

Modem

INHERITS FROM	Serial
DECLARED IN	Modem.h
WRITTEN BY	Charles G. Bennett

Version 2.0, Tues. Aug 04 1992

CLASS DESCRIPTION

The Modem class is a subclass of Serial that adds an object oriented interface to standard "Hayes" style modems.

FEATURES

- Builds on the Serial Class and provides an easy to use, object oriented interface to using a Modem.

INSTANCE VARIABLES

Inherited from Object
Class isa;

Inherited from Serial

int	ttyPtr;
int	baudrate;
int	par;
id	target;
SEL	theSelector;
BOOL	notifyStatus;
char	deviceName[20];
BOOL	reEnteredTtyHandler;
unsigned long	timeout;
unsigned	int notifyThreshold;

Declared in Modem

struct	modemStruct localModem;
int	modemStatus;
int	numRings;
char	lastResponse[100];
BOOL	autoAnswerOn;

METHOD TYPES

Open and initializing a Modem Object

- init;
- defineModem: (struct modemStruct) aModem;
- getModemDef: (struct modemStruct *) aModem;

Checking the modemstatus

- (int) getModemStatus;
- (char*) getLastResponse;

Controlling the Modem

- (int) modemReset;
- (int) modemToneDial:(BOOL)flag;
- (int) modemDial: (char*) number AutoBaud:(BOOL) autoBaud
- (int) modemRedial:(BOOL) autoBaud;
- (int) modemHangup;
- (int) modemAnswer;
- (int) modemAutoAnswer: (int) numberOfRings;
- (int) modemAutoAnswer;
- (int) modemReadRegister:(int) reg;
- (int) modemWriteRegister:(int) reg withValue:(int)value;

INTERNAL METHODS

None

CLASS METHODS

None

INSTANCE METHODS

defineModem:

- **defineModem:** (struct modemStruct*) aModem;

Sets the modem objects current modem definition. (see the structure modemStruct)

getModemStatus

- (int) **getModemStatus;**

Returns the modem status variable. The defined constants are;

ONHOOK
OFFHOOK
CONNECTED
BUSY
DIALING
REDIALING
IDLE
REMOTERING
NOCARRIER.

getModemDef:

- **getModemDef:** (struct modemStruct*) aModem;

Returns the current modem definition in the structure pointed to by

aModem.

getLastResponse

- (char*) **getLastResponse**;

Returns the FULL text of the last modem response. For example the modem status may be CONNECTED but the full text may be "CONNECT 2400" so you would know to change the serial port baudrate to 2400 before continuing. This is a shallow copy (we return the pointer to the internal string variable) so DON'T expect it to remain constant.

init

- **init**

Set up for internal variables and signal handlers.

modemAnswer

- (int) **modemAnswer**;

Causes the modem to pickup the phone and answer. Status = CONNECTED.

Returns **PORTNOTOPEN** if there was no port open
OK otherwise.

modemAutoAnswer:

- (int) **modemAutoAnswer**: (int) numberOfRings;

Tells the modem to answer the phone after **numberOfRings**.
Returns **PORTNOTOPEN** if there was no port open
OK otherwise.

modemAutoAnswer

- (int) **modemAutoAnswer**;

Returns the number of rings required before the modem will answer the phone. 0 means that the modem will NOT answer the phone.

Returns **PORTNOTOPEN** if there was no port open

modemDial: AutoBaud:

- (int) **modemDial:** (char*) number AutoBaud:(BOOL) autoBaud ;

Dials the requested number..

Returns **PORTNOTOPEN** if there was no port open.

Returns CONNECT is connected or BUSY ETC. Look at the method **getLastResponse** to find the full text. If **autoBaud** equals YES then we will try to figure out the speed from the modem response line and adjust the speed of the serial port to match. If the variable equals NO then we leave the port speed alone.

modemHangup

- (int) **modemHangup**;

Drops DTR for 1 second and sets modem status = IDLE.

Returns **PORTNOTOPEN** if there was no port open
OK otherwise.

modemRedial:

- (int) **modemRedial**:(BOOL) autoBaud;

Calls **modemDial**: with the last number dialed. If **autoBaud** equals YES then we will try to figure out the speed from the modem response line and adjust the speed of the serial port to match.

Returns same as **modemDial**:

modemReset

- (int) **modemReset**;

Sends the modem reset string to the modem.
Returns **PORTNOTOPEN** if there was no port open.
Returns **TIMEOUT** if there is no response.

modemToneDial:

- **modemToneDial**:(BOOL)flag;

Tells the modem to use the **TONE** or **PULSE** dialing string.
YES = TONE , NO = PULSE.

modemReadRegister:

- (int) **modemReadRegister:(int) reg;**

Returns the value in the modem register specified by **reg**.

Returns **PORTNOTOPEN** if there was no port open.

modemWriteRegister: withValue:

- (int) **modemWriteRegister:(int) reg withValue:(int)value;**

Sets the value in the modem register specified by **reg** to **value**

Returns **PORTNOTOPEN** if there was no port open.

CONSTANTS AND DEFINED TYPES

DEFINED IN SERIAL

#define OFF	0
#define ON	1
#define EVEN	1
#define ODD	2
#define NONE	3
#define SPACE	4
#define MARK	5
 #define SEC	 1000000L

#define SERIALOK 0

#define PORTNOTOPEN -1

#define TIMEOUT -2

#define BADPARITY -3

#define BADBAUD -4

#define BADLENGTH -5

DEFINED IN MODEM

#define OK 0

#define ONHOOK 1

#define OFFHOOK 2

#define CONNECTED 3

#define BUSY 4

#define DIALING 5

#define REDIALING 6

#define IDLE 7

#define REMOTERING 8

#define NOCARRIER 9

#define MODEMRESPONSEERROR 10

#define AUTOANSRING 0

#define RING_CNT 1

#define ESC_CHAR 2

#define RETURN_CHAR 3

#define LINEFEED_CHAR 4

#define BS_CHAR 5

```

#define          WAIT_DIALTONE      6
#define          WAIT_CARRIER      7
#define          LEN_PAUSE          8
#define          CD_GUARD            9
#define          LOST_CDTIME10
#define          TONE_RATE           11
#define          ESC_GUARD           12
#define  DELAY_DTR                    25
#define  RTS_CTS_DELAY                26

```

```

struct modemStruct {
    char    type[40];           //name of modem
    char    lastNum[40];        //Last Number dialed
    char    term[3];           //Normal command terminator
    char    tone[20];          //tone dial prefix
    char    pulse[20];          //pulse dial prefix
    BOOL    toneDial;           //YES = TONE    NO=PULSE
    char    hangup[20];         //code to cause a hangUp
    char    reset[20];          //code to reset modem
    char    answer[20];         //code to cause answer
};

```