Mini Rapport

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Kungliga Tekniska Högskolan

Johan N. & Erik B.

Maj 2017

IE1206 Inbyggd Elektronik

**Innehållsförteckning**

1 **Mini Rapport**  **2**

* 1. Titel på projektet . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  2. Beskrivning av tänkt ändamål . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  3. Beskrivning av prototyp . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  4. Diagram . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
  5. Kriterier för utfört projekt . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
  6. Publicerad kod . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  7. Fritzing . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
  8. Bilagor . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

1. **Mini Rapport**

1.1 **Titel på Projektet**

PIC Smartcard Klippkort

* 1. **Beskrivning av tänkt ändamål**

Produkt är ett klippkortssystem till ett tivoli där antalet åkturer för en person sparas med hjälp av ett smartcard. Varje gång en besökare tar en åktur så dras en åktur av med hjälp av en kortläsare som är monterad vid grinden.

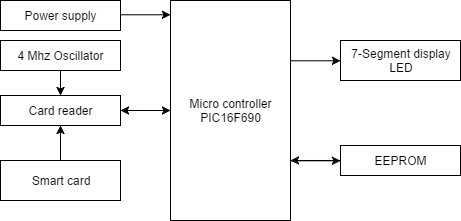
1.3 **Beskrivning av prototyp**

Programmet som är installerat på kortet skickar ut ett ID varje gång det sätts in i en kortläsare. Detta ID tas emot av en PIC-processor från kortläsaren och antalet åkturer för det ID minskas med ett. Alla värden och ID sparas i flashminnet på mikroprocessorn och inte på korten.

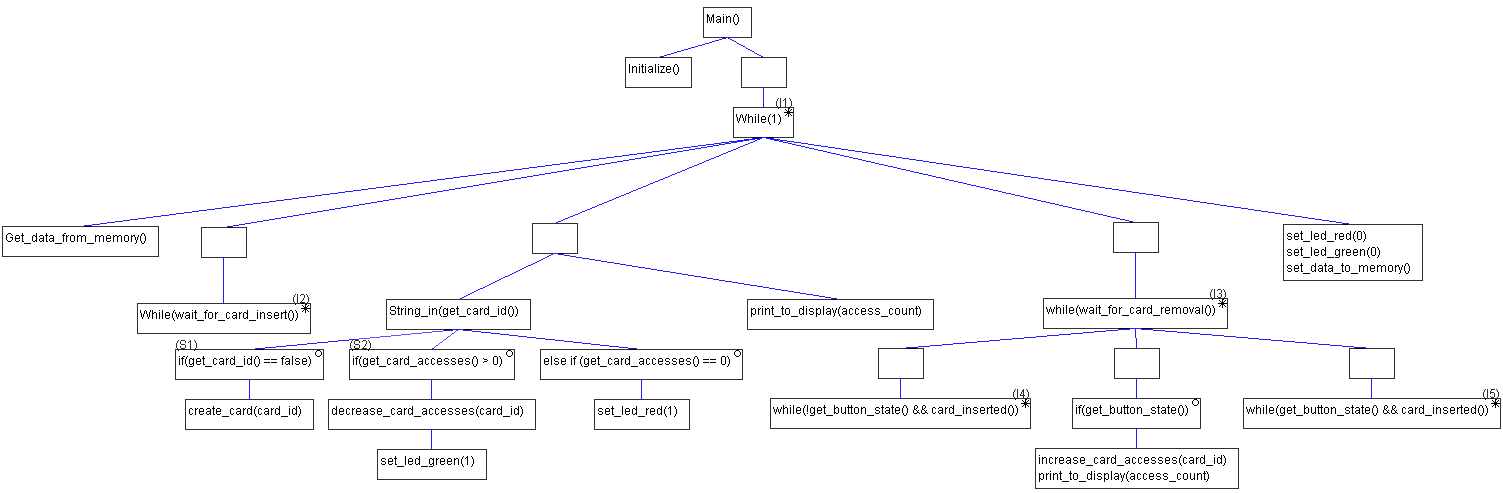
En 4 MHz kristall är kopplad till kortläsaren som kommunicerar med processorn. Processorn i sin tur kommunicerar med flashminnet och skriver/hämtar data under programmets gång. Antalet åkturer som är kvar visas sedan på en 7-segments display. Detta kan sedan kopplas till ytterligare komponenter som t.ex grind för ingång till en åkattraktion.

Med nuvarande processor finns det plats för två kort men det kan lätt ökas om processorn byts ut till en med mer RAM-minne.

1.4 **Diagram**



Figur 1: Blockdiagram



Figur 2: Strukturdiagram

JSP diagrammet som visas är till processorns Main funktion. Först initialiseras alla värden som krävs och sedan börjar Main loopen som hela programmet körs i. All data (kort ID och antal åkturer) hämtas från EPROM sedan. Koden väntar den på att kortet sätts i och att ett ID ska skickas från kortet. När det är mottaget kollar processorn hur många åkturer det kortet har och drar bort en om det finns mer än noll. Efter det finns möjligheten att fylla på kortet med mer åkturer genom att klicka på en knapp. Detta är dock bara för att testa funktionalitet och är inte del av kretskortet som levereras eftersom användaren inte är tillåten att fylla på sitt kort själv. Sedan sparas all data till EPROM minnet och Main loopen startas om.

1.5 **Kriterier för utfört projekt**

Eftersom kommunikation sker mellan olika delar och inte bara inom processorn så är det extra viktigt att testa så att programmet och delarna fungerar som det är tänkt. När följande krav är uppfyllda är programmet i princip felfritt.

* Antal åkturer för ett ID minskas med ett varje gång kortet sätts i
* Ovanstående funktion fungerar felfritt vid alternering med olika kort
* Inget fel uppstår om man sätter in ett kort uppochned
* Om ett kort inte finns i flashminnet så läggs detta till och kan användas
* Om strömmen stängs av under användning ska flashminnet fortfarande visa de senast uppdaterade värdena som skrevs in innan det

Alla dessa punkter är uppfyllda och därför anses programmet som fullt fungerande.

1.6 **Publicerad kod**

Koden till vårt projekt finner ni i följande GitHub repository.

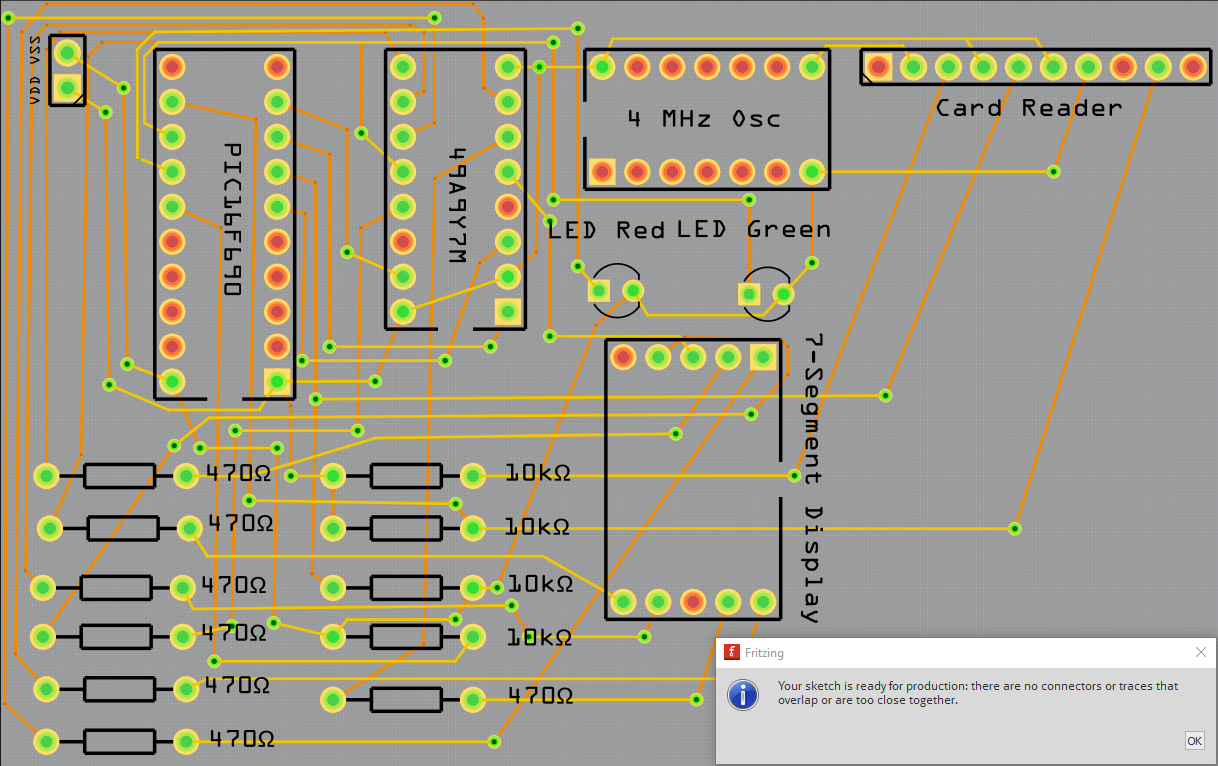
<https://github.com/vonNiklasson/KTH-smartcard>

En kopia av koden och tillhörande ASCII grafik finner ni bland bilagorna.

Kompilatorn som används är CC5X Knudsen. Den finner ni här:

<http://www.bknd.com/cc5x/>

1.7 **Fritzing**



Figur 2: PCB Diagram med Design Rules Check

1.8 **Bilagor**

Kod till processorn, smartcardet och ett testprogram finns här.

Om inget annat anges är koden skriven av oss.

**ASCII grafik**

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\* HARDWARE \*/

/\* \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* \*/

/\*

Use "PICkit2 UART Tool" as a 9600 Baud terminal to save data.

with BitBanging routines.

\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

| \/ |

+5V---|Vdd 16F690 Vss|---GND

|RA5 RA0/(PGD)|bbTx ->- PK2Rx/PGD

|RA4/AN3 AN1/REF/RA1/(PGC)|------<- PGC

|RA3/!MCLR/(Vpp) RA2/AN2/INT|-<- U

|RC5/CCP RC0|->- LED

|RC4 RC1|

|RC3 RC2|

|RC6 RB4|

|RC7 RB5/Rx|

|RB7/Tx RB6|-<- SW

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

SmartCard contact

\_\_ \_\_ \_\_

+5V |C1| C5| Gnd

|\_\_| \_\_|

MCLR |C2| |C6|

|\_\_| |\_\_|

OSC |C3| |C7| RB7/PGD I/O -><- Txd/Rxd half duplex

|\_\_| |\_\_|

|C4| |C8|

|\_\_|\_\_|\_\_|

Card Reader

\_\_\_\_\_\_

|C4 |

|CLK |<--- 4MHz Osc

|RST |

|VCC |

|SW2 |

|SW1 |

|GND |<--- GND

|VPP |<--- +5V

|I/O |<--- I/O

|C8 |

|\_\_\_\_|

4MHz Oscillator

\_\_\_\_\_\_\_\_\_\_\_

GND --->|GND VDD|<--- +5V

| |

| |

| |

| |

| CLK out|<--- Clk out

|\_\_\_\_\_\_\_\_\_|

7-Segment display

\_\_\_\_\_\_\_\_\_\_\_\_

|b h|

|a c|

|VDD VDD|

|f d|

|g e|

|\_\_\_\_\_\_\_\_\_\_\_|

L293B 7-segment driver

\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_

Pulse --->|Clock \/ VCC|<--- +5V

--->|Disable Reset|<---

--->|En. disp. Not Out|

|En. Out Out c|<--- c

GND --->|GDN Out b|<--- b

f --->|Out f Out e|<--- e

g --->|Out g Out a|<--- a

GND --->|VDD Out d|<--- d

|\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_|

\*/

**Processor**

/\* This file contains the main

\* function and the program logic \*/

/\* First include of chipkit-header \*/

#include "16F690.h"

#pragma config |= 0x00D4

#define MAX\_STRING 16

#define NEW\_ACCESS\_COUNT 5

/\* Allocate space for 7 cards \*/

char memory\_cards**[**8 **\*** 2**];**

char memory\_card\_count**;**

char create\_card**(**char **\*** card\_id**);**

void get\_data\_from\_memory**(**void**);**

void set\_data\_to\_memory**(**void**);**

char get\_card\_offset**(**char **\*** card\_id**);**

char get\_card\_accesses**(**char card\_offset**);**

void set\_card\_accesses**(**char card\_offset**,** const char accesses**);**

char increase\_card\_accesses**(**char card\_offset**);**

char decrease\_card\_accesses**(**char card\_offset**);**

void reg\_put\_char**(**char data**,** char EEPROMadress**);**

char reg\_get\_char**(**char EEPROMadress**);**

void reg\_put\_word**(**const char **\*** word**,** char reg\_offset**);**

void reg\_get\_word**(**char **\*** word**,** char reg\_offset**);**

/\*String related functions\*/

void put\_char**(**char d\_out**);**

char get\_char**(**void**);**

void string\_in**(**char **\*** string**);**

void string\_out**(**const char **\*** string**);**

bit compare\_string**(**char **\*** input\_string**,** const char **\*** candidate\_string**);**

/\*Hardware related functions\*/

void wait\_for\_card\_insert**(**void**);**

void wait\_for\_card\_withdraw**(**void**);**

bit get\_button\_state**(**void**);**

void set\_led\_red**(**bit state**);**

void set\_led\_green**(**bit state**);**

void print\_to\_display**(**char val**);**

void delay**(**char millisec**);**

/\*Initializing and registry clearing\*/

void initialize**(**void**);**

void overrun\_recover**(**void**);**

void main**(**void**)** **{**

/\* String to store text from card \*/

char card\_str**[**MAX\_STRING**];**

char card\_offset**;**

char card\_access\_count**;**

bit has\_access **=** 0**;**

bit test1 **=** 0**;**

bit test2 **=** 0**;**

/\* Initialize some code \*/

initialize**();**

/\* Extended initialize \*/

memory\_cards**[**0**]** **=** 0**;**

memory\_card\_count **=** 0**;**

/\* Loop forever, program logic below \*/

**while** **(**1**)** **{**

/\* Reset the display \*/

print\_to\_display**(-**1**);**

get\_data\_from\_memory**();**

/\* Wait for card insertion \*/

**while** **(**PORTC.3 **==** 0**);**

delay**(**100**);** /\* card debounce \*/

delay**(**50**);** /\* extra delay \*/

/\* ask the question \*/

string\_out**(**"Send the ID please\r\n"**);**

delay**(**100**);** /\* USART is buffered, so wait until all chars sent \*/

/\* empty the reciever FIFO, it's now full with garbage \*/

overrun\_recover**();**

/\* Get id from card, stored in card\_str \*/

string\_in**(&**card\_str**[**0**]);**

/\* Get the card offset id (if it exists) \*/

card\_offset **=** get\_card\_offset**(&**card\_str**[**0**]);**

**if** **(**card\_offset **==** **-**1**)** **{**

//Add new card

card\_offset **=** create\_card**(&**card\_str**[**0**]);**

**}**

card\_access\_count **=** get\_card\_accesses**(**card\_offset**);**

nop**();**

//Check if any accesses are left on the card and makes you refill

**if** **(**card\_access\_count **>** 0**)** **{**

set\_led\_green**(**1**);**

nop**();**

// Decrease the number of accesses

card\_access\_count **=** decrease\_card\_accesses**(**card\_offset**);**

**}**

**else** **if** **(**card\_access\_count **==** 0**)** **{**

set\_led\_red**(**1**);**

nop**();**

**}**

print\_to\_display**(**card\_access\_count**);**

//While card is inserted loop the

// ability to increase the number of access

**while** **(**PORTC.3 **==** 1**)** **{**

//Wait for button presses and add 1 access

**while** **(!**get\_button\_state**()** **&&** PORTC.3 **==** 1**);**

/\* If the button is pressed, increase the access count \*/

**if** **(**get\_button\_state**())** **{**

card\_access\_count **=** increase\_card\_accesses**(**card\_offset**);**

nop**();**

print\_to\_display**(**card\_access\_count**);**

**}**

/\* Wait for debounce of the button \*/

**while** **(**get\_button\_state**()** **&&** PORTC.3 **==** 1**);**

**}**

delay**(**10**);**

/\* Reset the LED:s \*/

set\_led\_red**(**0**);**

set\_led\_green**(**0**);**

delay**(**100**);** /\* card debounce \*/

/\* Write the new information to the memory \*/

set\_data\_to\_memory**();**

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTIONS

=========

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

char get\_card\_offset**(**char **\*** card\_id**)** **{**

int i**,** j**,** k**;**

char tmpChar1**,** tmpChar2**;**

/\* Loops through the number of cards \*/

**for** **(**i **=** 0**;** i **<** memory\_card\_count**;** i**++)** **{**

/\* Sets j to the card offset \*/

j **=** i **\*** 8**;**

/\* Iterate the chars in the card strings \*/

**for** **(**k **=** 0**;** k **<** 7**;** k**++)** **{**

/\* Check if the string matches \*/

tmpChar1 **=** memory\_cards**[**j **+** k**];**

tmpChar2 **=** card\_id**[**k**];**

**if** **(**tmpChar1 **!=** tmpChar2**)** **{**

**break;**

**}**

**}**

/\* If all chars matched, return the offset id \*/

**if** **(**k **==** 7**)** **{**

**return** i**;**

**}**

**}**

/\* Otherwise, return -1 \*/

**return** **-**1**;**

**}**

//Get the number of accesses of a specific card

char get\_card\_accesses**(**char card\_offset**)** **{**

char current\_accesses **=** memory\_cards**[(**card\_offset **\*** 8**)** **+** 7**];**

**return** current\_accesses**;**

**}**

//Get the number of accesses of a specific card

void set\_card\_accesses**(**char card\_offset**,** const char accesses**)** **{**

memory\_cards**[(**card\_offset **\*** 8**)** **+** 7**]** **=** accesses**;**

**}**

//Increase the number of accesses of a specific card by one

char increase\_card\_accesses**(**char card\_offset**)** **{**

char current\_access **=** get\_card\_accesses**(**card\_offset**);**

**if** **(**current\_access **<** 9**)** **{**

current\_access**++;**

**}**

set\_card\_accesses**(**card\_offset**,** current\_access**);**

**return** current\_access**;**

**}**

//Decrease the number of accesses of a specific card by one

char decrease\_card\_accesses**(**char card\_offset**)** **{**

char temp\_offset **=** card\_offset**;**

memory\_cards**[(**temp\_offset **\*** 8**)** **+** 7**]** **=** memory\_cards**[(**temp\_offset **\*** 8**)** **+** 7**]** **-** 1**;**

**return** memory\_cards**[(**temp\_offset **\*** 8**)** **+** 7**];**

**}**

char create\_card**(**char **\*** card\_id**)** **{**

/\* Get the new card offset \*/

char card\_offset **=** memory\_card\_count**;**

/\* Add 1 to the memory card count \*/

memory\_card\_count **=** memory\_card\_count **+** 1**;**

int i**;**

char temp\_char**;**

**for** **(**i **=** 0**;** i **<** 7**;** i**++)** **{**

temp\_char **=** card\_id**[**i**];**

memory\_cards**[(**8 **\*** card\_offset**)** **+** i**]** **=** temp\_char**;**

**}**

/\* Set the last byte to 0 (8 \* card\_offset) + i + 1 \*/

memory\_cards**[(**8 **\*** card\_offset**)** **+** i**]** **=** 0**;**

/\* Return the new card offset \*/

**return** card\_offset**;**

**}**

void get\_data\_from\_memory**(**void**)** **{**

/\* Get how many cards that are saved in the memory \*/

memory\_card\_count **=** reg\_get\_char**(**0**);**

/\* Temporay string for card data \*/

char card**[**8**];**

/\* Initialize temp vars \*/

int i**,** j**,** k**;**

char temp\_char**;**

/\* Count for how many cards that are stored \*/

**for** **(**i **=** 0**;** i **<** memory\_card\_count**;** i**++)** **{**

/\* Get card i from memory \*/

reg\_get\_word**(&**card**[**0**],** i**);**

/\* Get start offset for card in local string \*/

k **=** i **\*** 8**;**

/\* Loop through the next 8 bytes in the local string \*/

**for** **(**j **=** 0**;** j **<** 8**;** j**++)** **{**

temp\_char **=** card**[**j**];**

memory\_cards**[**k **+** j**]** **=** temp\_char**;**

**}**

**}**

**}**

void set\_data\_to\_memory**(**void**)** **{**

/\* Store the number of saved cards \*/

reg\_put\_char**(**memory\_card\_count**,** 0**);**

/\* Initialize temp vars \*/

char i**;**

/\* Count for how many cards that are stored \*/

**for** **(**i **=** 0**;** i **<** memory\_card\_count**;** i**++)** **{**

reg\_put\_word**(&**memory\_cards**[**0**],** i**);**

**}**

**}**

void reg\_put\_word**(**const char **\*** word**,** char reg\_offset**)** **{**

int offset **=** **(**reg\_offset **\*** 8**)** **+** 1**;**

char c**;**

int i**;**

**for** **(**i **=** 0**;** i **<** 8**;** i**++)** **{**

c **=** word**[**offset **-** 1 **+** i**];**

reg\_put\_char**(**c**,** offset **+** i**);**

**}**

**}**

void reg\_get\_word**(**char **\*** word**,** char reg\_offset**)** **{**

int offset **=** **(**reg\_offset **\*** 8**)** **+** 1**;**

char c**;**

int i**;**

**for** **(**i **=** 0**;** i **<** 8**;** i**++)** **{**

c **=** reg\_get\_char**(**offset **+** i**);**

word**[**i**]** **=** c**;**

**}**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void reg\_put\_char**(**char data**,** char EEPROMadress**)** **{**

/\* Put char in specific EEPROM-adress \*/

/\* Write EEPROM-data sequence \*/

EEADR **=** EEPROMadress**;** /\* EEPROM-data adress 0x00 => 0x40 \*/

EEDATA **=** data**;** /\* data to be written \*/

WREN **=** 1**;** /\* write enable \*/

EECON2 **=** 0x55**;** /\* first Byte in comandsequence \*/

EECON2 **=** 0xAA**;** /\* second Byte in comandsequence \*/

WR **=** 1**;** /\* write \*/

**while(** EEIF **==** 0**)** **;** /\* wait for done (EEIF=1) \*/

WR **=** 0**;**

WREN **=** 0**;** /\* write disable - safety first \*/

EEIF **=** 0**;** /\* Reset EEIF bit in software \*/

/\* End of write EEPROM-data sequence \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

char reg\_get\_char**(**char EEPROMadress**)** **{**

/\* Get char from specific EEPROM-adress \*/

/\* Start of read EEPROM-data sequence \*/

char temp**;**

EEADR **=** EEPROMadress**;** /\* EEPROM-data adress 0x00 => 0x40 \*/

EEPGD **=** 0**;**

RD **=** 1**;** /\* Read \*/

temp **=** EEDATA**;**

RD **=** 0**;**

**return** temp**;** /\* data to be read \*/

/\* End of read EEPROM-data sequence \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

/\* Sends one char \*/

void put\_char**(**char d\_out**)** **{**

**while** **(!**TXIF**);** /\* wait until previus character transmitted \*/

TXREG **=** d\_out**;**

**return;** /\* done \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

/\* Recieves one char \*/

char get\_char**(**void**)** **{**

char d\_in **=** '\r'**;**

**while** **(!**RCIF **&&** PORTC.3**);** /\* wait for character or card removal \*/

**if(!**RCIF**)** **return** d\_in**;**

d\_in **=** RCREG**;**

**return** d\_in**;**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void string\_in**(**char **\*** string**)** **{**

char charCount**,** c**;**

**for(**charCount **=** 0**;** **;** charCount**++)** **{**

c **=** get\_char**();** /\* input 1 character \*/

string**[**charCount**]** **=** c**;** /\* store the character \*/

// put\_char( c ); /\* don't echo the character \*/

/\* end of input \*/

**if((**charCount **==** **(**MAX\_STRING**-**1**))** **||** **(**c**==**'\r'**))** **{**

string**[**charCount**]** **=** '\0'**;** /\* add "end of string" \*/

**return;**

**}**

**}**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void string\_out**(**const char **\*** string**)** **{**

char i**,** k**;**

**for(**i **=** 0 **;** **;** i**++)** **{**

k **=** string**[**i**];**

**if(** k **==** '\0'**)** **return;** /\* found end of string \*/

put\_char**(**k**);**

**}**

**return;**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

bit compare\_string**(**char **\*** input\_string**,** const char **\*** candidate\_string**)** **{**

/\* compares input with the candidate string \*/

char i**,** c**,** d**;**

**for(**i**=**0**;** **;** i**++)** **{**

c **=** input\_string**[**i**];**

d **=** candidate\_string**[**i**];**

**if(**d **!=** c **)** **return** 0**;** /\* no match \*/

**if(**d **==** '\0'**)** **return** 1**;** /\* exact match \*/

**}**

**}**

/\* Stall program til card is inserted \*/

void wait\_for\_card\_insert**(**void**)** **{**

**while** **(**PORTC.3 **==** 0**);**

**}**

/\* Stall program til card is withdrawn \*/

void wait\_for\_card\_withdraw**(**void**)** **{**

**while** **(**PORTC.3 **==** 1**);**

**}**

bit get\_button\_state**(**void**)** **{**

bit input**;**

**while(**1**)** **{**

input **=** PORTC.1**;**

delay**(**10**);**

**if** **(**input **==** PORTC.1**)** **{**

**return** input**;**

**}**

**}**

**}**

void set\_led\_red**(**bit state**)** **{**

PORTA.2 **=** state**;**

nop**();**

**}**

void set\_led\_green**(**bit state**)** **{**

PORTB.4 **=** state**;**

nop**();**

**}**

void print\_to\_display**(**char val**)** **{**

nop**();**

/\* Print hex-value to 7-segment display \*/

char value **=** val**;**

char i**;**

**if** **(**value **==** **-**1**)** **{**

PORTC.4 **=** 0**;**

nop**();**

**}**

**else** **if** **(**value **>=** 0 **&&** value **<** 10**)** **{**

PORTC.4 **=** 1**;**

nop**();**

delay**(**1**);**

PORTC.7 **=** 1**;**

nop**();**

delay**(**1**);**

PORTC.7 **=** 0**;**

nop**();**

delay**(**1**);**

**for** **(**i **=** 0**;** i **<** value**;** i**++)** **{**

PORTC.6 **=** 1**;**

nop**();**

delay**(**1**);**

PORTC.6 **=** 0**;**

nop**();**

delay**(**1**);**

**}**

**}**

**}**

/\* Delays a multiple of 1 milliseconds at 4 MHz

using the TMR0 timer by B. Knudsen \*/

void delay**(**char millisec**)** **{**

OPTION **=** 2**;** /\* prescaler divide by 8 \*/

**do** **{**

TMR0 **=** 0**;**

**while** **(**TMR0 **<** 125**);** /\* 125 \* 8 = 1000 \*/

**}** **while** **(--**millisec **>** 0**);**

**}**

void initialize**(**void**)** **{**

TRISA.0 **=** 1**;** /\* RA0 not to disturb PK2 UART Tool \*/

ANSEL.0 **=** 0**;** /\* RA0 digital input \*/

TRISA.1 **=** 1**;** /\* RA1 not to disturb PK2 UART Tool \*/

ANSEL.1 **=** 0**;** /\* RA1 digital input \*/

/\* Initialize PIC16F690 serialcom port \*/

/\* One start bit, one stop bit, 8 data bit, no parity. 9600 Baud. \*/

TXEN **=** 1**;** /\* transmit enable \*/

SYNC **=** 0**;** /\* asynchronous operation \*/

TX9 **=** 0**;** /\* 8 bit transmission \*/

SPEN **=** 1**;**

BRGH **=** 0**;** /\* settings for 6800 Baud \*/

BRG16 **=** 1**;** /\* @ 4 MHz-clock frequency \*/

SPBRG **=** 25**;**

CREN **=** 1**;** /\* Continuous receive \*/

RX9 **=** 0**;** /\* 8 bit reception \*/

ANSELH.3 **=** 0**;** /\* RB5 digital input for serial\_in \*/

/\* More init \*/

TRISC.3 **=** 1**;** /\* RC3 card contact is input \*/

ANSEL.7 **=** 0**;** /\* RC3 digital input \*/

TRISA.2 **=** 0**;** /\* RC2 Red LED for no access \*/

PORTA.2 **=** 0**;** /\* RC2 initially off \*/

TRISB.4 **=** 0**;** /\* RB4 Green LED for access \*/

PORTB.4 **=** 0**;** /\* RB4 initially off \*/

TRISC.1 **=** 1**;** /\* RC1 Button is set to input \*/

ANSEL.5 **=** 0**;** /\* RC1 digital input \*/

//Display initialize

TRISC.7 **=** 0**;** /\* RC7 Output to reset pin on display \*/

PORTC.7 **=** 0**;** /\* RC7 initially off \*/

TRISC.6 **=** 0**;** /\* RC6 Output to display clock \*/

PORTC.6 **=** 0**;** /\* RC6 initially off \*/

TRISC.4 **=** 0**;** /\* RC4 Enable display \*/

PORTC.4 **=** 1**;** /\* RC4 on \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void overrun\_recover**(**void**)** **{**

char trash**;**

trash **=** RCREG**;**

trash **=** RCREG**;**

CREN **=** 0**;**

CREN **=** 1**;**

**}**

**Smartcard**

/\* main\_key\_template.c question and answer, compare strings \*/

/\* This program is for 16F84 Gold Card \*/

/\*

SmartCard contact

\_\_ \_\_ \_\_

+5V |C1| C5| Gnd

|\_\_| \_\_|

MCLR |C2| |C6|

|\_\_| |\_\_|

OSC |C3| |C7| RB7/PGD I/O -><- Txd/Rxd half duplex

|\_\_| |\_\_|

|C4| |C8|

|\_\_|\_\_|\_\_|

\*/

/\*

SERIAL COMMUNICATION

============================

One start bit, one stop bit, 8 data bit, no parity = 10 bit.

Baudrate: 9600 baud => 104.167 usec. per bit.

serial output PORTB.7 half duplex!

serial input PORTB.7 half duplex!

\*/

#include "16F84.h"

#define MAX\_STRING 16 /\* string input max 15 characthers \*/

#pragma config |= 0x3ff1

/\* Function prototypes \*/

void initserial**(** void **);**

void putchar**(** char **);**

char getchar**(** void **);**

void string\_out**(** const char **\*** string **);**

void string\_in**(** char **\*** **);**

bit check\_candidate**(** char **\*** input\_string**,** const char **\*** candidate\_string **);**

void delay**(** char **);**

void main**(** void**)**

**{**

char i**,** c**,** d**,** charCount**;**

char input\_string**[**MAX\_STRING**];** /\* 15 char buffer for input string \*/

bit compare**;**

delay**(**50**);** /\* delay to stabilize power \*/

initserial**();**

string\_in**(** **&**input\_string**[**0**]);**

delay**(**150**);** /\* give the lock time to get ready \*/

string\_out**(**"joherik\r\n"**);** /\* Change this to the ID of the card \*/

**while(**1**)** nop**();** /\* end of communication \*/

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTIONS

=========

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* EXAMPLE CODE FROM IE1206 \*/

void initserial**(** void **)** /\* initialise serialcom port \*/

**{**

PORTB.7 **=** 1**;**

TRISB.7 **=** 1**;** /\* input mode \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void putchar**(** char d\_out **)** /\* sends one char \*/

**{**

char bitCount**,** ti**;**

TRISB.7 **=** 0**;** /\* output mode \*/

PORTB.7 **=** 0**;** /\* set startbit \*/

**for** **(** bitCount **=** 10**;** bitCount **>** 0 **;** bitCount**--** **)**

**{**

/\* 104 usec at 3,58 MHz (5+27\*3-1+9=104) \*/

// ti = 27; do ; while( --ti > 0);

/\* 104 usec at 4 MHz (5+30\*3-1+1+9=104) \*/

ti **=** 30**;** **do** **;** **while(** **--**ti **>** 0**);** nop**();**

Carry **=** 1**;** /\* stopbit \*/

d\_out **=** rr**(** d\_out **);** /\* Rotate Right through Carry \*/

PORTB.7 **=** Carry**;**

**}**

nop2**();** nop2**();**

**return;** /\* all done \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

char getchar**(** void **)** /\* recieves one char \*/

**{**

/\* One start bit, one stop bit, 8 data bit, no parity = 10 bit. \*/

/\* Baudrate: 9600 baud => 104.167 usec. per bit. \*/

TRISB.7 **=** 1**;** /\* set input mode \*/

char d\_in**,** bitCount**,** ti**;**

**while(** PORTB.7 **==** 1 **)** /\* wait for startbit \*/ **;**

/\* delay 1,5 bit is 156 usec \*/

/\* 156 usec is 156 op @ 4 MHz ( 5+47\*3-1+2+9=156) \*/

ti **=** 47**;** **do** **;** **while(** **--**ti **>** 0**);** nop2**();**

**for(** bitCount **=** 8**;** bitCount **>** 0 **;** bitCount**--)**

**{**

Carry **=** PORTB.7**;**

d\_in **=** rr**(** d\_in**);** /\* rotate carry \*/

/\* delay 1 bit is 104 usec \*/

/\* 104 usec is 104 op @ 4 MHz (5+30\*3-1+1+9=104) \*/

ti **=** 30**;** **do** **;** **while(** **--**ti **>** 0**);** nop**();**

**}**

**return** d\_in**;**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void string\_in**(** char **\*** input\_string **)**

**{**

char charCount**,** c**;**

**for(** charCount **=** 0**;** **;** charCount**++** **)**

**{**

c **=** getchar**(** **);** /\* input 1 character \*/

input\_string**[**charCount**]** **=** c**;** /\* store the character \*/

//putchar( c ); /\* don't echo the character \*/

**if(** **(**charCount **==** **(**MAX\_STRING**-**1**))||(**c**==**'\r' **))** /\* end of input \*/

**{**

input\_string**[**charCount**]** **=** '\0'**;** /\* add "end of string" \*/

**return;**

**}**

**}**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void string\_out**(**const char **\*** string**)**

**{**

char i**,** k**;**

**for(**i **=** 0 **;** **;** i**++)**

**{**

k **=** string**[**i**];**

**if(** k **==** '\0'**)** **return;** /\* found end of string \*/

putchar**(**k**);**

**}**

**return;**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

bit check\_candidate**(** char **\*** input\_string**,** const char **\*** candidate\_string **)**

**{**

/\* compares input buffer with the candidate string \*/

char i**,** c**,** d**;**

**for(**i**=**0**;** **;** i**++)**

**{**

c **=** input\_string**[**i**];**

d **=** candidate\_string**[**i**];**

**if(**d **!=** c **)** **return** 0**;** /\* no match \*/

**if(** d **==** '\0' **)** **return** 1**;** /\* exact match \*/

**}**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void delay**(** char millisec**)**

/\*

Delays a multiple of 1 milliseconds at 4 MHz

using the TMR0 timer

\*/

**{**

OPTION **=** 2**;** /\* prescaler divide by 8 \*/

**do** **{**

TMR0 **=** 0**;**

**while** **(** TMR0 **<** 125**)** /\* 125 \* 8 = 1000 \*/

**;**

**}** **while** **(** **--** millisec **>** 0**);**

**}**

**Testprogram i Fritzing**

/\*TEST FILE FOR PROCESSOR\*/

/\* This file contains the main

\* function and the program logic \*/

/\* First include of chipkit-header \*/

#include "16F690.h"

#pragma config |= 0x00D4

#define MAX\_STRING 16

/\*String related functions\*/

char get\_char**(**void**);**

void string\_in**(**char **\*** string**);**

void string\_out**(**const char **\*** string**);**

/\*Hardware related funcitons \*/

void set\_led\_red**(**bit state**);**

void set\_led\_green**(**bit state**);**

void print\_to\_display**(**char val**);**

void delay**(**char millisec**);**

/\*Initializing and registry clearing\*/

void initialize**(**void**);**

void overrun\_recover**(**void**);**

void main**(**void**)** **{**

/\* String to store text from card \*/

char card\_str**[**MAX\_STRING**];**

/\* Initialize some code \*/

initialize**();**

/\* Loop forever, program logic below \*/

**while** **(**1**)** **{**

/\* Wait for card insertion \*/

**while** **(**PORTC.3 **==** 0**);**

delay**(**100**);** /\* card debounce \*/

delay**(**50**);** /\* extra delay \*/

/\* ask the question \*/

string\_out**(**"Send the ID please\r\n"**);**

delay**(**100**);** /\* USART is buffered, so wait until all chars sent \*/

/\* empty the reciever FIFO, it's now full with garbage \*/

overrun\_recover**();**

/\* Get id from card, stored in card\_str \*/

string\_in**(&**card\_str**[**0**]);**

set\_led\_red**(**1**);**

set\_led\_green**(**1**);**

//While card is inserted, loop

**while** **(**PORTC.3 **==** 1**)** **{**

**if(**i **<** 10**)**

print\_to\_display**(**i**);**

**else**

i **=** 0**;**

delay**(**1000**);**

**}**

delay**(**10**);**

/\* Reset the LED:s \*/

set\_led\_red**(**0**);**

set\_led\_green**(**0**);**

delay**(**100**);** /\* card debounce \*/

**}**

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTIONS

=========

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* EXAMPLE CODE FROM IE1206 \*/

void string\_in**(**char **\*** string**)** **{**

char charCount**,** c**;**

**for(**charCount **=** 0**;** **;** charCount**++)** **{**

c **=** get\_char**();** /\* input 1 character \*/

string**[**charCount**]** **=** c**;** /\* store the character \*/

// put\_char( c ); /\* don't echo the character \*/

/\* end of input \*/

**if((**charCount **==** **(**MAX\_STRING**-**1**))** **||** **(**c**==**'\r'**))** **{**

string**[**charCount**]** **=** '\0'**;** /\* add "end of string" \*/

**return;**

**}**

**}**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void string\_out**(**const char **\*** string**)** **{**

char i**,** k**;**

**for(**i **=** 0 **;** **;** i**++)** **{**

k **=** string**[**i**];**

**if(** k **==** '\0'**)** **return;** /\* found end of string \*/

put\_char**(**k**);**

**}**

**return;**

**}**

void set\_led\_red**(**bit state**)** **{**

PORTA.2 **=** state**;**

nop**();**

**}**

void set\_led\_green**(**bit state**)** **{**

PORTB.4 **=** state**;**

nop**();**

**}**

void print\_to\_display**(**char val**)** **{**

nop**();**

/\* Print hex-value to 7-segment display \*/

char value **=** val**;**

char i**;**

**if** **(**value **==** **-**1**)** **{**

PORTC.4 **=** 0**;**

nop**();**

**}**

**else** **if** **(**value **>=** 0 **&&** value **<** 10**)** **{**

PORTC.4 **=** 1**;**

nop**();**

delay**(**1**);**

PORTC.7 **=** 1**;**

nop**();**

delay**(**1**);**

PORTC.7 **=** 0**;**

nop**();**

delay**(**1**);**

**for** **(**i **=** 0**;** i **<** value**;** i**++)** **{**

PORTC.6 **=** 1**;**

nop**();**

delay**(**1**);**

PORTC.6 **=** 0**;**

nop**();**

delay**(**1**);**

**}**

**}**

**}**

/\* Delays a multiple of 1 milliseconds at 4 MHz

using the TMR0 timer by B. Knudsen \*/

void delay**(**char millisec**)** **{**

OPTION **=** 2**;** /\* prescaler divide by 8 \*/

**do** **{**

TMR0 **=** 0**;**

**while** **(**TMR0 **<** 125**);** /\* 125 \* 8 = 1000 \*/

**}** **while** **(--**millisec **>** 0**);**

**}**

void initialize**(**void**)** **{**

TRISA.0 **=** 1**;** /\* RA0 not to disturb PK2 UART Tool \*/

ANSEL.0 **=** 0**;** /\* RA0 digital input \*/

TRISA.1 **=** 1**;** /\* RA1 not to disturb PK2 UART Tool \*/

ANSEL.1 **=** 0**;** /\* RA1 digital input \*/

/\* Initialize PIC16F690 serialcom port \*/

/\* One start bit, one stop bit, 8 data bit, no parity. 9600 Baud. \*/

TXEN **=** 1**;** /\* transmit enable \*/

SYNC **=** 0**;** /\* asynchronous operation \*/

TX9 **=** 0**;** /\* 8 bit transmission \*/

SPEN **=** 1**;**

BRGH **=** 0**;** /\* settings for 6800 Baud \*/

BRG16 **=** 1**;** /\* @ 4 MHz-clock frequency \*/

SPBRG **=** 25**;**

CREN **=** 1**;** /\* Continuous receive \*/

RX9 **=** 0**;** /\* 8 bit reception \*/

ANSELH.3 **=** 0**;** /\* RB5 digital input for serial\_in \*/

/\* More init \*/

TRISC.3 **=** 1**;** /\* RC3 card contact is input \*/

ANSEL.7 **=** 0**;** /\* RC3 digital input \*/

TRISA.2 **=** 0**;** /\* RC2 Red LED for no access \*/

PORTA.2 **=** 0**;** /\* RC2 initially off \*/

TRISB.4 **=** 0**;** /\* RB4 Green LED for access \*/

PORTB.4 **=** 0**;** /\* RB4 initially off \*/

TRISC.1 **=** 1**;** /\* RC1 Button is set to input \*/

ANSEL.5 **=** 0**;** /\* RC1 digital input \*/

//Display initialize

TRISC.7 **=** 0**;** /\* RC7 Output to reset pin on display \*/

PORTC.7 **=** 0**;** /\* RC7 initially off \*/

TRISC.6 **=** 0**;** /\* RC6 Output to display clock \*/

PORTC.6 **=** 0**;** /\* RC6 initially off \*/

TRISC.4 **=** 0**;** /\* RC4 Enable display \*/

PORTC.4 **=** 1**;** /\* RC4 on \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void overrun\_recover**(**void**)** **{**

char trash**;**

trash **=** RCREG**;**

trash **=** RCREG**;**

CREN **=** 0**;**

CREN **=** 1**;**

**}**

**Test för smartcard**

/\* main\_key\_template.c question and answer, compare strings \*/

/\* This program is for 16F84 Gold Card \*/

/\*

SmartCard contact

\_\_ \_\_ \_\_

+5V |C1| C5| Gnd

|\_\_| \_\_|

MCLR |C2| |C6|

|\_\_| |\_\_|

OSC |C3| |C7| RB7/PGD I/O -><- Txd/Rxd half duplex

|\_\_| |\_\_|

|C4| |C8|

|\_\_|\_\_|\_\_|

\*/

/\*

SERIAL COMMUNICATION

============================

One start bit, one stop bit, 8 data bit, no parity = 10 bit.

Baudrate: 9600 baud => 104.167 usec. per bit.

serial output PORTB.7 half duplex!

serial input PORTB.7 half duplex!

\*/

#include "16F84.h"

#define MAX\_STRING 16 /\* string input max 15 characthers \*/

#pragma config |= 0x3ff1

/\* Function prototypes \*/

void initserial**(** void **);**

void putchar**(** char **);**

char getchar**(** void **);**

void string\_out**(** const char **\*** string **);**

void string\_in**(** char **\*** **);**

bit check\_candidate**(** char **\*** input\_string**,** const char **\*** candidate\_string **);**

void delay**(** char **);**

void main**(** void**)**

**{**

char i**,** c**,** d**,** charCount**;**

char input\_string**[**MAX\_STRING**];** /\* 15 char buffer for input string \*/

bit compare**;**

delay**(**50**);** /\* delay to stabilize power \*/

initserial**();**

string\_in**(** **&**input\_string**[**0**]);**

delay**(**150**);** /\* give the lock time to get ready \*/

string\_out**(**"joherik\r\n"**);** /\* Change this to the ID of the card \*/

**while(**1**)** nop**();** /\* end of communication \*/

**}**

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

FUNCTIONS

=========

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\* EXAMPLE CODE FROM IE1206 \*/

void initserial**(** void **)** /\* initialise serialcom port \*/

**{**

PORTB.7 **=** 1**;**

TRISB.7 **=** 1**;** /\* input mode \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void putchar**(** char d\_out **)** /\* sends one char \*/

**{**

char bitCount**,** ti**;**

TRISB.7 **=** 0**;** /\* output mode \*/

PORTB.7 **=** 0**;** /\* set startbit \*/

**for** **(** bitCount **=** 10**;** bitCount **>** 0 **;** bitCount**--** **)**

**{**

/\* 104 usec at 3,58 MHz (5+27\*3-1+9=104) \*/

// ti = 27; do ; while( --ti > 0);

/\* 104 usec at 4 MHz (5+30\*3-1+1+9=104) \*/

ti **=** 30**;** **do** **;** **while(** **--**ti **>** 0**);** nop**();**

Carry **=** 1**;** /\* stopbit \*/

d\_out **=** rr**(** d\_out **);** /\* Rotate Right through Carry \*/

PORTB.7 **=** Carry**;**

**}**

nop2**();** nop2**();**

**return;** /\* all done \*/

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

char getchar**(** void **)** /\* recieves one char \*/

**{**

/\* One start bit, one stop bit, 8 data bit, no parity = 10 bit. \*/

/\* Baudrate: 9600 baud => 104.167 usec. per bit. \*/

TRISB.7 **=** 1**;** /\* set input mode \*/

char d\_in**,** bitCount**,** ti**;**

**while(** PORTB.7 **==** 1 **)** /\* wait for startbit \*/ **;**

/\* delay 1,5 bit is 156 usec \*/

/\* 156 usec is 156 op @ 4 MHz ( 5+47\*3-1+2+9=156) \*/

ti **=** 47**;** **do** **;** **while(** **--**ti **>** 0**);** nop2**();**

**for(** bitCount **=** 8**;** bitCount **>** 0 **;** bitCount**--)**

**{**

Carry **=** PORTB.7**;**

d\_in **=** rr**(** d\_in**);** /\* rotate carry \*/

/\* delay 1 bit is 104 usec \*/

/\* 104 usec is 104 op @ 4 MHz (5+30\*3-1+1+9=104) \*/

ti **=** 30**;** **do** **;** **while(** **--**ti **>** 0**);** nop**();**

**}**

**return** d\_in**;**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void string\_in**(** char **\*** input\_string **)**

**{**

char charCount**,** c**;**

**for(** charCount **=** 0**;** **;** charCount**++** **)**

**{**

c **=** getchar**(** **);** /\* input 1 character \*/

input\_string**[**charCount**]** **=** c**;** /\* store the character \*/

//putchar( c ); /\* don't echo the character \*/

**if(** **(**charCount **==** **(**MAX\_STRING**-**1**))||(**c**==**'\r' **))** /\* end of input \*/

**{**

input\_string**[**charCount**]** **=** '\0'**;** /\* add "end of string" \*/

**return;**

**}**

**}**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void string\_out**(**const char **\*** string**)**

**{**

char i**,** k**;**

**for(**i **=** 0 **;** **;** i**++)**

**{**

k **=** string**[**i**];**

**if(** k **==** '\0'**)** **return;** /\* found end of string \*/

putchar**(**k**);**

**}**

**return;**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

bit check\_candidate**(** char **\*** input\_string**,** const char **\*** candidate\_string **)**

**{**

/\* compares input buffer with the candidate string \*/

char i**,** c**,** d**;**

**for(**i**=**0**;** **;** i**++)**

**{**

c **=** input\_string**[**i**];**

d **=** candidate\_string**[**i**];**

**if(**d **!=** c **)** **return** 0**;** /\* no match \*/

**if(** d **==** '\0' **)** **return** 1**;** /\* exact match \*/

**}**

**}**

/\* EXAMPLE CODE FROM IE1206 \*/

void delay**(** char millisec**)**

/\*

Delays a multiple of 1 milliseconds at 4 MHz

using the TMR0 timer

\*/

**{**

OPTION **=** 2**;** /\* prescaler divide by 8 \*/

**do** **{**

TMR0 **=** 0**;**

**while** **(** TMR0 **<** 125**)** /\* 125 \* 8 = 1000 \*/

**;**

**}** **while** **(** **--** millisec **>** 0**);**

**}**