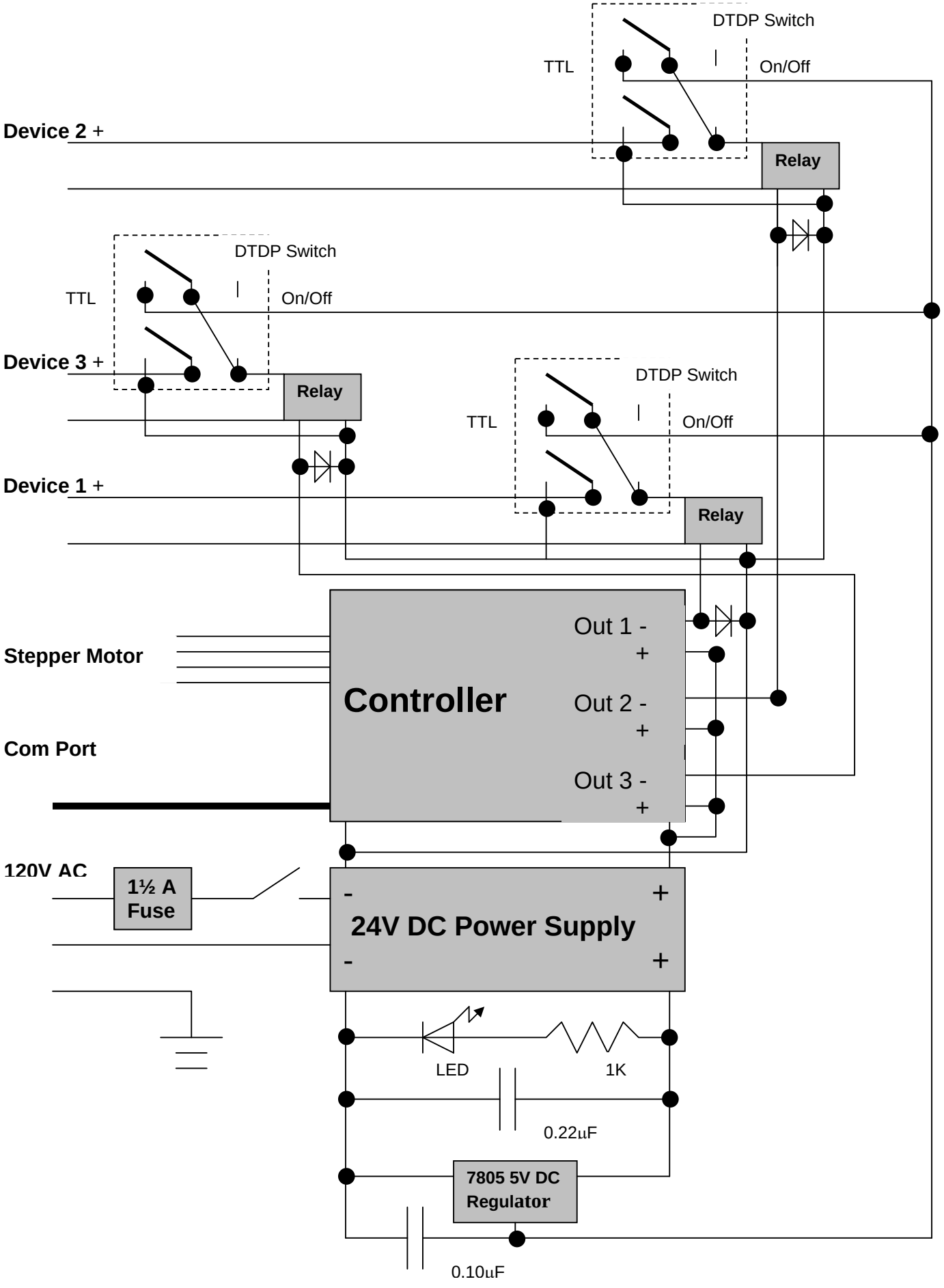
Stepper Motor & Device Controller



Parts List

Description	Source	Quantity	Part Number
24V DC Power Supply	Sola (Allied)	1	921-9350
Controller	AMP	1	3540i
Chassis	(Allied)	1	806-1200
Telephone Jack	(Allied)	1	512-5836
7-Pin Female Connector	WPI (Allied)	1	719-4520
DTDP Switches	K&E (Allied)	3	676-9241
Relay	SCR (Allied)	3	681-0015
4-Conductor Tel Wire		1 ft	
Modular Tel Plugs		2	
1K Resistor		1	
Terminal Block	Radio Shack	1	910-321B
Power Switch	Radio Shack	1	275-602A
Fuse Holder	Radio Shack	1	270-367A
1 1/2A Fuse	Radio Shack	1	27-1006
LED	Radio Shack	1	276-084A
Grommet	Radio Shack	1	64-3025
Circuit Board	Radio Shack	1	276-150A

Device Driver (for Microsoft Windows)

```
#include "../include/stdsdk.h"
#define AMP_MAXTRIAL          25

static BOOL bInit=FALSE;
static HANDLE hAMP=NULL;
static WORD wSwitch=0;
static FLOAT fSpeed=1.0;
static CRITICAL_SECTION cs;
BOOL AMPSet(INT nCommand,WORD wPort,LONG lParam);
BOOL AMPGet(PLONG plParam);
static BOOL AMPInit(VOID);
static BOOL AMPOn(WORD wPort);
static BOOL AMPOff(WORD wPort);
static BOOL AMPOnOff(WORD wPort, LONG lTime);
static BOOL AMPMotor(LONG lStep);
static BOOL AMPMotorSpeed(LONG lSpeed);
static BOOL AMPMotorHold(BOOL bHold);
static BOOL AMPCommand(LPSTR lpsCode);
static BOOL AMPQuery(LPSTR lpsCode,LPSTR lpsAnswer);

BOOL
AMPSet(INT nCommand,WORD wPort,LONG lParam)
{ // use this command for serielized access to AMP services
      BOOL bStatus;

      if(bInit==FALSE)
      {
            if(AMPInit()==FALSE)return FALSE;
            if(nCommand==CONTROLLER_INITIALIZE)return TRUE;
      }
      if(nCommand==CONTROLLER_OPEN)return TRUE;
      EnterCriticalSection(&cs);
      switch(nCommand)
```

```
	{
		case CONTROLLER_INITIALIZE:
			bStatus=AMPInit();
			break;
		case CONTROLLER_DEINITIALIZE:
			DeleteCriticalSection (&cs);
			CloseHandle (hAMP);
			hAMP = NULL;
			return TRUE;
		case CONTROLLER_ON:
			if((BOOL)lParam==TRUE)bStatus=AMPOn(wPort);
			else bStatus=AMPOff(wPort);
			break;
		case CONTROLLER_ONOFF:
			if(lParam==0)bStatus=TRUE;
			else bStatus=AMPOnOff(wPort,lParam);
			break;
		case CONTROLLER_MOTOR:
			bStatus=AMPMotor(lParam);
			break;
		case CONTROLLER_MOTORHOLD:
			bStatus=AMPMotorHold((BOOL)lParam);
			break;
		case CONTROLLER_MOTORSPEED:
			bStatus=AMPMotorSpeed(lParam);
			break;
		default:
			bStatus=FALSE;
			break;
	}
	LeaveCriticalSection(&cs);
	return bStatus;
}

BOOL
AMPGet(PLONG plParam)
{
	*plParam=(LONG)wSwitch;
	return(TRUE);
}

BOOL
AMPInit(VOID)
{
	CHAR szBuffer[32];
	WORD w;
	FLOAT fData;
	DCB dcb;
	COMMTIMEOUTS timeouts;

	if(hAMP!=NULL)
	{
		CloseHandle(hAMP);
		hAMP=NULL;
	}
	w=1; //use COM1
	sprintf(szBuffer,"COM%d",w);
	hAMP=CreateFile (szBuffer,GENERIC_READ | GENERIC_WRITE,
		0,NULL,OPEN_EXISTING,FILE_FLAG_WRITE_THROUGH,NULL);
	if(GetCommState(hAMP,&dcb)==FALSE)return FALSE;
	dcb.BaudRate=CBR_9600;
	dcb.fParity=FALSE;
	dcb.Parity=NOPARITY;
	dcb.ByteSize=8;
	dcb.StopBits=ONESTOPBIT;
	if (SetCommState(hAMP,&dcb)==FALSE)return FALSE;
```

```
	if(GetCommTimeouts(hAMP,&timeouts)==FALSE)return FALSE;
	timeouts.ReadIntervalTimeout=1000;
	timeouts.ReadTotalTimeoutMultiplier=10;
	timeouts.ReadTotalTimeoutConstant=10;
	timeouts.WriteTotalTimeoutMultiplier=10;
	timeouts.WriteTotalTimeoutConstant=10;
	if(SetCommTimeouts(hAMP,&timeouts)==FALSE)return FALSE;
	if(AMPCommand("ST\r")==FALSE)return FALSE;
	if(AMPQuery("RS\r",szBuffer)==FALSE)return FALSE;
	if(szBuffer[0]!='R')return FALSE;

	fData=(FLOAT)0.66; //0.66A motor
	sprintf(szBuffer,"CC%4.2f\r",fData);
	AMPCommand(szBuffer);
	sprintf(szBuffer,"CI%4.2f\r",fData/2);
	AMPCommand(szBuffer);   //current

	AMPCommand("MR3\r");   //microstepping 1/10
	AMPCommand("SP0\r");   //set current position to 0
	AMPCommand("SO1H\r");
	AMPCommand("SO2H\r");
	AMPCommand("SO3H\r");
	wSwitch=0;
	if(!bInit)InitializeCriticalSection (&cs);
	bInit=TRUE;
	return TRUE;
}

BOOL
AMPOn(WORD wPort)
{
	if(wPort & 0x0001)AMPCommand("SO1L\r");
	if(wPort & 0x0002)AMPCommand("SO2L\r");
	if(wPort & 0x0004)AMPCommand("SO3L\r");
	wSwitch = (WORD)(wSwitch|wPort);
	return TRUE;
}

BOOL
AMPOff(WORD wPort)
{
	if(wPort & 0x0001)AMPCommand("SO1H\r");
	if(wPort & 0x0002)AMPCommand("SO2H\r");
	if(wPort & 0x0004)AMPCommand("SO3H\r");
	wSwitch = (WORD)(wSwitch&(~wPort));
	return TRUE;
}

BOOL
AMPOnOff(WORD wPort,LONG lTime)
{ // lTime in unit of 10 ms, must be <= 65535 or 655.35 sec
	CHAR szBuffer[32];

	if(lTime>655350)return FALSE;
	if(AMPOn(wPort)==FALSE)return FALSE;
	sprintf(szBuffer,"WT%.2f\r",(FLOAT)lTime/100.0);
	if(AMPCommand(szBuffer)==FALSE)return FALSE;
	if(AMPOff(wPort)==FALSE)return FALSE;
	return TRUE;
}

BOOL
AMPMotor(LONG lStep)
{
	CHAR szBuffer[32];
```

```
	if(lStep!=0)
	{
		sprintf(szBuffer,"DI%d\r",lStep);
		if(AMPCommand(szBuffer)==FALSE || AMPCommand("FL\r")==FALSE)return FALSE;
		if(lStep>0)Sleep((INT)(lStep/(fSpeed*2))+5);
		else Sleep((INT)(-lStep/(fSpeed*2))+5);
	}
	return TRUE;
}

BOOL
AMPMotorSpeed(LONG lSpeed)
{
	CHAR szBuffer[32];

	fSpeed=(FLOAT)(lSpeed/1000.0);
	sprintf(szBuffer,"VE%4.2f\r",fSpeed);
	return(AMPCommand(szBuffer));
}

BOOL
AMPMotorHold(BOOL bHold)
{
	if(bHold)return(AMPCommand("ME\r"));
	else return(AMPCommand("MD\r"));
}

BOOL
AMPCommand(LPSTR lpsCode)
{
	DWORD dwData;
	INT i;

	for(i=0;i<AMP_MAXTRIAL;++i)
	{
		if(WriteFile(hAMP,lpsCode,lstrlen(lpsCode),&dwData,NULL)==FALSE ||
			dwData!=(DWORD)lstrlen(lpsCode))continue;
		else break;
	}
	if(i==AMP_MAXTRIAL)return FALSE;
	else return TRUE;
}

BOOL
AMPQuery(LPSTR lpsCode,LPSTR lpsAnswer)
{
	BOOL bErr=TRUE;
	CHAR *lpc;
	DWORD dwData;
	INT i;

	for(i=0;i<AMP_MAXTRIAL;++i)
	{
		if(WriteFile(hAMP,lpsCode,lstrlen(lpsCode),&dwData,NULL)==FALSE ||
			dwData!=(DWORD)lstrlen(lpsCode))continue;
		bErr=TRUE;
		lpc=lpsAnswer;
		do
		{
			if (ReadFile(hAMP,lpc,1,&dwData,NULL)==FALSE || dwData!=1)
				bErr=FALSE;
			else ++lpc;

		}while(*lpc!='\r' && bErr==TRUE);
		if(bErr==TRUE)break;
	}
```

```
      if(i==AMP_MAXTRIAL)return FALSE;
      else return TRUE;
}
```