CyberX10

COLLABORATORS			
	TITLE : CyberX10		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		January 5, 2023	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

CyberX10

1.1 CyberX10 documentation

CyberX10

1.2 Legal Mumbo Jumbo

CyberX10 is not in the public domain. All source files, along with the resulting executable, are copyright by C. Wichura. You may not sell CyberX10. The only allowed charge that may be placed on CyberX10 is for media and/or mailing costs.

CyberX10 may be freely redistributed via BBSs, InterNet/Usenet, and disk libraries such as Fred Fish's, as long as the archive is not modified. Disk magazines and services that charge extra for file transfers may not distribute CyberX10.

In using CyberX10, you accept the responsibility for any damage or loss of productivity/money that may occur through or during its use. C. Wichura is not and cannot be held accountable.

1.3 Supplied Files

All source to CyberX10 is included in the Source/* directory. ↩ Please do not redistribute changes you make. Instead, send them to me for inclusion in the master distribution.

The Catalogs directory is where translation catalogs for languages that CyberX10 has been converted to are found. If you are running under Workbench 2.1 and would like to use CyberX10 in your language, copy the appropriate file into your LOCALE:Catalogs/<language> directory.

1.4 What are X10 devices?

The X10 line of devices from BSR allow you to control lights and appliances remotely. Each device has an "address" in the form of a housecode (from A-P) and a unit number (from 1-16), giving a total of 256 possible addresses. Multiple receiver devices can share the same address, making them all respond to the same commands (for example, if you have three lights in your living room that you normally turn all off or all on and one time, you could give them the same address and be able to turn all three on or off with the touch of a single button).

There are several different types of transmitters and receivers available. Receivers include replacement wall switches (to control ceiling lights, for example), appliance units (where you can plug something into it and then plug the unit into the wall to control miscelaneous appliances), replacement wall outlets (allows the same sort of functionality as appliance units), heavy duty appliance units (220V capacity) as well as some more esoteric units such as a contact switch which could be used for things like garage door openers, etc. As far as transmitters go, there are simple ones that can control up to 8 devices by the press of a button, telephone responders (you can dial in with a touch-tone phone to turn things on or off), clocks (allows one to set times to turn a couple devices on and off automatically), a portable unit (like having a remote for your lights), and, of course, a couple versions of computer interfaces.

CyberX10 knows how to talk to the CP290 computer interface (it may support the others, but I don't have any so can't test it or even know if they use the same command set). The CP290 can access all 256 addresses and can 1) send immediate events (i.e., dim any light right now, etc.) as well as be programmed with up to 128 timed events, thus not requiring that it be connected to the computer at all times.

X10 stuff can be obtained from a number of places. I got mine through
Radio Shack, where it is refered to as "Plug'n Power". Receiver units are
generally around $12-$15 a piece, while transmitters range from $13 to
about $80 (the CP290).

1.5 How to use CyberX10

You can start CyberCron from either the CLI or Workbench. Workbench's
multi-selection is fully supported, so you can create several icons to turn
various devices on or off, etc.

CyberX10 understands the following arguments:

```
    [
            ON
             or
            OFF
             or
            DIM
             <dimness level from 1-16>] [
            TARGETS
            ]
    [
            BASEHC
             <housecode>]
    [
            SETCLOCK
            ]
    [
            DIAG
            ]
    [
            DOWNLOAD
             <filename of events to download to CP290>]
    [
            UPLOAD
             <filename to write events uploaded from CP290 to>]

    [
            DEVICE
             <serial device driver used to talk to CP290>]
    [
            UNIT
             <serial device's unit number that the CP290 uses>]
    [
            ATTEMPT
            ]
```

The only arguments with defaults are
DEVICE
,
UNIT
and
ATTEMPT

which default to serial.device, 0 and NO, respectively.

You can only do one "type" of operation at a time. For example, you can not turn a unit on and download events to the CP290 at the same time.

To use these options from a Workbench icon, simply make each tooltype be in the form of <option>=<setting>. For example, if you wanted to turn units A1, A2 and A4 on with the CP290 hooked up to fooserial.device unit 5, you would use

```
ON

TARGETS
=A1

TARGETS
=A2

TARGETS
=A4

DEVICE
=fooserial.device

UNIT
=5
```

Notice that to specify multiple targets, the TARGETS = option must be repeated.

1.6 The »ON« option

The ON option allows you to turn multiple units on. The command(s) are issued in direct mode, meaning that they are sent immediately.

When using the ON option, you must specify at least one target.

1.7 The »OFF« option

The OFF option allows you to turn multiple units off. The command(s) are issued in direct mode, meaning that they are sent immediately.

When using the OFF option, you must specify at least one target.

1.8 The »DIM« option

The DIM option allows you to set the dimness level of a lamp appliance.
The dimness value may be anything between 1 (brightest) to 16 (dimmest).

When using the DIM option, you must specify at least one target.

1.9 The »TARGETS« option

This is where you specify the targets of a direct mode command (ON, OFF or DIM). Targets take the form of a single character housecode followed immediately by a 1 or 2 digit unit number. For example, A1 and P16 are both legal targets. Q1 is not (housecode is invalid), nor is "A 1" because it contains a space.

1.10 The »BASEHC« option

This allows you to set the CP290's base housecode. It takes a single character argument as it's only option. Please note that setting the base housecode will erase any programming in the CP290's memory, so you may want to upload events to a file first.

The base housecode determines which units are accessible from the rocker buttons on the CP290's keypad.

As an example, to make C the base housecode, you would use

```
  BaseHC C
```

1.11 The »SETCLOCK« option

This will set the CP290's internal clock to match the Amiga's system clock.

1.12 The »DIAG« option

This will run the CP290's diagnostic test. Be patient, as it takes a little time. Please note that running the diagnostic will erase any programming in the CP290's memory, so you may want to upload events to a file first.

If the unit passes it's test, CyberX10 will print the settings of the CP290's current house code and internal clock.

If the CP290 fails it's diagnostic, one of two things will be reported. Either a timeout error will occur (the unit took too long to respond) or a status flag error will occur (the unit replied in time, but replied that a fault was detected). Either way, try running the diagnostic once or twice more and if the problem is persistant, see about having you CP290 repaired.

1.13 The »DOWNLOAD« option

This will read timer events and graphics data from the specified file and download them to the CP290's memory. Please be patient using this option, as it can take up to three and a half minutes to complete. See the section on CyberX10's file format for programming the CP290 for how to create a file suitable for downloading.

Downloading completely replaces any programming that may have been present in the CP290's memory. Thus, if you wish to modify, delete or add an event you should probably upload the CP290's current programming to a file, edit it, and then download the edited file.

1.14 The »UPLOAD« option

This will upload the timer events and graphics data from the CP290 and write them to a file. This file can then be saved (to back up the CP290's memory) or edited and then downloaded (to alter the CP290's programming).

See the section on CyberX10’s file format for programming the CP290 for more information on the file format written.

1.15 The »DEVICE« option

Allows one to specify the serial device to use when talking to the CP290. This option is case sensitive. The default device is "serial.device".

If you use a different device than the default, you may wish to make a shell alias that specifies the device for you automatically. For example,

```
  alias x10 "CyberX10 device siosbx.device []"
```

1.16 The »UNIT« option

Allows one to specify the unit number to use when talking to the CP290. The default unit number is zero.

If you use a different unit number than the default, you may wish to make a shell alias that specifies the unit for you automatically. For example,

```
  alias x10 "CyberX10 unit 2 []"
```

1.17 The »ATTEMPT« option

On systems that have OwnDevUnit.library installed, CyberX10 will normally block waiting for the serial device used to talk to the CP290 to come free. This is usually the desired action, as it allows one to have several CyberX10 commands started at the same time (say by entries in a crontab file) and no commands will get dropped.

Using the ATTEMPT option, CyberX10 will not block. Instead, it will wait up to five seconds for the device to become free. If the device doesn’t become available within that time, CyberX10 will give up and the command will not be executed.

1.18 How to program the CP290

CyberX10 allows one to program the CP290’s internal memory. This can be used to create timed events as well as store graphics data. (While CyberX10 doesn’t actually make use of the graphics data, support is provided for it as a means of backing up the CP290’s memory or to allow a front-end to CyberX10 to store graphics data.)

One programs the CP290 by writing a text file with a special format (described below) and then using the download option to send the file to the CP290.

Each line of the download file is considered a single timer event or graphics data item. Lines that start with a # character are treated as comments. Blank lines are also acceptable.

Each line must start with either "timer" or "graphics". This tells CyberX10 what type of event is on this line so it knows how to parse it.

Graphics data is very simple. After the "graphics" keyword, there should be two numbers, both with a legal range of 0 to 254 (not 255!). For example,

```
  graphics 1 2
  graphics 3 15
```

are both legal graphics data items.

Timer events are more complex. They take the form of

```
    [
               NORMAL
                or
               SECURITY
               ]
    [
               TODAY
                or
               TOMORROW
                or any combination of

               EVERYDAY
               ,
               WEEKENDS
               ,
               WEEKDAYS
               ,
               MON
               ,
               TUE
               ,
               WED
               ,
               THU
               ,
               FRI
               ,
               SAT
                and
               SUN
               ]
    [
               HOUR
                <hour with range 0-23>]
    [
               MINUTE
                <minute with range 0-59>]
    [
               ON
                or
               OFF
                or
               DIM
                <dimness level from 1-16>]
    [
               HOUSECODE
                <housecode>] [
               TARGETS
               ]
```

Some example timer events are:

```
  timer
                normal

                everyday

                on

                housecode
                 a 1 2 3
  timer
                security

                weekends

                mon

                wed

                fri

                off

                housecode
                 b 10
```

Please note that the CP290's timer event structure does not allow for units with different housecodes to be specified in the same event.

1.19 The »NORMAL« timer event option

```
                Tells the CP290 to execute the event at the time specified.  This ↩
                   is in
contrast to the
                SECURITY
```

option, which tells the CP290 to wait a random period of time of up to one hour after the time specified before sending the command.

1.20 The »SECURITY« timer event option

Tells the CP290 to wait a random period of time up to one hour after the time specified before sending the command. The idea here is that it makes the house seem more "lived in".

1.21 The »TODAY« timer event option

Tells the CP290 that the event is to be executed at the time specified on the day of programming and then never again.

1.22 The »TOMORROW« timer event option

Tells the CP290 that the event is to be executed at the time specified on the day after programming and then never again.

1.23 The »EVERYDAY« timer event option

Tells the CP290 that the event is to be executed every day.

1.24 The »WEEKENDS« timer event option

Tells the CP290 that the event is to be executed on weekends.

1.25 The »WEEKDAYS« timer event option

Tells the CP290 that the event is to be executed on weekdays.

1.26 The »MON« timer event option

Tells the CP290 that the event is to be executed on Mondays.

1.27 The »TUE« timer event option

Tells the CP290 that the event is to be executed on Tuesdays.

1.28 The »WED« timer event option

Tells the CP290 that the event is to be executed on Wednesdays.

1.29 The »THURSDAY« timer event option

Tells the CP290 that the event is to be executed on Thursdays.

1.30 The »FRIDAY« timer event option

Tells the CP290 that the event is to be executed on Fridays.

1.31 The »SATURDAY« timer event option

Tells the CP290 that the event is to be executed on Saturdays.

1.32 The »SUNDAY« timer event option

Tells the CP290 that the event is to be executed on Sundays.

1.33 The »HOUR« timer event option

Specifies the hour that the event is to be executed at. The hour is specified in 24 hour time, with a range of 0 (midnight) to 23 (11pm).

1.34 The »MINUTE« timer event option

Specified the minute that the event is to be executed at. Range is 0 to 59.

1.35 The »ON« timer event option

Indicates that the targets specified are to be turned on.

1.36 The »OFF« timer event option

Indicates that the targets specified are to be turned off.

1.37 The »DIM« timer event option

Indicates that the targets specified are to be dimed. The dimness value may range from 1 (brightest) to 16 (dimmest).

1.38 The »HOUSECODE« timer event option

Specifies the housecode of the targets .

1.39 The »TARGETS« timer event option

Specifies the unit numbers of the targets. Multiple unit numbers may be specified.

1.40 How to make CyberX10 speak your language

When running under Workbench 2.1, CyberX10 has the ability to use a translation catalog which contains the text strings it uses in a different language. These translation files are stored in the user's LOCALE:Catalogs/<language> directory. For example, the French string catalog for CyberX10 would be the file:

```
  LOCALE:Catalogs/français/CyberX10.catalog
```

Please note that case is important for the filename portion of the above filespec.

It is also possible to make a Catalogs sub-directory in the directory where the CyberX10 executable resides. This reduces the amount of clutter in the user's LOCALE: volume. If you placed the CyberX10 executable in SYS:WBStartup, then the corresponding location of the French catalog would be:

```
  SYS:WBStartup/Catalogs/français/CyberX10.catalog
```

If there isn’t a translation file for your language present in this distribution (and there most likely won’t be as I don’t know any foreign languages :->), you can make your own. To do this, you need to:

1) edit the dummy.ct file in the Source directory, translating it as appropriate. you should be looking at the Source/CyberX10_Strings.cd file as you do this since it contains many important notes about what the different strings are used for.

2) run CatComp (a Commodore developer tool which I can’t include) to make the actual catalog that CyberX10 will use. For example:

```
    catcomp Source/CyberX10_Strings.cd <language>.ct catalog
        locale:Catalogs/<language>/CyberX10.catalog
```

    (the above is all on one CLI line, of course)

If you don’t have access to CatComp then feel free to e-mail me your translation (lha and uuencode it, though, or any special accent marks, etc, will be lost) and I’ll run it through CatComp. I’d like any translation files people come up with, anyway, so I can distribute them with future versions of CyberX10. Also, if you feel that one of the currently available translation files has something which could be translated in a better way, feel free to mail me those changes as well.

1.41 The history of CyberX10

Version 1.1 -- Initial release

1.42 Credits

Willy Langeveld for some beta testing.

1.43 Contacting the author

If you happen to find a bug or have a suggestion for CyberX10, or just want to say "hey, cool program", please contact me using one of the ways listed below. Even if you wanna say "CyberX10 sucks", let me know and be sure to say why you feel this way so that I might be able to fix what you think is wrong with the program.

These electronic forms are the most prefered means of contacting me. They will get you a response pretty quick.

```
  e-mail: caw@miroc.chi.il.us
  BIX: caw

Snail Mail is pretty slow and I'm not known for being very good about
responding to it...  :-)

  Christopher A. Wichura
  5450 East View Park
  Chicago, Il.  60615
  USA

You can also reach me by phone.  However, please try to limit your calling
to evening hours (I'm in the central time zone).  If I'm not home and you
leave a message, call back again anyway.  Around here, one tends to get
maybe 5% of the messages left for them, if lucky...  :-)

  (312)/684-2941
```