4 PROGRAMMING GUIDE

SIGN CONVENTIONS

REASONING

The sign convention used is intuitive, not scientific. Both lane changes and evasive maneuvers generally begin with a left turn. Also, left/right stripcharts with positive-upward are visually correct with initial left turns.

PROGRAM

The output of the program is offset binary, which is bipolar. It must be positive for the first steer input. The binary program output is multiplied by the output of the Command Module in conversion to analog form.

COMMAND MODULE DIRECTION SWITCH

The direction +/- switch in the Command Module controls the polarity of the Program digital/analog converter. For "+" switch position the Command Module output is positive. This output multiplies the positive binary Program output to create a positive analog signal, to command an initial left turn.

The direction switch also turns on the left-hand Klutzlight.

REVERSAL OF SIGN CONVENTION

To reverse the sign convention:

1. Reverse the PHASE A and PHASE B leads on the motor encoder connector;
2. Reverse the motor leads;
3. Reverse the Klutzlight leads, pins 7 and 8 in the Handwheel D-connector.

PROGRAMMING GUIDE

(updated for Romsteer rev 1.45 thru 1.46)

The program EPROM for the steering machine is created using the Windows software program described here to design the program set (16 programs per ROM) and create binary EPROM files for the steer ROM and the auxiliary ROM. These files are then burned into EPROMs using an EPROM programmer's hardware and software.

The sign convention used for steer is positive to the left. In the program the initial steer input must be positive so that the klutzlight warning (controlled solely by the sign input from the command module) is in the correct direction. If the sign on the command module is set to "-" all actual steer inputs are reversed, including the initial steer going to the right (negative).

In the following descriptions a **program set** refers to the set of sixteen programs that are saved in a .ROM file and are burned into an EPROM, while a **program** refers to one of those sixteen.

Program syntax rules are as follows:

1. Comments may be added to the program before the Begin statement without using any special characters, although a preceding semicolon is still recommended. After the Begin statement, any comment must be preceded by a semicolon. In short, all text except for a MaxSteer command is ignored before Begin, and all text following a semicolon on any line is ignored.
2. Blank lines and extra spaces are also allowed and ignored.
3. Upper/lower case is unimportant.
4. All noncomment/nonblank lines must start with a valid function command, followed by any necessary parameters in parentheses and separated by commas.
5. Parameters must be correctly formatted, followed by at least the first character of the corresponding parameter id. Characters following the first are actually ignored (e.g. draw = d = degrees).
6. Valid parameter id's are:
   - hz (for clock rate and sine/square/triangle frequency)
   - degrees (for steer amplitude and sine/square/triangle phase angle)
   - seconds (for time lengths)
   - cycles (for auto-calculated time lengths of sine/square/triangle)
   - bit (for bit selection in Aux ROM functions)
7. The parameters may be listed in any order except in some cases where more than one parameter with the same id is needed.
8. With the exception of clock rate and bit number, all parameters are real numbers.

Note that cycles (if nonzero) supersedes seconds if both are present.
Also note that steer amplitudes are +/-*d* degrees, not peak-to-peak, for sine, triangle, and square waves

Defaults:

rate=1024 hz	frequency=0 hz
steer position=0 deg	phase=0 deg
time length=0 sec	number of cycles=0
bit=0 (invalid – correct bit # must be specified in command)	

The function command set is as follows (optional parameters in [ ] brackets):

MaxSteer(*n* degrees)	This command sets the scale factor for the EPROM data. The parameter *n* is the value that is entered on the command module and must be greater than or equal to the maximum absolute value steer in the program. This is the only command that is recognized before Begin.
Begin	Signifies the beginning of the body of the program. Everything before this is ignored except for a MaxSteer command. Any comments after this must be preceded by a semicolon.
SetRate(*f* hz[,*t* sec])	Sets the program step rate. The value of *f* must be 256, 1024, 2048, or 4096. May occur at any time during program. Effect is delayed for *t* seconds (*t* may be negative).
SetCurrentHigh[(*t* sec)]	Obsolete function that applied only to Sprint 1 machines. It is ignored in later models.
SetCurrentLow[(*t* sec)]	Obsolete function that applied only to Sprint 1 machines. It is ignored in later models.
Wait(*t* sec)	Holds steer for *t* seconds.
StepTo(*d* deg [,*t* sec])	Steers with maximum speed to *d* degrees. Optionally hold that steer for *t* seconds.
StepUp(*d* deg [,*t* sec])	Steers with maximum speed to existing angle plus *d* degrees (*d* may be negative). Optionally hold that steer for *t* seconds.
RampTo(*d* deg ,*t* sec)	Ramp to *d* degrees at a constant rate over *t* seconds.
RampUp(*d* deg ,*t* sec)	Ramp to existing angle plus *d* degrees at a constant rate over *t* seconds.
Sine(*d* deg, *f* hz [,*t* sec][,*c* cycles][,*p* deg])	Sine wave with frequency *f* and amplitude angle ±*d* degrees, for *c* cycles or *t* seconds (cycles take precedence if both are present). The function begins at the existing steer angle, and at phase angle *p* (0 if *p* is omitted).
Square(*d* deg, *f* hz [,*t* sec][,*c* cycles]][,*p* deg])	Square wave with frequency *f* and amplitude ±*d* degrees, for *c* cycles or *t* seconds (cycles take precedence if both are present). The function begins at the existing steer angle, and at phase angle *p* (0 if *p* is omitted).
Triangle(*d* deg, *f* hz [,*t* sec][,*c* cycles]][,*p* deg])	Triangle wave with frequency *f* and amplitude ±*d* degrees, for *c* cycles or *t* seconds (cycles take precedence if both are present). The function begins at the existing steer angle, and at phase angle *p* (0 if *p* is omitted).
SineSweep(*d* deg, *f1* hz, *f2* hz, *t1* sec [,*t2* sec])	Sine wave linear frequency sweep. Always low-high-low (*f1* and *f2* are swapped if *f1>f2).* If optional parameter t2 is missing, then the sweep is from *f1* to *f2* and back to *f1* in *t1* seconds. If *t2* is present then the sweep is from *f1* to *f2* in *t1* seconds and back to *f1* in *t2* seconds. In this case either (not both) of the time parameters may be zero (e.g. if t2=0 then sweep will end at *f2*). Cycle count is not valid for this function.

Function Command set continued

PulseFlag(*b* bit [,*t1* sec [,*t2* sec]])	Sets Flag bit #*b* in Aux ROM for *t1* seconds, then clears it. Minimum pulse width is 1/steprate. Default *t1*=0 (minimum pulse). Optionally delay effect for *t2* seconds (*t2* may be negative). t1 must be used (may be 0) in order to use t2. Note pulse width here does not delay any part of main ROM program. (e.g. A rampto command following a pulseflag will begin at the start of the pulse, not the end.)
WaitForRollPeak[(*t* sec)]	Sprint1 & Sprint2: Pulses bit 1 in Aux ROM. Sprint3: Pulses MS bit in steer ROM. Optionally delay effect for *t* seconds (*t* may be negative). Pulse width is one program step. For Sprint 1&2 function is equivalent to: Pulseflag(1b, 0s, *t* s). Sprint3 has no equivalent function. This is for use with the module that pauses the program until roll velocity crosses 0.
PulseBrake(*t1* sec [,*t2* sec])	Pulses bit 2 in Aux ROM. Pulse width is *t1* sec. Optionally delay effect for *t2* seconds (*t2* may be negative). This function is equivalent to: Pulseflag(2b, *t1* s, *t2* s) This is for use with the Test Control Module and Brake Actuator.
RepeatProgram	Immediately resets the program address counter so program continues from the beginning. No commands that follow this will be executed.
End	Ends the program with an address counter inhibit command. Current steer is held until address counters are reset. No commands that follow this will be executed.

```
; sample program #1

; Note that comments are indicated by a preceding semicolon.
; Anything after a semicolon is ignored on that line.

; Blank lines and extra spaces are also allowed and ignored.
; Upper/lower case is unimportant.

; All noncomment/nonblank lines must start with a valid function command,
; followed by any necessary parameters in parentheses, separated by commas.
; parameters must be a correctly formatted, followed by at least the first
; character of the corresponding parameter id
; parameter id's are:
;      hz  (for clock rate and sine/square/triangle frequency)
;      degrees  (for steer amplitude)
;      seconds  (for time lengths)
;      cycles   (for auto-calculated time lengths of sine/square/triangle)
; Note that cycles (if nonzero) supersedes seconds if both are present.
; Also note that amplitudes are +/-, not peak-to-peak for sine, triangle,
; and square waves
;

; default rate=1024hz, position=0deg, time length=0sec, 0 cycles, freq=0hz

; examples:

maxsteer(100 deg)      ; a maxsteer line like this must appear before "begin".
                       ; the ROM is actually programmed with all steer values
                       ; set to a percentage of this value (e.g. 60 deg = 60%
                       ; of 100 deg). If this program is run with the command
                       ; module set to 200 deg, all actual steer angles will
                       ; be twice those indicated here.
begin
setrate(512 hz)        ; valid entries for rate are 256,512,1024,2048
wait(1s)               ; stay at 0 for 1 second
StepTo(60d)            ; jumps to 60 degrees as fast as possible
wait(1s)               ; wait 1 sec. Note this could be added to above line
StepUp(30deg, 3 sec)   ; increase 30 deg as fast as possible and stay for 3sec
                       ; note use negative numbers to step or ramp down
rampto(50 deg,0.8s)    ; ramps down to 50 degrees in 0.8 seconds
wait(1.2s)             ; stay for 1.2 sec - can't add this to the rampto line
RAmpUp(-35d,1s)        ; ramps down 20 degrees (from 50) in 1 second
sine(80d,1 hz,2sec)    ; 1 hz sine wave with amplitude=+/-80 deg for 2 sec
                       ; centered wherever it starts
sine(50d,2 hz,2cycles) ; 2 hz sine wave with amplitude=+/-50 deg for 2 cycles
                       ; note cycles will supercede sec if both are present
SetCurrentHigh
square(40d,2.4 h,4s)   ; 2.4 hz square wave amplitude=+/-40 deg, hold 1/2
                       ; cycle where it starts then jumps up 40 for 1/2 cycle
setcurrentlow(-2s)     ; low current for last 2 sec of above square wave
triangle(32d,1h,3s)    ; 1 hz triangle wave amplitude=32,
                       ; centered wherever it starts
end                    ; this is added automatically if not present. It is
                       ; ignored if preceded by a valid Recycle command (see
                       ; program #1) end inhibits the count so the program
                       ; counters will not recycle.
```

```
; Sample program #2
; This example is a 0.2 hz sine wave made continuous by using the recycle
; function. An equivalent sine function would have 5 sec instead of 1 cycle.
; The End command should still be present, but has no clock inhibiting effect.

Maxsteer(100deg)
begin
 setrate(2048h)
 sine(80deg,0.2hz,1cycle)
 repeatprogram
end
```

```
; Sample program #3
; This example is a 0.2 hz 0 to 80 degree continuous triangle wave.
; Note the quarter cycle at the beginning to offset the center.

Maxsteer(100deg)
begin
 setrate(2048h)
 triangle(40deg,0.2hz,0.25cycle)  ; could also use rampto(40d,1.25s)
 triangle(40deg,0.2hz,0.75cycle)  ; ±40 deg centered at +40 deg
 repeatprogram
end
```

Program Editor

The editor screen is broken up into two main sections: the text editor screen and the graph. There is also a main menu, a status bar, and several function buttons.

The graph shows a strip-chart representation of the program. Its display is only updated by clicking on “Show” with an error-free program. If errors are found the compilation will be aborted and the plot will not occur. When the Windows pointer is over the graph it becomes a crosshair, the center coordinates of which are displayed on the right side of the status bar.

If the option to view the Aux ROM bits is enabled, a separate overlay is generated for each enabled bit using the selected color. The display will be updated whenever the viewing selection is changed. It is not necessary to click “Show” again unless the program has been changed. These bits are on/off, so they are displayed such that “on” is near the top of the graph window and “off” is near the bottom. The y-axis scale is not relevant to the Aux ROM bits.

The color of the steer program graph has different meanings for the status of the data at the time position:

- Black: Normal, low current operation.
- Red: High current. Affects Sprint 1 machines only.
- Yellow: Program inhibited. The graph displays the entire time available in the program given the step rates. All time positions after the End command are inhibited as indicated by the yellow portion of the graph. These points are always low current and are always at the same steer angle as the End point.
- Blue: Program has restarted. As with yellow, the graph displays the entire time available in the program. The actual data in the file is actually at the same steer angle as the RepeatProgram point, but the blue graph indicates the operation has recycled by repeating the waveform.
  A nonzero end to the graph does not indicate a discontinuity in the program. Any discontinuity will be seen at the black (or red) to blue transition point. A warning box will appear when the program is compiled if this is the case.
  Note that while this section of the graph is always blue, the high/low current pattern is repeated, just as in the black/red portion.

The editor screen displays the text of one of the sixteen programs in the program set. The program displayed can be selected by clicking on the notebook tabs at the bottom of the editor screen.

The text editor allows the standard Windows editor commands for blocking text, copying to clipboard (ctrl-c), cutting to clipboard (ctrl+x), and pasting from clipboard (ctrl+v), etc. There are currently no commands available for searching for or replacing text. The status bar also shows the current cursor position (line and column numbers).

The following buttons are provided on screen for the most-used functions:

Open: Replaces the current program set with one from a .ROM file on disk.

Import: Optionally inserts program data or replaces the currently displayed program with the contents of a file on disk. A dialog will appear asking for specifics. If the Insert radio button is selected the new data will be inserted at the current cursor position in the program; otherwise the current program will be cleared before inserting the new data. The way the file is handled depends on the type of file:

Text: A text file (.TXT) will be copied directly into the current program.

ROM: The selected program in the .ROM file will be copied directly.

Time & Value Ascii: The .TXT file will be converted to a program having the specified step rate and max steer, and a series of rampto and wait commands. The program will be as close a match as possible to the input, given the time and steer resolutions available, and using linear interpolation between given points. Any steer in the input file greater than max steer will be truncated at max steer. It is possible to exceed the maximum program length of 32K in which case the operation will abort with an error message.

Export: Allows the current program to be exported as a text file or a time & value ascii file.

Save: Saves the program set under the current name unless it is untitled, in which case it first asks for a name.

Save As: Saves the program set under a new name.

Show: Compiles the currently displayed program. If successful, the resulting data is displayed on the graph. If unsuccessful, error messages will pinpoint any problems found.

Build: Compiles all programs in the current program set. If successful, the binary EPROM files for the steer ROM (.BIN) and the Aux ROM (.AUX) are created.

The **File** drop-down menu includes the **Open**, **Save**, and **Save As** items as described above as well as the following:

New: Clears all programs in the current program set.

Preferences:

Default Directories: Allows you to specify default directories for reading and saving .ROM, .TXT, and binary (.BIN and .AUX) files. Clicking on “OK” in the dialog will save these options to a file in the program directory.

Sprint Machine version: Used to set the Sprint machine version for which EPROM files are to be built. “Sprint 1. 2, 3 compatibility” will create EPROM files that can be used in all machines. If used in a Sprint 3 machine the compatibility switch on the Program module must be set correctly. “Sprint 3 compatibility only” will create EPROM files that may only be used in Sprint 3 machines, and then only if the compatibility switch on the Program module is set correctly. Sprint 3 machines using this mode can utilize roll rate feedback without the need for an AuxRom.

There is actually no difference between the two modes if the WaitForRollPeak function is not used in any program in the EPROM.

Editor Font: Allows you to change the font and size used in the editor. If the font selected is available to the printer (e.g. Truetype) this font will also be used for printing.

Print: Prints either the current graph or the program text. First a dialog will open asking which to print, then another dialog for printing options and setup. Both Graph and text printing are full page, either portrait or landscape as selected in print setup, without options for margin changes as yet.

Print Setup: Opens a standard print setup dialog.

The **Edit** drop-down menu includes items for cutting, copying, and pasting text to the clipboard as well as the **Clear** function which erases the currently visible program only.

The **View** drop-down menu allows access to the following options for viewing the bits in the Auxiliary ROM:

Display Selected Aux ROM Bits: Enables the display of all selected bits when checked. The graph display will immediately be updated to reflect view changes.

Select Aux ROM Bits to Display: Opens a dialog allowing selection of individual bits to display as well as the color used to draw each. The graph display will immediately be updated to reflect changes.

The **Help** item on the main menu displays this file in a help format. Other help buttons in dialogs are currently inactive.

The **About** item on the main menu brings up a dialog with program version and contact information.

PROGRAMMING RULES FOR MOTOR HEATING CONSIDERATIONS

All programs must be designed to avoid violation of the limits in the Pulse Duration/Duty Cycle graph found under Motor Heating Considerations in section 9 of the manual.

In Sprint 1 machines, the servoamplifier output current will not automatically fold back in reverse steer operation (e.g. sine wave input). Special consideration is taken in programming to only allow maximum current for short periods of time. The maximum motor current, selected using the SetCurrentHigh command, is to be used only for high rate steer transitions (e.g. from 0.2 sec before thru 0.2 sec after a transition). The software enforces a limit of 5 seconds of high-current in any 15 second period. While this enforcement is useful, it is still possible to create a repeating program of less than 15 seconds that violates the rule without a compiler error. This will violate the motor limits and should not be done.

For Sprint2 and Sprint3 machines the SetCurrentHigh and SetCurrentLow commands have no effect.

Note that it is possible to violate the motor limits in the Sprint2 and Sprint3 machines as well as in Sprint1 even without using SetCurrentHigh in the program at all, especially with the current level selection set too high (see the discussion in the service manual).

Batronix BX40 Bagero quick reference for use with Steering Machine

---

Note: The Batronix BX40 currently ships with new machines (2008 to present), Prior to this different models from Needhams Electronics were used, but they closed in 2007. Needhams model EMP-100 (USB, windows software) was briefly used in 2007. Model EMP-111 (parallel port with windows software) was used from 2002 through 2006. Model EMP-10 (parallel port with DOS only software) shipped with new machines prior to year 2002. If you are using an EMP-10, EMP-11, or EMP-100 then please see the notes on the following pages.

Please see the Batronix documentation for initial installation instructions.

Run the Batronix Prog-Express program. It should automatically find the programmer, which is USB powered. The program will start up in the same mode it was last used, normally "Program Chip" mode.

Programming steps:

1. If the program is not already at the "program Chip" screen then select the program option to "Program a chip with data from a file".
2. Select "Please choose a chip". Then find the correct chip under EPROM in the chip browser. The manufacturer and part number are printed on the chip (see reference chips below). Sprint 3 machines ship with ST (SGS Thomson) part 27C4002-10F1 chips. To select this chip, browse the list:
   Eprom => ST (SGS Thomson => M27C4002 => Dip Version => 10F1
3. Next "Choose an existing file": browse for and select the desired file. Romsteer uses the ".bin" extension for 16-bit steer roms, and an ".aux" extension for 8-bit Auxroms. The .aux file is always created, but the AuxROM should not be needed when using Sprint 3, only the .bin file is used.
4. Please do not set any Chip options or serial numbers
5. Insert the EPROM and click on "Start program process" to begin the programming sequence.
   The file will be loaded into a buffer; the chip type will be checked; the EPROM will be blank-checked, then programmed and verified. Any errors will be displayed (e.g. "Chip isn't empty" if blank check fails).
6. Remove the EPROM and continue with the next copy or next file one if desired.
7. Please remove any EPROM from the programmer before exiting program or disconnecting the cable.

Some EPROM part numbers:

Manufacturer	**Steer ROM: 256k x 16 bit**	**Aux ROM: 256k x 8 bit**
Advanced Micro Devices (AMD)	AM27C4096	AM27C020
Atmel	AT27C4096	AT27C020
Hitachi	HN27C4096	?
Macronix	MX27C4002	?
National Semiconductor (NS)	NM27C210	NM27C020
ST Microelectronics (SGS-Thompson)	**ST27C4002**	ST27C2001
Texas Instruments (TI)	TMS27C240	TMS27C020
Toshiba	TC574096D	?

Needham's EMP-11 or EMP-100 quick reference for use with Steering Machine

---

Note: Model EMP-11 (parallel port with windows software) was used from 2002 through 2006. Model EMP-100 was briefly used in 2007 before Needhams closed. Model EMP-10 (parallel port with DOS only software) shipped with new machines prior to year 2002. Current machines ship with a Batronix "BX40 Bagero" programmer. If you are using an EMP-10 then please see the notes on the following page. If you have the Batronix BX40 then please see the notes on the preceding page.

Run the Needham's Electronics Device Programming Software program.
It should find the correct printer port by itself unless a nonstandard port is used.

Note there is no power switch on the programmer. It is turned on by the software and off again when the program exits. The LED is lit only when on, not when simply plugged in.

If no family module is inserted the program will ask for the last one used to be inserted before it will continue.

Programming steps:

8. Select the Device button.
   Under "Device Type" select EPROM so that devices are easier to find on the list.
   Then select the manufacturer and device number. If a different family module is required, a bop up dialog will request that it be installed. (usually 111B for 16-bit EPROMs; 111A for 8-bit EPROMs).
9. Using the File menu, select "Open" and browse to find the necessary file. Romsteer uses the ".bin" extension for 16-bit steer roms, and an ".aux" extension for 8-bit Auxroms.
10. Click "ok" to preload the buffer with an erase value and accept the default buffer size.
11. Insert EPROM. It's best not to insert it until the software is running and ready to read or program it.
12. Click on the "Blank Check" button. Besides checking that the EPROM is blank, this should assure that the EPROM was inserted correctly.
13. Click on the "Program" button to write the data to the EPROM. Another blank check will occur, followed by a program cycle and a verify cycle.
14. Remove EPROM before exiting program.

Note that on the right side of the screen there are settings for buffer range and device range as well as "splits" and "sets".
Splits and Sets should both default to 1:1.
The device range will default to 000000000 to 0003FFFF for both 8-bit and 16-bit EPROMs.
The buffer range will be 000000000 to 0003FFFF for the 8-bit Aux EPROM and 00000000 to 0007FFFF for the 16-bit EPROM. These defaults are correct for the machine EPROMs.

Some EPROM part numbers:

Manufacturer	**Steer ROM: 256k x 16 bit**	**Aux ROM: 256k x 8 bit**
Advanced Micro Devices (AMD)	AM27C4096	AM27C020
Atmel	AT27C4096	AT27C020
Hitachi	HN27C4096	?
Macronix	MX27C4002	?
National Semiconductor (NS)	NM27C210	NM27C020
ST Microelectronics (SGS-Thompson)	ST27C4002	ST27C2001
Texas Instruments (TI)	TMS27C240	TMS27C020
Toshiba	TC574096D	?

Needham's EMP-10 quick reference for use with Steering Machine

---

Note: The EMP-10 (parallel port with DOS only software) shipped with new machines prior to year 2002. Model EMP-11 (parallel port with windows software) was used from 2002 through 2006. Model EMP-100 was briefly used in 2007 before Needhams closed. Current machines ship with a Batronix "BX40 Bagero" programmer. If you are using an EMP-11, EMP-100, or BX-40 then please see the notes on the preceding pages.

Install software in DOS:
From A:> prompt **install emp10 a: c:**

Run software by changing to c:\EMP10 directory and typing: **emp10**
It should find the correct printer port by itself unless a nonstandard port is used.

Note there is no power switch on the programmer. It is turned on by the software and off again when the program exits. The LED is lit only when on, not when simply plugged in.

Programming steps:

15. Select Device (#5 on menu). The table will indicate the correct family module that must be installed (M2A for 16-bit EPROMs; M1A for 8-bit EPROMs).
16. Set filename (V on menu). Type in the path and filename or use [F1] key to browse for the file.
17. Load file (#8 on menu). Defaults for 16-bit EPROMs should be correct for buffer range (0,7FFFF) and file type (binary).
18. Insert EPROM. It's best not to insert it until the software is running and ready to read or program it.
19. Verify Device erased (#2 on menu). Besides checking that the EPROM is blank, this should assure that the EPROM was inserted correctly.
    Note that for the 27C4096 the counter will increment to 400 hex, indicating 40000 16-bit data words checked (or 0 thru 7FFFF hex bytes of data). I thought at first that it was not reading/writing the whole EPROM.
20. Program device (#1 on menu). The defaults should be correct for buffer range and device range (0,7FFFF), split (1), and set (1). The counter will again increment to 400 hex and will then automatically run a verify cycle.
21. Remove EPROM before exiting program.

Note for the 8-bit Aux EPROMs the buffer range will be (0,3FFF) and will still increment to 400 hex.

Some EPROM part numbers:

Manufacturer	**Steer ROM: 256k x 16 bit**	**Aux ROM: 256k x 8 bit**
Advanced Micro Devices (AMD)	AM27C4096	AM27C020
Atmel	AT27C4096	AT27C020
Hitachi	HN27C4096	?
Macronix	MX27C4002	?
National Semiconductor (NS)	NM27C210	NM27C020
ST Microelectronics (SGS-Thompson)	ST27C4002	ST27C2001
Texas Instruments (TI)	TMS27C240	TMS27C020
Toshiba	TC574096D	?