/\*

\* LedControl.cpp - A library for controling Leds with a MAX7219/MAX7221

\* Copyright (c) 2007 Eberhard Fahle

\*

\* Permission is hereby granted, free of charge, to any person

\* obtaining a copy of this software and associated documentation

\* files (the "Software"), to deal in the Software without

\* restriction, including without limitation the rights to use,

\* copy, modify, merge, publish, distribute, sublicense, and/or sell

\* copies of the Software, and to permit persons to whom the

\* Software is furnished to do so, subject to the following

\* conditions:

\*

\* This permission notice shall be included in all copies or

\* substantial portions of the Software.

\*

\* THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,

\* EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES

\* OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND

\* NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT

\* HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,

\* WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING

\* FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR

\* OTHER DEALINGS IN THE SOFTWARE.

\*/

#include "LedControl.h"

//the opcodes for the MAX7221 and MAX7219

#define OP\_NOOP 0

#define OP\_DIGIT0 1

#define OP\_DIGIT1 2

#define OP\_DIGIT2 3

#define OP\_DIGIT3 4

#define OP\_DIGIT4 5

#define OP\_DIGIT5 6

#define OP\_DIGIT6 7

#define OP\_DIGIT7 8

#define OP\_DECODEMODE 9

#define OP\_INTENSITY 10

#define OP\_SCANLIMIT 11

#define OP\_SHUTDOWN 12

#define OP\_DISPLAYTEST 15

LedControl::LedControl(int dataPin, int clkPin, int csPin, int numDevices) {

SPI\_MOSI=dataPin;

SPI\_CLK=clkPin;

SPI\_CS=csPin;

if(numDevices<=0 || numDevices>8 )

numDevices=8;

maxDevices=numDevices;

pinMode(SPI\_MOSI,OUTPUT);

pinMode(SPI\_CLK,OUTPUT);

pinMode(SPI\_CS,OUTPUT);

digitalWrite(SPI\_CS,HIGH);

SPI\_MOSI=dataPin;

for(int i=0;i<64;i++)

status[i]=0x00;

for(int i=0;i<maxDevices;i++) {

spiTransfer(i,OP\_DISPLAYTEST,0);

//scanlimit is set to max on startup

setScanLimit(i,7);

//decode is done in source

spiTransfer(i,OP\_DECODEMODE,0);

clearDisplay(i);

//we go into shutdown-mode on startup

shutdown(i,true);

}

}

int LedControl::getDeviceCount() {

return maxDevices;

}

void LedControl::shutdown(int addr, bool b) {

if(addr<0 || addr>=maxDevices)

return;

if(b)

spiTransfer(addr, OP\_SHUTDOWN,0);

else

spiTransfer(addr, OP\_SHUTDOWN,1);

}

void LedControl::setScanLimit(int addr, int limit) {

if(addr<0 || addr>=maxDevices)

return;

if(limit>=0 && limit<8)

spiTransfer(addr, OP\_SCANLIMIT,limit);

}

void LedControl::setIntensity(int addr, int intensity) {

if(addr<0 || addr>=maxDevices)

return;

if(intensity>=0 && intensity<16)

spiTransfer(addr, OP\_INTENSITY,intensity);

}

void LedControl::clearDisplay(int addr) {

int offset;

if(addr<0 || addr>=maxDevices)

return;

offset=addr\*8;

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

status[offset+i]=0;

spiTransfer(addr, i+1,status[offset+i]);

}

}

void LedControl::setLed(int addr, int row, int column, boolean state) {

int offset;

byte val=0x00;

if(addr<0 || addr>=maxDevices)

return;

if(row<0 || row>7 || column<0 || column>7)

return;

offset=addr\*8;

val=B10000000 >> column;

if(state)

status[offset+row]=status[offset+row]|val;

else {

val=~val;

status[offset+row]=status[offset+row]&val;

}

spiTransfer(addr, row+1,status[offset+row]);

}

void LedControl::setRow(int addr, int row, byte value) {

int offset;

if(addr<0 || addr>=maxDevices)

return;

if(row<0 || row>7)

return;

offset=addr\*8;

status[offset+row]=value;

spiTransfer(addr, row+1,status[offset+row]);

}

void LedControl::setColumn(int addr, int col, byte value) {

byte val;

if(addr<0 || addr>=maxDevices)

return;

if(col<0 || col>7)

return;

for(int row=0;row<8;row++) {

val=value >> (7-row);

val=val & 0x01;

setLed(addr,row,col,val);

}

}

void LedControl::setDigit(int addr, int digit, byte value, boolean dp) {

int offset;

byte v;

if(addr<0 || addr>=maxDevices)

return;

if(digit<0 || digit>7 || value>15)

return;

offset=addr\*8;

v=pgm\_read\_byte\_near(charTable + value);

if(dp)

v|=B10000000;

status[offset+digit]=v;

spiTransfer(addr, digit+1,v);

}

void LedControl::setChar(int addr, int digit, char value, boolean dp) {

int offset;

byte index,v;

if(addr<0 || addr>=maxDevices)

return;

if(digit<0 || digit>7)

return;

offset=addr\*8;

index=(byte)value;

if(index >127) {

//no defined beyond index 127, so we use the space char

index=32;

}

v=pgm\_read\_byte\_near(charTable + index);

if(dp)

v|=B10000000;

status[offset+digit]=v;

spiTransfer(addr, digit+1,v);

}

void LedControl::spiTransfer(int addr, volatile byte opcode, volatile byte data) {

//Create an array with the data to shift out

int offset=addr\*2;

int maxbytes=maxDevices\*2;

for(int i=0;i<maxbytes;i++)

spidata[i]=(byte)0;

//put our device data into the array

spidata[offset+1]=opcode;

spidata[offset]=data;

//enable the line

digitalWrite(SPI\_CS,LOW);

//Now shift out the data

for(int i=maxbytes;i>0;i--)

shiftOut(SPI\_MOSI,SPI\_CLK,MSBFIRST,spidata[i-1]);

//latch the data onto the display

digitalWrite(SPI\_CS,HIGH);

}