du.se/projekt/144/dokument/rapportanvisningar.PDF?iProjektId=144

[43]. McDonough, John (2003-2005), *PIC Elmer 160 Course Lessons*
http://www.amqrp.org/elmer160/lessons/

[44]. M.A.Smith University of Brighton (1995-1999), *Tags for creating forms in HTML*
http://snowwhite.it.brighton.ac.uk/~mas/mas/courses/html/html2.html#FORM-Hidden

[45]. Marshall, James (1996-2002), *CGI Made really easy*
http://www.jmarshall.com/easy/cgi/

[46]. Christensen, Ward (1982-01-01), *Modem Protocol Documentation*
http://www.textfiles.com/apple/xmodem

Användarmanual

Följande utgör en användarmanual för användning av vår applikation. En förenklad engelsk version kommer att ligga på webbsidan för enkel åtkomst.

1. Kom igång

Innan du lagt in applikationen i din PIC-mikrokontroller måste du välja om du vill lägga MPFS-imagen (webbsidan) i ett externt EEPROM (som i detta fall finns på ethernetkortet) eller i PICens programminne. Om du väljer att lägga MPFS-imagen i programminnet måste du lägga till Source-filen ”MPFSImg.c” till MPLAB-projektet (”Sterners.mcp”). Sedan programmerar du PICen med detta projekt. Du kan då hoppa över avsnitt 2.

Väljer du istället att lägga imagen i EEPROMet programmerar du PICen med samma projekt, men utan ”MPFSImg.c”. Du måste då också ladda ner imagen till EEPROMet med hjälp av Hyperterminal, vilket förklaras i avsnitt 2.

Notering: Vi rekommenderar att MPFS-imagen läggs i ett externt EEPROM då detta sparar mycket programminne.

För att ställa in ethernetkortets grundinställningar och för att lägga en MPFS-image i det externa EEPROMet måste du upprätta en seriell anslutning mellan din PC och PICen. Anslut bara en rak RS-232 kabel mellan PCn och kortet du använder. Öppna sedan programmet Hyperterminal.

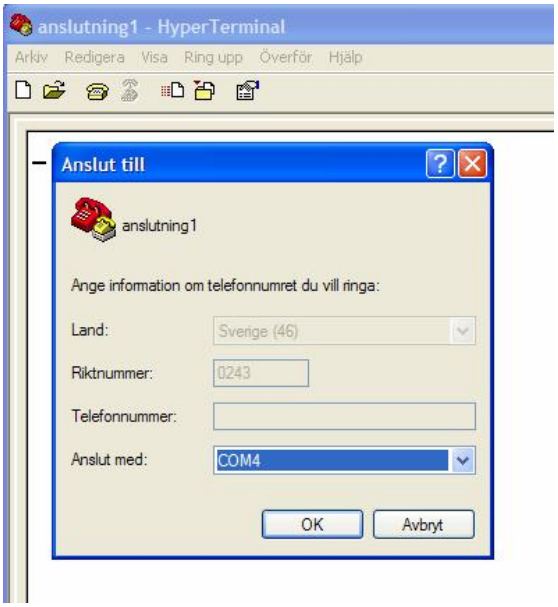
Notering: Hyperterminal hittar du i Windowsbaserade PCs under Start -> Program -> Tillbehör -> Kommunikation -> Hyperterminal.

Starta en Hyperterminal-anslutning:

Steg 1. Starta en ny anslutning

Välj först valfritt namn på din anslutning.

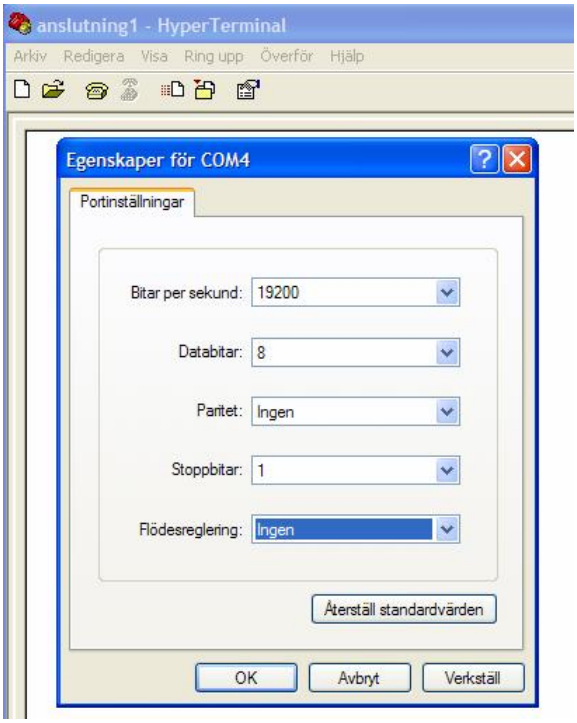
Tryck sedan OK.



Steg 2. Välj port att ansluta genom

Ange i listan ”Anslut med” vilken com-port din dator är ansluten till kortet med.

Tryck OK.



Steg 3. Egenskaper för COM-porten

Korrekta inställningar är:

- 19200 bitar per sekund (Baud Rate)
- 8 Databitar
- Ingen paritet.
- 1 stoppbit
- Ingen flödesreglering (flow control)

Tryck OK.

För att få fram menyn i Hyperterminal resetar du programmet med knapp MCLR (Master clear), håller in den, trycker in knapp RB0 och släpper sedan upp resetknappen. Du ska nu ha följande meny i Hyperterminalen:

```
MCHPStack Config Application (MCHPStack 3.02, May 13 2006)

        1: Change Board serial number.
        2: Change default IP address.
        3: Change default gateway address.
        4: Change default subnet mask.
        5: Enable DHCP.
        6: Disable DHCP.
        7: Download MPFS image.
        8: Save & Quit.


Enter a menu choice (1-8):
```

Den här Setupmenyn använder du för att göra grundinställningarna. Tryck 1 för att ställa in ethernetkortets serienummer. Detta hittar du generellt på en klisterlapp på undersidan av ethernetkortet. Genom att lägga in detta serienummer ger du ditt ethernetkort en unik MAC-adress, eftersom serienumret är detsamma som MAC-adressens fyra sista siffror.

Med alternativen 2-4 kan du ställa in en stationär IP-adress, gateway-adress och subnet mask för att kunna ansluta till ethernetkontrollern via TCP/IP. Genom alternativen 5-6 kan du slå på/av (enable/disable) DHCP. DHCP är ett protokoll som automatiskt sätter IP-adresser till kända MAC-adresser och måste vara disable om du inte kopplar in ethernetkortet i ett nätverk som sätter IP-adresser dynamiskt eller om du inte lagt ethernetkortets MAC-adress i DHCP-listan, annars kommer du inte att kunna ansluta till kortet. Sätter du DHCP till disable kommer ethernetkortet få den stationära IP-adress du angett.

Notering: DHCP är default enabled.

Alternativ 7 använder du för att lägga in en MPFS-image i ethernetkortets EEPROM. Läs om det i nästa avsnitt.

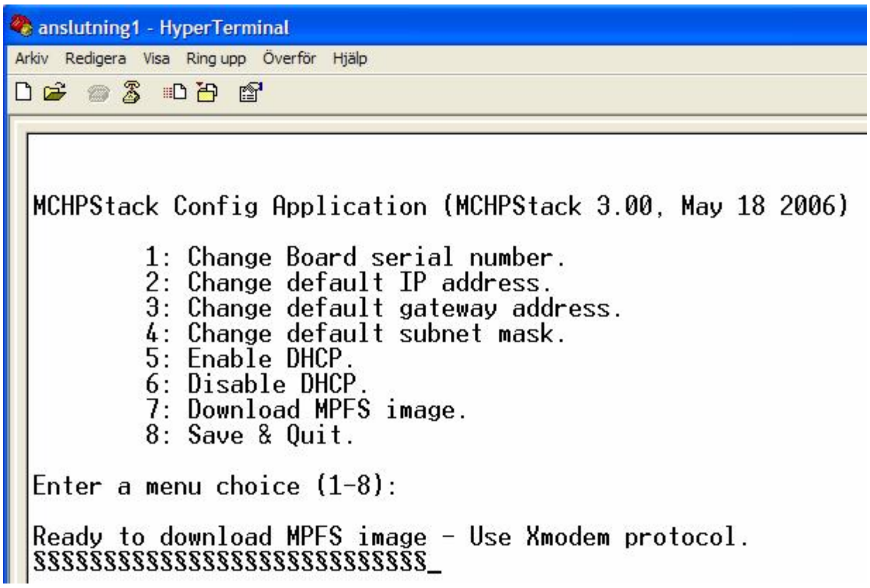
Kom ihåg att spara inställningarna genom att trycka 8 när du är färdig. Då kommer också applikationen att köra igång igen. Om du ändrat IP-adress eller ändrat DHCP enable/disable kommer kortets nya IP-adress att synas i Hyperterminalfönstret när du sparat inställningarna.

Notering: När du gjort de grundinställningar du vill ha i Hyperterminal behöver du inte använda den igen. IP-adress, default gateway, subnet mask och DHCP-inställning kan sedan enkelt ändras via webbsidan.

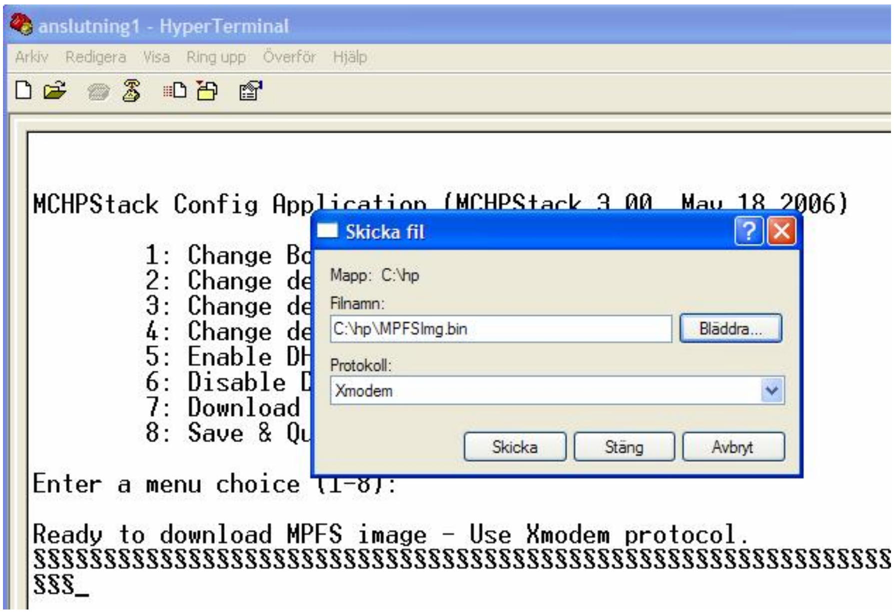
2. Att lägga MPFS-imagen i ett externt EEPROM

Om du väljer att lägga MPFS-imagen i ett externt EEPROM, måste du använda Hyperterminal för att ladda in imagen i EEPROMet. Detta går till så här:

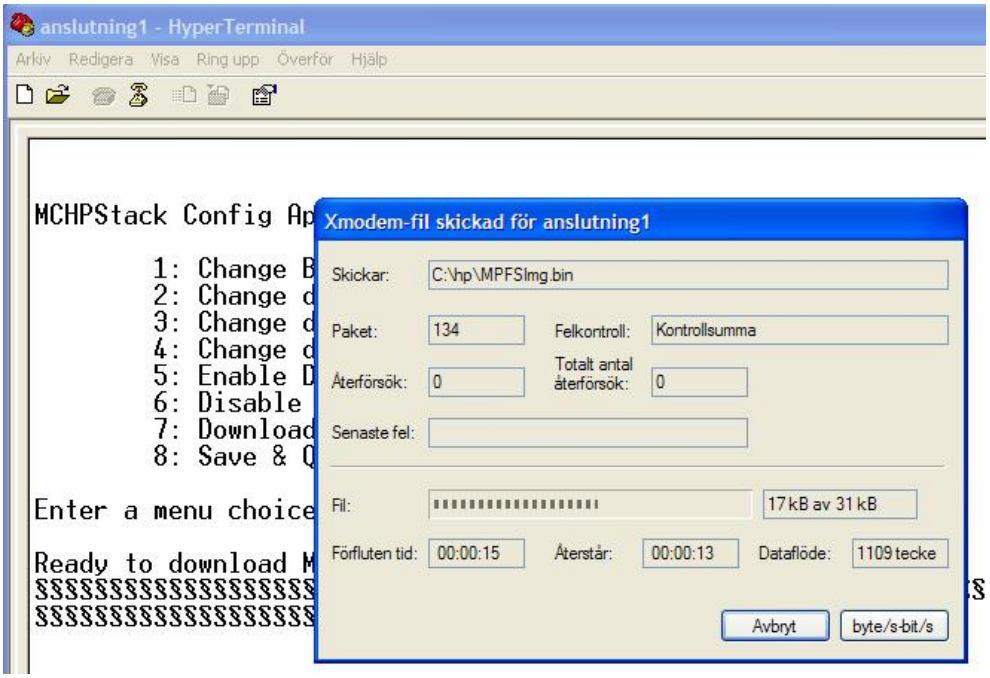
Välj alternativ sju (Download MPFS image) i Setupmenyn. När du gjort det kommer du att se följande:



§-tecken skrivs ut i Hyperterminalfönstet, vilket indikerar att du kan börja ladda ned imagen. Gör detta genom att gå till ”Skicka fil” genom Överför => Skicka fil. Du bör nu se detta i Hyperterminalförnstret:



Under ”Filnamn” skriver du in sökvägen till den .bin-fil som är din MPFS-image. Xmodem är det protokoll som ska användas vid överföringen. Tryck på sedan på ”Skicka”. Nu bör du se följande i Hyperterminalfönstret:

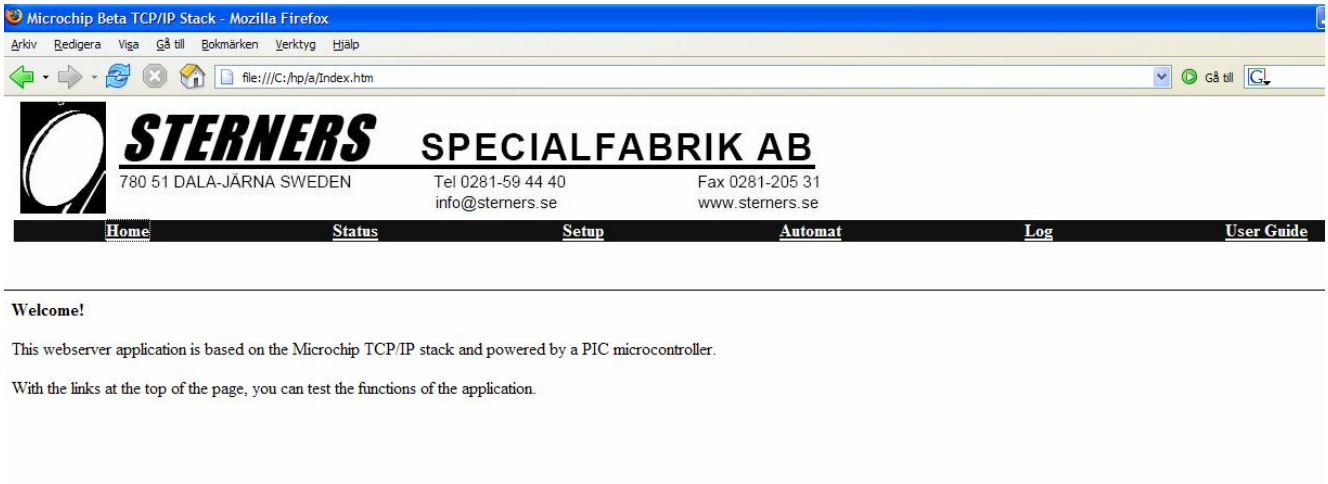


Du kan här se statistik för överföringen, bland annat överföringshastigheten. När överföringen är klar kommer du tillbaka till Setupmenyn. Där kan du spara dina inställningar genom alternativ åtta.

Notering: Ibland kan Hyperterminal hänga sig efter överföringen av MPFS-imagen. Om detta skulle hända, koppla ifrån och anslut igen genom att trycka på ”koppla ifrån” och sedan ”ring upp”. Tryck sedan Enter på tangentbordet så ska Setupmenyn komma upp igen och du kan spara dina inställningar.

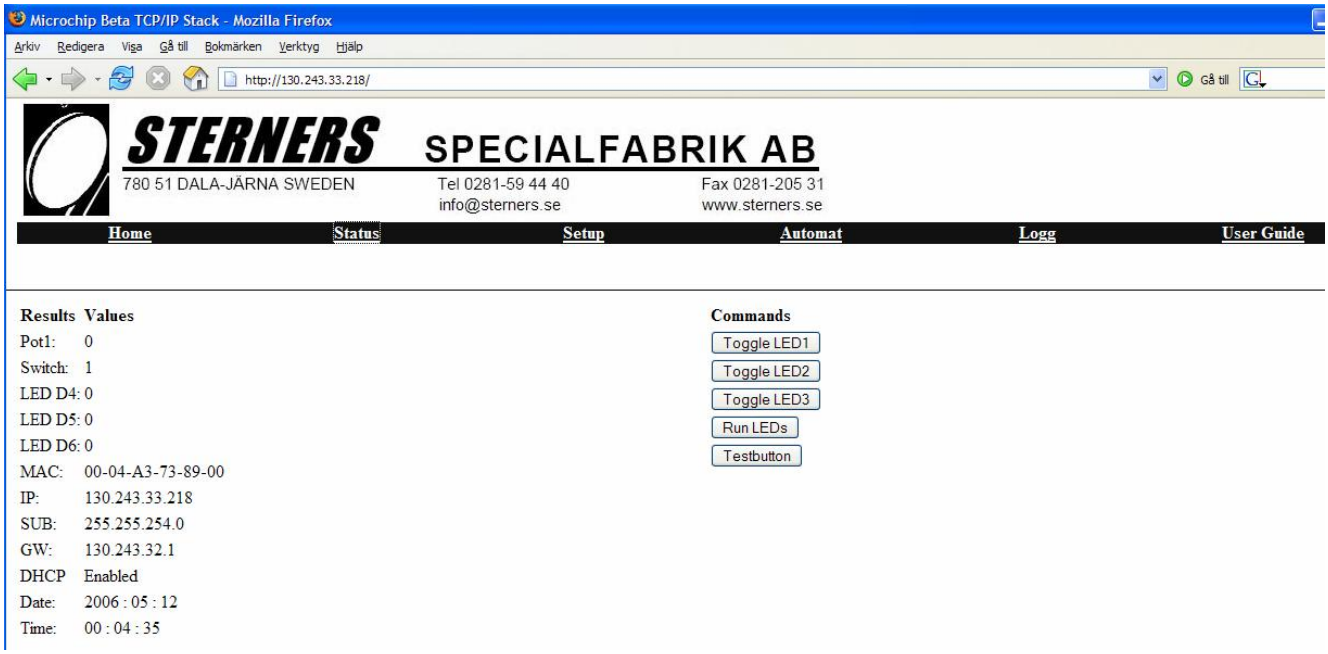
3. Applikationen

Du kan nu ansluta genom att koppla en vanlig nätverkskabel mellan kortet och ditt nätverk. Öppna en webbläsare och skriv in den IP-adress ethernetkortet har fått. Du ska nu se den här sidan i din webbläsare:



Detta är startsidan. Med länkarna högst upp på sidan kommer du åt de funktioner som finns i applikationen. Genom att trycka på ”Hem” återgår du till startsidan. Här följer en genomgång av alla funktioner sida för sida. Under ”User Guide” hittar en förkortad version av denna genomgång.

3.1 Statussidan

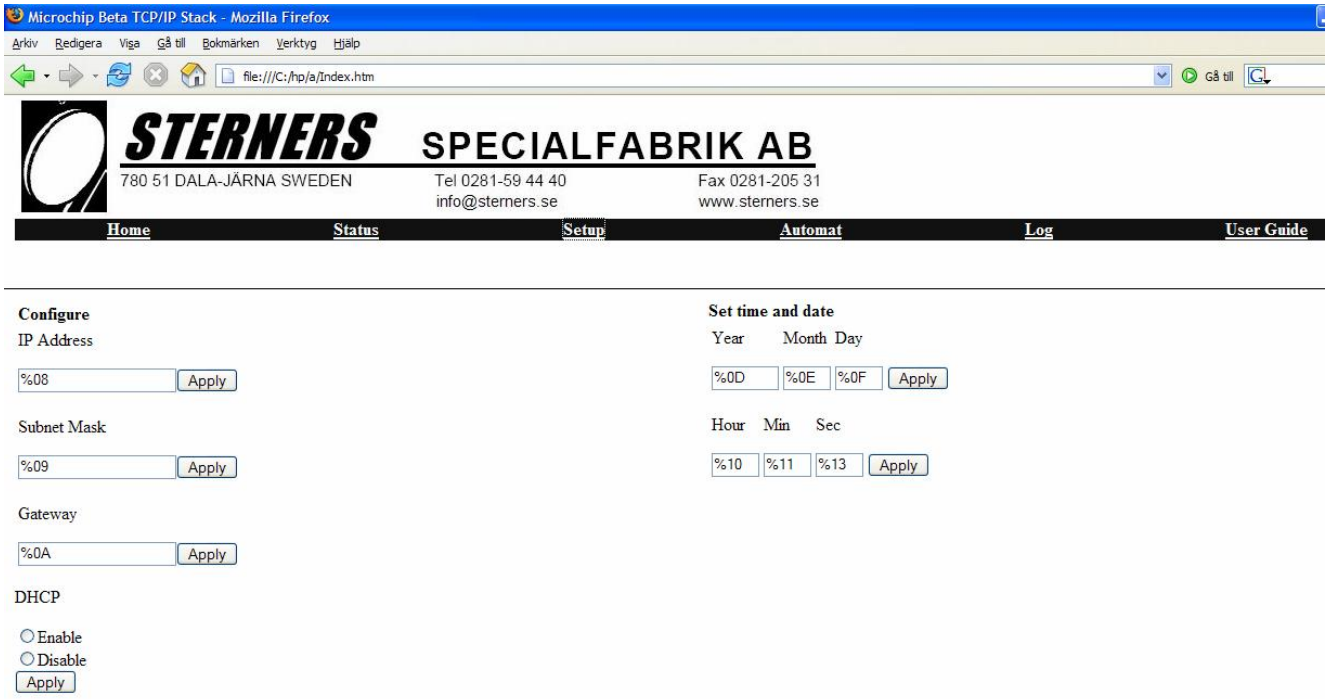


På Statussidan kan du till vänster se statusen för explorerkortet. De här är de funktioner du kan kolla status för:

- Pot – Visar det aktuella värdet på potentiometern som finns på explorerkortet.
- Switch – Visar en nolla om knapp RB0 är intryckt, annars ’1’.
- LED 1,2,3 – Visar en etta om LED D4, D5 respektive D6 är aktiverade (lyser), annars 0.
- MAC – Visar ethernetkortets aktuella MAC-adress.
- IP – Visar ethernetkortets aktuella IP-adress. (Tänk på att om DHCP är aktiverat så visas inte den statiska IP-adress du angett.)
- SUB – Visar aktuell Subnet mask.
- GW – Visar aktuell Default gateway adress.
- DHCP – Visar DHCP ’Enabled’ eller ’Disabled’.
- Date – Visar aktuellt datum (om du ställt in datumet på setupsidan).
- Time – Visar aktuell tid (om du ställt in tiden på setupsidan).

På Statussidan kan du också testa några enkla LED-funktioner. Klicka på en knapp för att tända respektive LED eller klicka på ”RUN LEDS” för att tända och släcka alla LEDs stegvis. Knappen ”Testbutton” finns för att testa så att loggen fungerar. Tryck på knappen och gå till Logsidan och kolla så att knapptryckningen har registrerats och lagts till i loggen.

3.2 Setupsidan

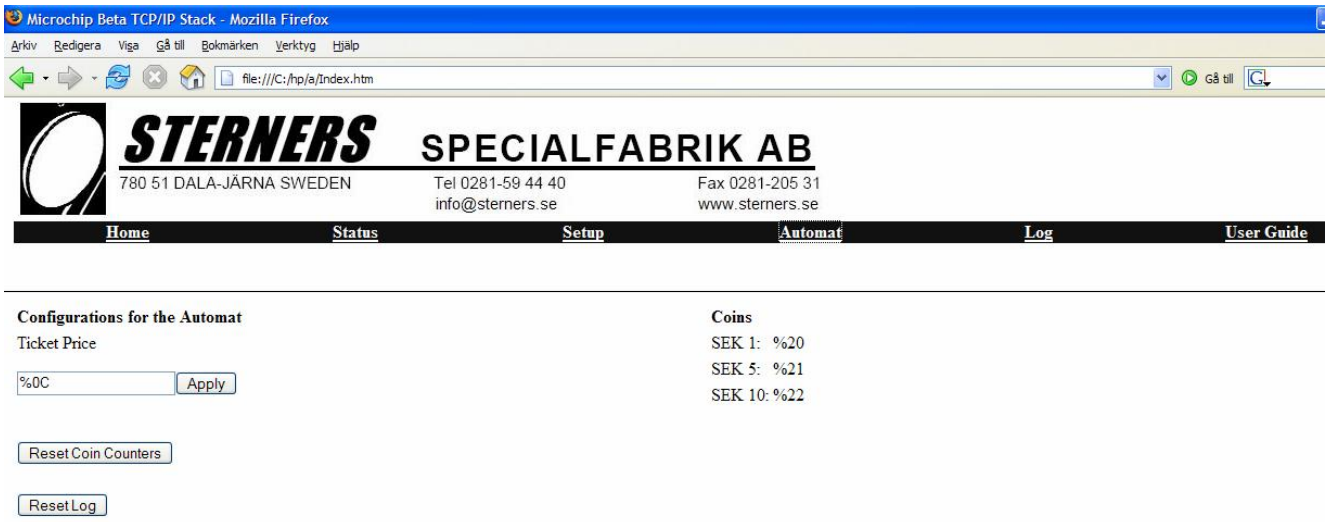


På Setupsidan kan du ändra IP-adress, Default gateway adress, Subnet mask och slå på/av DHCP.

För att ändra t.ex. IP-adress skriver du in den IP-adress du vill ha i textrutan och trycker på ”Apply”-knappen till höger om rutan. Skriv t.ex. 130.30.35.3 => tryck ”Apply”, så blir det din nya statiska IP-adress. Kom ihåg att slå av DHCP om du vill använda den statiska adressen. DHCP sätter du enkelt till enable/disable genom att klicka för ettdera alternativet och sedan trycka på ”Apply”.

På Setupsidan kan du också ställa in datum och tid. Skriv bara in dagens datum (t.ex. 2006 10 20) och tryck på ”Apply”. Samma sak gäller för inställning av aktuell tid.

3.3 Automatsidan



På Automatsidan hittar du några funktioner som ska simulera händelser på en automat. Du hittar en simulering av en ändring av biljettpris. Genom att skriva in en siffra i textrutan under ”Ticket Price” skriver du över det gamla värdet i det externa EEPROMet. Det aktuella värdet (priset) visas i textrutan. Det maximala värde som kan anges är 65535.

”Coins” simulerar antal mynt som stoppas i en automat. SEK 1, 5 och 10 står för enkronor, femkronor respektive tiokronor. Detta sker genom tryck på knapp RB0 på Explorerkortet och beroende på vad potentiometern har för värde. Har potentiometern ett värde på 100 eller mindre, dvs. den står på min, och knappen trycks in räknas antalet enkronor upp. Antalet femkronor räknas upp om potentiometern har ett värde mellan 100 och 900 och antalet tiokronor räknas upp vid ett värde på 900 eller mer, dvs. den står på max (max = 1024).

Antalet mynt adderas hela tiden och sparas i PICens interna EEPROM. Antal enkronor tar upp två minnesplatser (två bytes) i EEPROMet, vilket betyder att det maximala värdet som kan sparas är 65535. Femkronor och tiokronor tar upp en minnesplats var, vilket betyder maximalt 255 av varje valör. Du kan när som helst ansluta till webbsidan och se hur många knapptryckningar som gjorts för respektive inställning på potentiometern (dvs. hur många mynt som lagts i automaten). De aktuella värdena läses från EEPROMet och visas här på Automatsidan.

På Automatsidan kan du också nollställa både räkningen av antalet mynt, genom att trycka på knappen ”Reset Coin Counters”, och loggen (se nästa avsnitt) genom att trycka på knappen ”Reset Log”.

3.4 Logsidan

På Logsidan finns en till funktion som simulerar händelser på en automat. Funktionen här simulerar en händelselogg. Varje gång det händer något (t.ex. DHCP enable/disable) skrivs detta ut här tillsammans med datum och tid då detta hände. Med denna funktion är det meningen att det ska gå att felsöka och se vad som gjordes då det blev fel.

Loggarna sparas i det externa EEPROMet. 50 logghändelser kan sparas i den här versionen av applikationen.