TLdemo

Ken Shillito shillito@tpg.com.au

COLLABORATORS

	TITLE : TLdemo	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Ken Shillito shillito@tpg.com.au	August 8, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

TLdemo

1.1 TLdemo Guide

```
                TLdemo.guide Version 0.02  March 25, 2000

© 1999      Ken Shillito    Winchelsea     Australia

                            Email:       shillito@tpg.com.au
                            Home page:   www2.tpg.com.au/users/shillito

This guide is a demo of the capabilties of tandem.library, which is an
optional adjunct to Tandem, which you can use if you want to.


                OK - I'll look at it
```

1.2 OK TLdemo.guide - a Demo of the Capabilites of Tandem.library

```
                A Demonstration of tandem.library's Capabilities

The point of this guide is to demonstrate the tandem.library GUI - what
it can do, and how easy it is to program.

The guide is set out as a linked set of programs. I say a little about each
program. You may then press a button to see the program operate, and another
button to see its source code.


                Go to the First Program
                (You can also click "Index" to jump to particular programs, rather ↩
                    than
 wading through the linked list).

n.b. The source codes are taken from the Tandem/Teaching drawer. You can
also see the source codes there, with more detailed comments.
```

1.3 xxx

```
                Index to Demo.guide

A. Sorted by Number

                IncAll.i
                TLerror
                TLreqshow I
                TLputilbm
                 Scrollrs II


                TLgetfont TLnewfont
                TLassdev
                TLprogress
                TLellpse II

                Front.i
                TLtsize TLwcheck
                Window Suite
                TLdata
                TLtabmon

                TL CLI Routines
                TLaslfont TLwupdate
                TLmultiline
                TLreqshw III
                TLslimon


                TLreqchoose
                TLfloat
                TLresize
                TLbutmon


                TLreqinput
                TLbusy,unbsy
                TLEllipse I
                TLprefs

                TLwindow
                TLaslfile
                TLreqshow II
                TLgetarea
                Custom Reqs

                TLstring TLtrim
                TLreqmenu
                TLreqcolor
                TLreqedit
                Help,Guide

                TLreqarea TLreqbv
                TLreqinfo I
```

Reqstr hooks
TLreqfont
TLpassword

TLkeyboard
TLreqinfo II
TLgetilbm
TLscrollrs I
TLdropdown
B. Sorted by Featured TL routine

TLaschex

TLinput

TLreqchek

TLtabs

TLaslfile

TLkeyboard

TLreqchoose

TLtext

TLaslfont

TLmget

TLreqcls

TLtrim

TLassdev

TLmultiline

TLreqcolor

TLtsize

TLbusy

TLnewfont

TLreqedit

TLunbusy

TLbutmon

TLoffmenu

TLreqfont

TLwcheck

TLbutprt

TLonmenu

TLreqfull

TLwclose

TLbutstr

TLopenread

TLreqinfo

TLwfront

TLbuttxt

TLopenwrite

TLreqinput

TLwindow

TLchip

TLoutput

TLreqmenu

TLwpoll

TLclosefile

TLpassword

TLreqmuclr

TLwpop

TLdata

TLpict

TLreqmuset

TLwritefile

TLdropdown

TLprefdir

TLreqoff

TLwscroll

TLellipse

TLpreffil

TLreqon

TLwslof

TLerror

TLprefs

TLreqredi

TLwsub

TLfloat

TLprogdir

TLreqshow

wupdate

TLfsub

TLprogess

TLresize

TLgetarea

TLpublic

TLslider

TLgetfont

TLputilbm

TLslimon

TLgetilbm

TLreadfile

TLstra0

TLhexasc

TLreqarea

strbuf

TLhexasc16

```
                TLreqbev

                tabmon
```

1.4 p21 Demonstrate the First Program (called 21)

```
                Demonstrate the First Program  (called "21")

First, click this to demonstrate the first program, program 21. All its does
is beeps the screen.

   Run Program 21        <- Beep the screen

                Source for Program 21
                 <- Tandem/Teaching/21.asm


Now, click this button, if you don't understand assembly language:


                Help, Help, I can't understand Assembly Language
                (After you read what's below, click this button, or the button at  ↩
                   the end)


                Forward to the second program (Program 23)
                Read on if you can understand assembly language...

As you just saw, program 21 merely beeped the screen. The interesting
thing is not so much the program, as its source code. Click the above
"Source for Program 21" button and read the complete source code.

Complete? But what about

   XREF _AbsExecBase
   XREF _LVOOpenLibrary, etc.

What about an INCLUDE of a header file or two?

Well, that's the point of the line:

   INCLUDE 'IncAll.i'

That line is actually a feature of Tandem, not tandem.library - if you
assemble the program with Tandem, then IncAll.i includes all
the Amiga OS3.1 INCLUDE's, and all the Amiga.FD's - all
instantly assembled! You don't need XREF's for all your _LVO's, and you
don't need INCLUDE's of the Amiga OS3.1 header files. You can have them if
you want, but you don't need them. All you need is  INCLUDE 'IncAll.i'.

(Actually, the program would be still more compact if you used all the
 features fo tandem.library - the above introduces IncAll.i, and further
 examples will introduce the other hackwork-saving features of
 tandem.library).
```

Forward to the second program (Program 23)

1.5 cant

Help, Help, I don't Understand Assembly Language"

Stick around - you soon will!

Here's what you should do:

First, go through all the nodes of this guide, clicking the buttons which run programs 21 through 72. Ask yourself - would you like to be able to write programs to do what these buttons do? If yes, then:

1. Close this AmigaGuide.

2. Open the AmigaGuide whose icon is in the Tandem drawer, called

   Tandem.guide

3. In that guide, you'll see a button on the contents page:

   Learn Assembly Language

4. Click that button, and Tandem.guide (assisted by Tandem itself) will teach you assembly language in an interactive fashion.

Forward to the next demonstration program (Program 23)

1.6 21asm

Forward to program 23

```
* 21.asm    introduce IncAll.i    version 0.00   1.9.97

 bra Start                ;go to cold start

 include 'IncAll.i'       ;get all Amiga OS3.1 FD's and .i's

intname: dc.b 'intuition.library',0
 ds.w 0
intbase: ds.l 1

Start:
 move.l _AbsExecBase,a6   ;open intuition.library
 lea intname,a1
 moveq #37,d0             ;at least release 2.04
 jsr _LVOOpenLibrary(a6)
```

```
 move.l d0,intbase

 beq.s Abort              ;go if can't open

 move.l intbase,a6        ;beep all screens
 sub.l a0,a0
 jsr _LVODisplayBeep(a6)

 move.l a6,a1             ;close intuition library
 move.l _AbsExecBase,a6
 jsr _LVOCloseLibrary(a6)

 clr.l d0                 ;quit good
 rts

Abort:
 moveq #-1,d0             ;quit bad
 rts


                Forward to Program 23
```

1.7 p23 Demonstrate Program 23

```
                Program 23

Program 23 merely prints a greeting on the CLI window:

   Run Program 23       Say, "Hello, world"

                Source of program 23
                 -> Teaching/23.asm

Program 23 features the TL routines "TLStrbuf" & "TLOutput".


After you read what's below, click this button, or the button at the bottom:


                Forward to Program 24
                Program 23 (above) instroduces the instruction:

   INCLUDE 'Front.i'

Front.i includes within itself "INCLUDE 'IncAll.i'", so you thus have all
the features of IncAll.i

But Front.i also sets up a 1024 byte workspace, of which the first 512
bytes are a general purpose scratchpad called the "buff" or "buffer".
The rest of the workspace contains many other useful things.
To INCLUDE Front.i, you must have "strings DC.B 0" in your program, and
also make the entry point your program a subroutine called "Program".
All calls to tandem.library must have A4 pointing to the 1024 byte
workspace set up by Front.i
```

```
TLStrbuf transfers one of "strings" to "buff", while TLOutput sends whatever
is in "buff" to the CLI.

Click the "Source of program 23" button above, to see the complete
source code, to show you how simple this is.


                Forward to program 24
```

1.8 23asm

```
                Forward to program 24
                * 23.asm      Introduce Tandem's standard frontend    version 0.00 ↩
                      1.9.97

; This program simply puts a message on a CLI (Shell) window. When you
; run this program, you will see the message appear.


 include 'Front.i'          ;opens libraries etc and BSR's Program

* text strings
strings: dc.b 0             ;Front.i requires this line
 dc.b 'Hello, world',0      ;string 1
 ds.w 0                     ;re-align mc

* print a hello message
Program:                    ;Entry point is Program, A4 points to xxp_tndm
 move.l xxp_tanb(a4),a6     ;get tanbase (the base of tandem.library)
 moveq #1,d0
 jsr _LVOTLStrbuf(a6)       ;tfr string 1 to xxp_buff
 jsr _LVOTLOutput(a6)       ;ouput xxp_buff to output window
 rts                        ;quit
```

1.9 p24 Demonstrate Program 24

```
                Program 24

   Run Program 24       -> do various CLI things.

                Source of Program 24
                 -> Teaching/24.asm

When programs call tandem.library, I recommend they use the MACROs in
IncAll.i to do so; there is a MACRO for each routine, with the same name
but the 1st character lower case. I call tandem.library routines TL routines
```

```
for short.


Click this button, or the button at the end, when you've read the text below


                Forward to Program 27
                The program below demonstrates the following elementary TL  ↩
                   routines:

TLpublic      get public memory          TLoutput  write buff to CLI
TLchip        get chip memory            TLinput   read from CLI to buff
TLopenread    open a file for reading    TLoutstr  tfr string to buff
TLopenwrite   open a file for writing    TLoutstr  combines TLstrbuf,TLoutput
TLreadfile    read from open file        TLhexasc  convert hex to ASCII
TLwritefile   write to open file         TLaschex  convert ASCII to hex
TLclosefile   close file

24.asm still uses the scabby CLI interface - but soon we'll be using
Amiga's intuition windows.

(you can't see most of what this program does - look up its
 source to see the way it calls the above TL routines).

The above are simple routines that merely call amiga ROM kernel routines,
but do so in a convenient way. tandem.library keeps track of all resources
your program uses, and Front.i frees them if you haven't already done so.
(e.g. you don't need to close files or free memory).


                Forward to program 27
```

1.10 24asm

```
                Forward to program 27
                * 24.asm    Program to demonstrate tandem.library    version 0.00  ↩
                    1.9.97

 include 'Front.i'        ;*** change to Tandem.i to step thru TL's ***

; The program below conducts a conversation through the CLI window.
; The program shows how you can use TL routines to do some basic tasks, and
; and illustrates a primitive user interface via the CLI window.


* message strings
strings: dc.b 0
 dc.b $0C,'Hello, CLI window',0 ;1
 dc.b 'RAM:Temp',0 ;2
 dc.b 'Out of memory: no public memory',0 ;3
 dc.b 'Out of memory: no chip memory',0 ;4
 dc.b 'Everything worked ok',0 ;5
 dc.b 'Input an unsigned integer (less than 2 billion)',0 ;6
 dc.b '(All done: Press <Return> to acknowledge)',0 ;7
```

```
 dc.b 'Read/Write error',0 ;8
 dc.b 'You input:',0 ;9
 dc.b '10,000 bytes of public mem created OK.',0 ;10
 dc.b 'Now, I''ll save string 12 to RAM:Temp, The MACROs are:',0 ;11
 dc.b 'This is string 12 (34 bytes long)',0 ;12
 dc.b '  TLstrbuf #2        (tfr string 2, i.e. ''RAM:Temp'', to buff)',0
 dc.b '  TLopenwrite        (open RAM:Temp)',0 ;14
 dc.b '  TLstra0 #12        (point a0 to string 12)',0 ;15
 dc.b '  TLwritefile a0,#34 (write 34 bytes [= len of string 12] to file)',0
 dc.b '  TLclosefile        (close file)',0 ;17
 dc.b 'Now, I''ll read RAM:Temp back in back again. The MACROs are:',0 ;18
 dc.b '  TLstrbuf #2        (tfr string 2, i.e. ''RAM:Temp'', to buff)',0
 dc.b '  TLopenread         (open RAM:Temp)',0 ;20
 dc.b '  TLreadfile a4,#76  (read up to 76 bytes to buff)',0 ;21
 dc.b '  TLclosefile        (close file)',0 ;22
 dc.b '  TLoutput           (send string 12 to CLI)',0 ;23
 dc.b 'Now, I''ll get a message from you. The MACROs are:',0 ;24
 dc.b '  TLoutstr #6        (prompt)',0 ;25
 dc.b '  TLinput            (get input from user)',0 ;26
 dc.b '  TLaschex a4        (get hex of number in buff to d0)',0 ;27
 dc.b '  move.l d0,-(a7)    (save d0)',0 ;28
 dc.b '  TLoutstr #9        (send string 9)',0 ;29
 dc.b '  TLhexasc (a7)+,a4  (convert input back to ascii in buff)',0 ;30
 dc.b '  clr.b (a0)         (delimit number)',0 ;31
 dc.b '  TLoutput           (echo number back to CLI window)',0 ;32
 dc.b '(Press <Return> to acknowledge)',$0A,0 ;33
 dc.b '10,000 bytes of chip mem created OK.',0 ;34
 dc.b 'RAM:Temp read & printed - string 12 s/be above this line.',0 ;35
 dc.b 0 ;36
 dc.b 'Now, I''ll issue the MACRO:   TLpublic 10000',0 ;37
 dc.b 'Now, I''ll issue the MACRO:   TLchip 10000',0 ;38
 dc.b 'RAM:Temp now created.',0 ;39

 ds.w 0

* memory pointers
pubmem: ds.l 1             ;pointer to 10000 bytes of public memory here
chipmem: ds.l 1            ;pointer to 10000 bytes of chip memory here

* program entry point - called by Front.i with A4 pointing to xxp_tndm
Program:

 TLoutstr #1               ;* greetings
 TLoutstr #36

 TLoutstr #37              ;* will create 10000 publilc
 TLpublic #10000           ;create 10000 bytes of public mem
 move.l d0,pubmem          ;remember where
 beq Pr_bad1               ;go if out of public mem
 TLoutstr #10              ;success
 TLoutstr #36

 TLoutstr #38              ;* success, will create 10000 chip
 TLchip #10000             ;create 10000 bytes of chip mem   } Front.i
 move.l d0,chipmem         ;remember where                   } does so
 beq Pr_bad2               ;go if out of chip mem            } automaitcally
 TLoutstr #34              ;success
```

```
TLoutstr #36

TLoutstr #33              ;* wait for acknowledge
TLinput

TLoutstr #11              ;send strings to CLI
TLoutstr #13
TLoutstr #14
TLoutstr #15
TLoutstr #16
TLoutstr #17
TLstrbuf #2               ;tfr 'RAM:Temp' to buff
TLopenwrite               ;open RAM:Temp for writing
beq Pr_bad3               ;go if can't
TLstra0 #12               ;point a0 to string 12
TLwritefile a0,#34        ;write 33 bytes (= len of string 12)
beq Pr_bad3               ;go if can't (TLwritefile closes file if bad)
TLclosefile               ;close the file.  RAM:Temp now exists
TLoutstr #39              ;success
TLoutstr #36

TLoutstr #18              ;send more strings to CLI
TLoutstr #19
TLoutstr #20
TLoutstr #21
TLoutstr #22
TLoutstr #23
TLstrbuf #2               ;tfr 'RAM:Temp' to buff
TLopenread                ;open RAM:Temp for reading
beq Pr_bad3               ;go if can't
TLreadfile a4,#76         ;read it back into buff (should read 33 bytes)
beq Pr_bad3               ;go if can't (TLreadfile closes file if bad))
TLclosefile               ;close the file; string 12 s/be at start of buff
TLoutput                  ;send string 12 (as saved and read) to CLI window
TLoutstr #35              ;success
TLoutstr #36

TLoutstr #33              ;wait for acknowledge
TLinput

TLoutstr #24              ;send strings to CLI
TLoutstr #25
TLoutstr #26
TLoutstr #27
TLoutstr #28
TLoutstr #29
TLoutstr #30
TLoutstr #31
TLoutstr #32
TLoutstr #33
TLoutstr #6
TLinput                   ;get user response to string 6

TLaschex a4               ;get hex of number in buff to d0
move.l d0,-(a7)           ;save d0
TLoutstr #9               ;send string 9
move.l (a7)+,d0           ;get num that was input
```

```
 TLhexasc d0,a4             ;convert d0 back to ascii in buff
 clr.b (a0)                 ;delimit number (TLHexasc points A0 past)
 TLoutput                   ;echo number back to CLI window
 bra.s Pr_good              ;go wrap up

Pr_bad1:
 moveq #3,d0                ;error message - string 3
 bra.s Pr_quit

Pr_bad2:
 moveq #4,d0                ;error message - string 4
 bra.s Pr_quit

Pr_bad3:
 moveq #8,d0                ;error message - string 8
 bra.s Pr_quit

Pr_good:
 moveq #5,d0                ;ok message - string 5

Pr_quit:
 TLoutstr d0                ;send closing message
 TLoutstr #7                ;ask for acknowledge
 TLinput                    ;wait for closing message
 rts                        ;back to Front.i to close down


                Forward to program 27
```

1.11 p27 Demonstrate Program 27

```
                Program 27

This program opens a screen & window, & waits for a response:

   Run Program 27    -> open screen & windows

                Source program 27
                 -> Teaching/27.asm

It demonstrates the the following TL routines:

   TLscreen        open a private screen
   TLwindow        open a window
   TLkeyboard      wait for IDCMP

Click this button, or the button at the end when you've read the text below


                Forward to Program 28
                TLscreen is optional; if you don't call it, TL routines operate on ↩
                    the
default public screen (i.e. usually the workbench).
```

```
tandem.library maintains a suite of up to 10 windows which it keeps track
of for you. You can choose borderless, with borders, or with scrollers,
or specify a custom set of WFLG_'s.

TLkeyboard makes getting IDCMP simple; as well as returning ASCII, it returns
codes for non-ASCII things like the function keys, window close, &c.

If you look at  Tandem/Teaching/27.asm  you will see how simple & compact
the TL routines make opening a screen & windows.


                Forward to program 28
```

1.12 27asm

```
                Forward to program 28
                * 27.asm      Use TL routines to open a window    version 0.00    ↩
                   1.9.97

 include 'Front.i'

strings: dc.b 0
st_1: dc.b 'My Screen',0 ;1
st_2: dc.b 'My Window',0 ;2

 ds.w 0

dpen: dc.l -1              ;default pens structure

* open screen & window; close & exit when close gadget clicked
Program:
 TLscreen #2,#st_1,#dpen   ;open screen: 2 planes, title st_2, pens dpen
 beq Pr_bad                ;go if can't
 TLwindow #0,#20,#10,#100,#20,#400,#150,#0,#st_2
                           ;open window 0
                           ;posn 20,10  minsize 100,20  maxsize 400,150
                           ;flags: #0=Front.i's default (i.e. HIRES)
                           ;title st_2
 beq Pr_bad                ;go if can't
Pr_wait:
 TLkeyboard                ;get IDCMP
 cmp.b #$93,d0             ;close window?
 bne Pr_wait               ;no, keep waiting
 rts                       ;Front.i closes everything
Pr_bad:
 move.l xxp_intb(a4),a6
 sub.l a0,a0
 jsr _LVODisplayBeep(a6)   ;if bad, beep existing screens
 rts


                Forward to program 28
```

1.13 p28 Demonstrate Program 28

```
                Program 28

   Run Program 28    -> write text on a window

                Source program 28
                 -> Teaching/28.asm

This program demonstrates TLtext, TLtrim, and TLstring, which are used
for writing text on the window.


                Forward to program 29
```

1.14 28asm

```
                Forward to program 29
                * 28.asm    Demonstrate TLstring, TLtrim      Version 0.01     ↩
                   8.6.97

; This program prints a message to a custom window, using TLstring.
; (this program doesn't refresh).

; You will note in the program below that you do not need to close
; windows or other things created by tandem.library calls. Front.i calls
; TLWclose when you exit from Program, which automatically releases
; all resources used by tandem.library calls. This makes programming
; more convenient.


 include 'Front.i'


strings: dc.b 0
st_1: dc.b 'Demonstrate TLstring & TLtrim',0 ;1
 dc.b 'Hello, Intuition',0 ;2
 dc.b '28.asm failed: Out of memory',0 ;3
 dc.b '(I don''t refresh - resize the window & see what I mean.)',0 ;4
 dc.b 'Click the window close gadget when finished.',0 ;5
 ds.w 0


* open window & print message; close & exit when close gadget clicked
Program:
 TLwindow #0,#20,#10,#80,#50,#500,#150,#0,#st_1 ;open window 0
 beq Pr_bad

 TLstring #2,#4,#2         ;show string 2 at (4,2)
```

```
 TLstring #4,#4,#16        ;show string 4 at (4,16)
 TLstring #5,#4,#30        ;show string 5 at (4,30)

Pr_wait:
 TLkeyboard                ;get any input
 cmp.b #$93,d0             ;close gadget?
 bne Pr_wait               ;no, keep waiting
 bra.s Pr_quit             ;return ok  (Front0.i closes everything)

Pr_bad:
 TLerror                   ;TLerror sends error reports to the monitor
 TLbad #3                  ;"TLbad #3" makes the monitor wait for
                           ;  acknowledge before closing, with
Pr_quit:
 rts
```

Forward to program 29

1.15 p29 Demonstrate Program 29

```
                Program 29

   Run Program 29    -> draw rectangles & plain/bevelled boxes

                Source program 29
                 -> Teaching/29.asm
```

This program demonstrates TLreqarea, which colours in a rectangle on the window, and TLreqbev, which makes plain or bevelled boxes on the window. You will note that these routines trim the box/area if it doesn't fit on the current window size. Also, tandem.library MACROs have parameters which position relative to the area inside the window borders, which is very convenient.

Forward to program 30

1.16 29asm

Forward to program 30

```
* 29.asm    TLReqarea,TLReqbev   version 0.01    2.10.98


; TLReqarea & TLReqbev "clip" boxes & area which go past the printable
; area of the window.

; The below are examples of a new art form - making artistic combinations
; of bevelled boxes. Note that the plain rectangles have double width
```

```
; sides, which look better than single width.

 include 'Front.i'

strings: dc.b 0
st_1: dc.b 'Demonstrate TLReqarea & TLReqbev',0 ;1
 dc.b 'Error: out of chip memory',0 ;2
 dc.b 'A plain bevelled box   A recessed bevelled box   A plain rectangle',0
 dc.b 'A bevelled box with custom pens   A rectangle with custom pen',0 ;4
 dc.b 'A plain bevelled box filled with pen 3    40X1  1X15  2X15',0 ;5
 dc.b 'Clipped boxes',0 ;6
 dc.b 'Combinations (Art?) ... ',0 ;7

 ds.w 0


* test program
Program:
 TLwindow #0,#0,#0,#640,#200,#640,#200,#0,#st_1 ;open window 0
 bne.s Pr_cont
 TLbad #2                  ;report if can't open window
 rts

Pr_cont:
 TLstring #3,#6,#3                 ;top three boxes
 TLreqbev #66,#18,#40,#15          ;  plain
 TLreqbev #258,#18,#40,#15,rec     ;  recessed
 TLreqbev #450,#18,#40,#15,box     ;  rectangle

 TLstring #4,#6,#44                ;second row of boxes
 TLreqbev #106,#59,#40,#15,,,#3,#6 ;  bev, pens 5,6
 TLreqbev #366,#59,#40,#15,box,,#3 ;  rect, pen 3

 TLstring #5,#6,#85                ;third row
 TLreqarea #130,#100,#40,#15,#3    ;  fill with pen 3
 TLreqbev #130,#100,#40,#15        ;  plain bev
 TLreqbev #342,#100,#40,#1,box     ;  40X1
 TLreqbev #392,#100,#1,#15,box     ;  1X15
 TLreqbev #440,#100,#2,#15,box     ;  2X15

 move.l xxp_AcWind(a4),a5          ;clipped boxes
 moveq #0,d6
 moveq #0,d7
 move.w xxp_PWidth(a5),d6
 move.w xxp_PHeight(a5),d7
 sub.w #104,d6
 sub.w #40,d7
 TLstring #6,d6,d7
 add.w #84,d6
 add.w #12,d7
 TLreqbev d6,d7,#40,#15            ;clipped horz
 add.w #20,d7
 TLreqbev d6,d7,#40,#15            ;clipped both
 sub.w #46,d6
 TLreqbev d6,d7,#40,#15            ;clipped vert

 TLstring #7,#6,#126               ;fourth row
```

```
TLreqbev #6,#136,#60,#30          ;  1st
TLreqbev #7,#137,#58,#28
TLreqbev #76,#136,#60,#30,rec     ;  2nd
TLreqbev #77,#137,#58,#28,rec
TLreqbev #146,#136,#60,#30        ;  3rd
TLreqbev #148,#137,#56,#28,rec
TLreqbev #216,#136,#60,#30,rec    ;  4th
TLreqbev #218,#137,#56,#28
TLreqarea #286,#136,#60,#20,#3    ;  5th
TLreqbev #286,#136,#60,#20
TLreqarea #292,#139,#48,#14
TLreqbev #292,#139,#48,#14,rec
TLreqarea #286,#161,#60,#20,#3    ;  6th
TLreqbev #286,#161,#60,#20
TLreqarea #292,#164,#48,#14
TLreqbev #292,#164,#48,#14,rec
TLreqbev #293,#165,#46,#12,rec
TLreqarea #366,#141,#40,#20,#3    ;  7th
TLreqbev #366,#141,#40,#20
TLreqarea #376,#146,#40,#20
TLreqbev #376,#146,#40,#20
TLreqbev #366,#141,#32,#16,box
TLreqarea #356,#136,#40,#20
TLreqbev #356,#136,#40,#20

TLkeyboard                        ;wait for response
rts
```

Forward to program 30

1.17 p30 Demonstrate Program 30

Program 30

Run Program 30 -> TLkeyboard

Source program 30
-> Teaching/30.asm

This program waits while you type various sorts of things on the keyboard, and resize the window, click the mouse &c., and shows what TLkeyboard returns in D0,D1,D2 and D3. This shows the convenience and flexibility of TLkeyboard. You can of course still get messages direct if you want, and in fact TLkeyboard stores the "raw" message, so you have it if you want it.

Forward to program 31

1.18 30asm

```
                Forward to program 31
                * 30.asm     Demonstrate TLKeyboard     Version 0.01   8.6.99

 include 'Front.i'

; This program demonstrates the use of TLKeyboard which which returns an
; ASCII value in D0, or dummy values for non-printable keys.


strings: dc.b 0
 dc.b 'Press or click anything - Close window when finished',0 ;1
 dc.b 'D0 D1 D2 D3',0 ;2
st_3: dc.b '.. .. .. .. ',0 ;3
st_4: dc.b 'TLKeyboard demo',0 ;4
 dc.b 'Error: out of memory',0 ;5
 ds.w 0

* open screen & window; show Keyboard until Close Window clicked
Program:
 TLwindow #1,#0,#0,#200,#100,#400,#150,#0,#st_4 ;open window 1
 beq Pr_bad              ;bad if out of chip ram
 TLstrbuf #2             ;string 2 to buffer
 TLtext #10,#5           ;print string 2 at 10,5

Pr_wait:
 TLkeyboard              ;get from keyboard - see TLKeyboard in tandem.guide
 cmp.b #$93,d0           ;($93 is my dummy code for IDCMP_CLOSEWINDOW)
 beq.s Pr_close          ;quit if Close Window clicked
 lea st_3,a0             ;poke the inputs (D0-D3) into string 3
 bsr Hex
 move.l d1,d0
 bsr Hex
 move.l d2,d0
 bsr Hex
 move.l d3,d0
 bsr Hex
 TLstrbuf #3             ;string 3 to buffer
 TLtext #10,#24          ;print string 3 at 10,24
 bra Pr_wait             ;wait for next input
Pr_close:
 rts                     ;return ok
Pr_bad:
 TLbad #5
 rts


* put ASCII (2 hex digits) for hex of D0.B in (A0)+
Hex:
 move.l d0,-(a7)
 lsr.l #4,d0
 bsr.s Hx_n
 move.l (a7)+,d0
 bsr.s Hx_n
 move.b #$20,(a0)+
 rts
```

```
Hx_n:
 and.b #15,d0
 add.b #'0',d0
 cmp.b #':',d0
 bcs.s Hx_p
 add.b #'A'-':',d0
Hx_p:
 move.b d0,(a0)+
 rts
```

Forward to program 31

1.19 p31 Demonstrate Program 31

Program 31

Run Program 31 -> TLerror

Source program 31
-> Teaching/31.asm

This program demonstrates TLError, which gives an error report for any
TL program which goes wrong. TLError also calls dos.library's _LVOFault,
and if that is non-zero, outputs the error to CLI. TLError puts out
tandem.library's error (if any) to CLI, & leaves it in buff where you
can put up a requester to report it if you want.

Forward to program 32

1.20 31asm

Forward to program 32

```
* 31.asm     Demonstrate TLerror       Version 0.00   1.9.97

 include 'Front.i'

; Most tandem.library routines can return with an error condition. Their
; MACRO's will generally return EQ if in error. Your program can find the
; specifics of the error in xxp_errn(a4). Programs which can return an
; error leave xxp_errn(a4)=0 if ok, else they return an error number. If
; you call TLerror while xxp_errn(a4)<>0, then it will put an error report
; in (a4) (where you can put it up on a requester for acknowledgement), and
; TLerror also sends the error report to the monitor. If there is a DOS
; error, TLerror also sends that to the monitor.

; The program below does everything on the monitor (CLI), rather than using
; intuition windows.
```

```
strings: dc.b 0
 dc.b $0C,'Demonstrate TLerror',0 ;1
 dc.b 'First, I will try to "Makedir PRT:Fred", which is illegal',0 ;2
 dc.b 'After that failed, I called TLError, which puts this in buffer:',0 ;3
 dc.b '(Press <return> to acknowledge)',0 ;4
st_5: dc.b 'Makedir >NIL: PRT:Fred',0 ;5     <- an illegal DOS command
st_6: dc.b 'List >NIL: RAM:',0 ;6     <- a legal DOS command
 dc.b 'Now, I''ll send a (hopefully) legal command:  "List >NIL: RAM:"',0 ;7
 dc.b 'Here is what TLerror reports (i.e. nothing since legal)',0 ;8
 dc.b 0 ;9

 ds.w 0


* sample program
Program:
 TLoutstr #1               ;greetings
 TLoutstr #9

 TLoutstr #2               ;send string 2
 TLoutstr #4
 TLinput

 TLoutstr #3               ;string 3 first (else TLerror reports on this)

 move.l xxp_dosb(a4),a6    ;set up to execute st_5 (=string 5) as a command
 move.l #st_5,d1
 moveq #0,d2
 moveq #0,d3
 jsr _LVOExecute(a6)       ;send an illegal command, to cause an error

 TLerror                   ;get error report to buff

 TLoutstr #9
 TLoutstr #4
 TLinput

 TLoutstr #9
 TLoutstr #7               ;now send strings 7 and 8
 TLoutstr #8
 move.l #st_6,d1           ;now send a legal command (at st_6, string 6)
 moveq #0,d2               ;(if you re-run this, it will be illegal, since
 moveq #0,d3               ; RAM:Fred will already exist)
 jsr _LVOExecute(a6)

 TLerror                   ;get error report to buff

 TLoutstr #9
 TLoutstr #4               ;send to op window
 TLinput                   ;wait for acknowledge
 rts


                Forward to program 32
```

1.21 p32 Demonstrate Program 32

```
                Program 32

This program introduces the selection of fonts.

   Run Program 32    -> TLgetfont and TLnewfont

                Source program 32
                 -> Teaching/32.asm

The program demonstrates TLgetfont & TLnewfont. Front.i has a set of 10
fonts (which can be in various styles &c.), and TLgetfont puts a
predetermined font into one of the 10 fonts. TLnewfont attaches one of those
fonts to a window.


                Forward to program 33
```

1.22 32asm

```
                Forward to program 33
                * 32.asm    TLgetfont and TLnewfont     version 0.01   8.6.99

 include 'Front.i'

; This program introduces the selection of fonts. There are 5 subrountines
; for this. they use a table of fonts which is pointed to by xxp_FSuite.
; There can be 10 fonts, numbered 0-9, and each of these can have 4 forms,
; theoretically, 40 fonts in all. The window(s) can each have a font &
; font style attached. Or, they can have Topaz/8 attached, i.e. font 0,
; as they do initially.

; Font 0 is always Topaz/8. So, you can use fonts 1-9. If you use TLgetfont
; or TLaslfont on any font already opened, it will be closed, and any
; windows using it will revert to font 0, i.e. Topaz/8.

; Here are the subroutines:
;
; TLGetfont:  puts a predetermined fontname and height in an FSuite entry,
;             and closes anything already there. Does not open the font.
; TLAslfont:  as TLGetfont, but puts up a requester for the font.
; TLNewfont:  attaches an xxp_FSuite font & style to the xxp_ window.
; TLFsub:     closes an Fsuite font (called automatically by TLWclose)

; The commonest font styles are:
;   0 plain
;   1 bold
```

```
;   2 italic
;   3 bold+italic

strings: dc.b 0
st_1: dc.b 'Demonstrate TLGetfont and TLNewfont',0 ;1
st_2: dc.b 'Times.font',0 ;2
 dc.b 'This is in Font 0..',0 ;3
 dc.b 'But this is in Font 1!!',0 ;4
 dc.b 'Oh - back to Font 0. Life goes on.',0 ;5
 dc.b 'Perhaps it shouldn''t. So, click my close gadget.',0 ;6

 ds.w 0


* Demonstrate fonts
Program:
 TLwindow0
 beq.s Pr_quit
 bsr Test                  ;do test of Getfont,&c
Pr_quit:
 rts

* test Getfont, &c
Test:
 TLgetfont #st_2,#1,#24    ;name=Times.font, FSuite number=1, height=24
 TLstring #3,#10,#15       ;print string 3 at (10,15) in font 0 (Topaz/8)
 TLnewfont #1,#0,#0        ;attach font 1 (Times/24), style 0 (plain)
 beq Te_quit        ;go if bad (can only happen on 1st call of a font)

 TLstring #4,#10,#25       ;print string 4 at (10,25) in times/24 plain
 TLnewfont #0,#0,#0        ;back to font 0 for window
 TLstring #5,#10,#50       ;print string 5 at (10,50) in prefs font
 move.l xxp_AcWind(a4),a5  ;point a5 to active window (i.e. window 0)
 move.b #2,xxp_FrontPen(a5) ;use colour 2 for variety (default colour 1)
 move.w #4,xxp_Tspc(a5)    ;and spread out text spacing (default spacing 0)
 TLstring #6,#10,#60       ;print string 6 at (10,60) in prefs font

Te_wait:
 TLkeyboard                ;wait for response
 cmp.b #$93,d0             ;re-try if not close window
 bne Te_wait

Te_quit:
 rts                       ;note that Front0.i closes all fonts & windows


                Forward to program 33
```

1.23 p33 Demonstrate Program 33

```
                Program 33

   Run Program 33    -> TLtsize, TLreqcls
```

```
                Source program 33
                 -> Teaching/33.asm

This program demonstrates TLtsize, which allows you to calculate the
pixel dimensions of a given string in the prevailing font. This info was
used in the program to calulate where in the window to place it.

The program also calls TLreqcls, which clears the window before re-drawing
it, if the window is resized.


                Forward to Program 34
```

1.24 33asm

```
                Forward to program 34
                * 33.asm    Demonstrate TLtsize        version 0.01  8.6.99

 include 'Front.i'

; This program introduces TLTsize, which gets text size without actually
; printing it. This is useful for making your programs font sensitive.
; The program below allows for the window to be resized. This will be
; covered in a more rigorous fashion in later examples. (TLtsize
; corrects some bugs in the Amiga graphics.library text sizing routines).

strings: dc.b 0
st_1: dc.b 'Test TLTsize',0 ;1
 dc.b 'This text is to appear at the bottom right',0 ;2
 dc.b 'This text is to be spread out',0 ;3
 dc.b '(Click the close window gadget)',0 ;4

 ds.w 0


* sample program
Program:
 TLwindow #0,#0,#0,#400,#150,#640,#200,#0,#st_1
 beq.s Pr_quit
 bsr Test                  ;do test of TLtsize,&c
Pr_quit:
 rts

* test TLTsize
Test:
 move.l xxp_AcWind(a4),a5  ;a5 = the currently popped window
 move.w #$0100,xxp_FrontPen(a5) ;pens 1,0

 TLreqcls                  ;clear window, call TLwupdate

 TLstrbuf #2               ;string 2 to buff
 TLtsize                   ;get string size
```

```
 moveq #0,d2                ;calculate rightmost posn printable
 move.w xxp_PWidth(a5),d2   ;(use D2 since TLwcheck changes D0)
 sub.l d4,d2                ;D2=rightmost printable
 moveq #0,d1                ;calculate botmost posn printable
 move.w xxp_PHeight(a5),d1
 sub.w d6,d1                ;D1=botmost posn printable

 TLtrim d2,d1               ;print the text at the bot right
 move.w #8,xxp_Tspc(a5)     ;set inter-chr spc to 8 (normally 0)
 TLstring #3,#10,#19        ;print string 3 at 10,19 (spread out)
 clr.w xxp_Tspc(a5)         ;inter-chr spc back to 0
 move.b #2,xxp_FrontPen(a5) ;colour 2 to highlight string 4
 TLstring #4,#4,#29         ;print string 4 at 4,29 (not spread out)

Te_wait:
 TLwcheck                   ;go redraw if window resized
 bne Test
 TLkeyboard                 ;wait for close gadget
 cmp.w #$93,d0
 bne Te_wait
 rts
```

Forward to program 34

1.25 p34 Demonstrate Program 34

Program 34

Run Program 34 -> Refeshing, Choosing a font

Source program 34
-> Teaching/34.asm

This program demonstrates TLaslfont, which allows the user to choose a font.

It also introduces a simple method of window refreshing using TLwupdate and TLwcheck. You can of course use intuition.library's method of refreshing if you want to. (All tandem.library's windows are smart refresh).

Forward to Program 35

1.26 34asm

```
Forward to program 35
* 34.asm     Demonstrate TLaslfont, TLwupdate      version 0.01  ↩
      8.6.99
```

```
 include 'Front.i'

; I here introduce the TLaslfont subroutine. It is the same as TLgetfont,
; but puts up a requester for you to choose the font and size. Note how
; easy tandem.library makes this.

; The TLbad MACRO shows a convenient way to send an error report to the
; user if things go wrong. First, it sends a message to the monitor with
; TLoutput. Then, it sets xxp_ackn<>0; this tells Front0.i to ask the user
; to press <return> to acknowledge before closing down in a CLI error
; condition.

; This program refreshes the window, by using redrawing whenever TLKeyboard
; returns a window resized IDCMP. This method works adequately for smart
; refresh windows. You need to put the drawing of the window into a
; subroutine, and use TLTrim (called by TLstring), not TLText. TLText is
; faster, but TLTrim checks that the text fits, so use it if there is any
; doubt.

strings: dc.b 0
st_1: dc.b 'Demonstrate TLAslfont',0 ;1
 dc.b 'Here is some normal font',0 ;2
 dc.b 'This is in your selected font!',0 ;3
 dc.b '(Close window gadget to quit; other to recycle)',0 ;4
 dc.b 'Error: Text too large to fit in window',0 ;5
 dc.b 'Error: out of chip RAM:',0 ;6
 dc.b 'Use zoom gadget to demonstrate refreshing.',0 ;7

 ds.w 0


* program to test Aslfont
Program:
 TLwindow #0,#0,#0,#200,#50,#640,#200,#0,#st_1 ;open window 0
 beq.s Pr_quit             ;go if can't
 bsr Test                  ;do test of Aslfont
 rts            ;quit ok

Pr_quit:
 TLbad #6                  ;report error
 rts


* test TLAslfont
Test:
 TLreqcls                  ;clear window
 TLaslfont #1              ;select font 1
 bne.s Te_chosen           ;go if ok
 tst.l xxp_errn(a4)        ;EQ if cancel, else can't open requester
 beq Te_quit               ;quit without error if cancel selected

 TLbad #6                  ;else error condition (asl out of mem)
 rts

Te_chosen:
 bsr Refresh               ;print on window
 TLkeyboard                ;get response
```

```
 cmp.b #$96,d0
 beq Te_chosen              ;redraw if size window
 cmp.b #$93,d0              ;reccyle until close window (or cancel asl)
 bne Test

Te_quit:
 rts


* print everything on window (self contained, so can use as refresh)
Refresh:
 TLwupdate                  ;update window dims
 TLnewfont #0,#0,#0         ;attach topaz/8 plain to main window
 TLstring #2,#0,#0          ;print normal font
 TLnewfont #1,#0,#0         ;attach selected font - style plain, main window
 TLstring #3,#0,#12         ;print message in selected font
 TLtsize                    ;get message size
 add.l #12,d6               ;go to bot of text (text height + 12)
 TLnewfont #0,#0,#0         ;re=attach topaz/8 plain
 TLstring #4,#0,d6          ;message below string 3
 add.w #10,d6
 TLstring #7,#0,d6          ;further message below string 4
 rts


                Forward to program 35
```

1.27 p35 Demonstrate Program 35

```
                Program 35

This program introduces the first of tandem.library's requesters, the
"choose" requester.

   Run Program 35    -> The "choose" requester

                Source program 35
                 -> Teaching/35.asm

tandem.library's requesters are font sensitive, and you may modify them
quite simply also. The user also can set preferences over the pens used
and other appearance of the requesters.


Click this button, or the button at the end, when you've read the text below


                Forward to Program 36
                The calling of choose requesters is very simple. All you need is a ↩
                    set of
strings, and you call it with 2 parameters. If you have the title and 4
choices in 5 strings from string 13, then you'd merely put:

      TLreqchoose #13,#4
```

```
And it will return D0 = 1 to 4.    (or 0 if out of chip memory / won't fit)

It's that simple. If the font or other aspects of the layout are too large
to fit on the screen, tandem.library uses smaller fonts &c until such times
as it can fit the requester on the screen.

You can easily attach help to tandem.library requesters, and they all have
built-in default help also.


                Forward to Program 36
```

1.28 35asm

```
                Forward to program 36
                * 35.asm   TLReqchoose        version 0.01   8.6.99

 include 'Front.i'

; TLReqchoose puts up a requester to allow the user to select from
; several choices. The idea is to put a series of strings: a header,
; and at least 1 choice, in strings, and set D0 to the no. of the 1st
; string, and then let D1 be the number of choices. The requester is
; adjusted to accommodate the operative font size.

; Alternately in TLReqchoose you can set D1 to 0, when D0 is ignored, and
; the requester simply asks the user to acknowledge whatever is in buffer.
; Both types are demonstrated below.


strings: dc.b 0
st_1: dc.b 'Demonstrate TLreqchoose',0 ;1
 dc.b 'Choose from these buttons:',0 ;2
 dc.b 'This is a button!',0 ;3
 dc.b 'And this is another!',0 ;4
 dc.b 'And yet a third!',0 ;5
 dc.b '(A last choice)',0 ;6
st_7: dc.b 'Times.font',0 ;7
 dc.b 'You chose '
st_8: dc.b '.',0 ;8
 dc.b 'Error - requester won''t fit or out of memory',0 ;9
 dc.b 'Error: Can''t load Times/24 font',0 ;10
 dc.b 'Here is some help:',0 ;11
 dc.b 'Click one of the boxes to choose',0 ;12
 dc.b 'Or, press the Function key in the box',0 ;13

 ds.w 0

* program to test TLReqchoose
Program:
 TLwindow0                  ;open a window
 beq.s Pr_quit              ;go if can't
```

```
 bsr Test                  ;do test of Reqchoose
 TLwclose                  ;close window & screen

Pr_quit:
 rts


* test Reqchoose
Test:
 move.w #12,xxp_Help(a4)   ;set up help - 2 lines from string 12
 move.w #2,xxp_Help+2(a4)
 TLgetfont #st_7,#1,#24    ;put times/24 in font #1
 TLnewfont #1,#1,#1        ;font=1 1=Bold 1=req window
 beq Te_bad1               ;go if can't load Times/24
 TLreqchoose #2,#4         ;header=string 2, 4 choices
 beq Te_bad2               ;go if ok
 clr.l xxp_Help(a4)        ;clear help (now obsolete)
 add.b #'0',d0             ;poke choice (in ASCII) into string 8
 move.b d0,st_8
 moveq #8,d0               ;string 8 if ok
 bra.s Te_rep

Te_bad1:
 moveq #10,d0              ;string 10 if can't load font
 bra.s Te_rep

Te_bad2:
 moveq #9,d0               ;string 9 if font won't fit / out of mem

Te_rep:
 TLstrbuf d0               ;string (8-10) to buffer
 TLreqchoose #0,#0         ;put up requester to acknowledge buffer contents
 rts
```

Forward to p36

1.29 p36 Demonstrate Program 36

```
                Program 36
```

This program demonstrates TLreqinput, which allows you to type a string or number into the computer.

```
   Run Program 36    -> String / Number input

                Source program 36
                 -> Teaching/36.asm
```

Once again, it's extremely simple. You put the prompt into the buff, and then enter something like:

```
   TLreqinput #5,str,#10     ;header string 5, string, 10 characters max
```

```
And you get the string in buff, or if a number, in D0. (or EQ if cancel)

You can have a typing tablet narrower than the maximum string width to
scroll sideways if required. Numberic input can be decimal or hex.


                Forward to Program 37
```

1.30 36asm

```
                Forward to program 37
                * 36.asm      TLreqinput       version 0.01   8.6.99

 include 'Front.i'

; TLreqinput puts up a requester to allow the user to type in a number or
; string. The initial form of the string (the "prompt") is put in xxp_buff.
; TLreqinput adjusts the requester size to allow for the operative font.
; TLreqinput can scroll the input sideways if the input tablet is too
; narrow.

strings: dc.b 0
st_1: dc.b 'Test TLReqinput',0 ;1
 dc.b 'This is a Reqinput requester',0 ;2
 dc.b 'Edit this string',0 ;3
 dc.b 'You chose to cancel',0 ;4
 dc.b 'Error: the Requester was too big, or out of mem',0 ;5
st_6: dc.b 'Times.font',0 ;6
 dc.b 'Error: can''t load Times/24 font',0 ;7
 dc.b 'Here is some help...',0 ;8
 dc.b 0 ;9
 dc.b '1. Brush your teeth regularly.',0 ;10
 dc.b '2. Have a clean handkerchief.',0 ;11
 dc.b 'Type your input string.',0 ;12
 dc.b 'Then:',0 ;13
 dc.b 'Press <Return> or click "OK"',0 ;14
 dc.b 'Or, press <Esc> or click "Cancel"',0 ;15

 ds.w 0


* program to demonstrate TLreqinput
Program:
 TLwindow #0,#0,#0,#300,#120,#640,#256,#0,#st_1
 beq.s Pr_quit             ;go if can't
 bsr Test                  ;do test of TLReqwindow

Pr_quit:
 rts


* test TLReqinput
Test:
```

```
 move.w #12,xxp_Help(a4)   ;help 4 lines from 12
 move.w #4,xxp_Help+2(a4)
 TLgetfont #st_6,#1,#24    ;font #1 = times/24
 TLnewfont #1,#0,#1        ;use font #1, type plain, req windows
 beq Te_bad1               ;go if can't open times/24

 TLstrbuf #3               ;prompt to buffer
 TLreqinput #2,str,#30     ;header=string 2, type string,max 30 chrs
 bne.s Te_ok               ;echo the input if ok
 tst.l xxp_errn(a4)
 beq.s Te_canc
 bra.s Te_bad

Te_ok:
 move.w #8,xxp_Help(a4)    ;ok - update help 4 lines from 8
 move.w #4,xxp_Help+2(a4)
 bra.s Te_rep              ;go report string as edited

Te_bad1:
 moveq #7,d0               ;bad 1: str 7
 bra.s Te_bad

Te_bad2:
 moveq #5,d0               ;bad 2: str 5
 bra.s Te_bad

Te_canc:
 moveq #4,d0               ;canc: str 4

Te_bad:
 TLstrbuf d0               ;put report in buff

Te_rep:
 TLnewfont #0,#0,#1        ;Topaz/8 to req windows
 TLreqchoose               ;report cancel/too big/can't load/ok
 rts


               Forward to program 37
```

1.31 p37 Demonstrate Program 37

```
               Program 37

   Run Program 37    -> Get a filename / path

               Source program 37
                -> Teaching/37.asm
```

This program demonstrates TLaslfile, which allows the user to specify a filename for loading / saving. The demo program does not actually load or save anything, it just illustrates the process of getting a file path from the user.

```
                Forward to Program 38
```

1.32 37asm

```
                Forward to program 38
                * 37.asm     TLaslfile          version 0.01    8.6.99

 include 'Front.i'

; TLaslfile puts up a requester to allow the user to choose a filepath
; for loading or saving. The demonstration below does not actually go on
; to open the file; so it is just for demonstration purposes. The filepath
; will be relative to the CD when TLAslfile runs. To use the MACRO
; TLaslfile, you:
;
;  point \1 to a DS.B to hold the directory part of the filepath
;  point \2 to a DS.B to hold the file part of the filepath
;  set \3 to the string number of the hail
;  set \4 to ld or sv   (if you omit \4, ld is assumed)
;
; the \1 and \2 should be of length 128 and 32 to be safe. You can make
; them null strings, or give them initial values to act as prompts.
; The Asl requester updates the \1 and \2 buffer to contain the path you
; choose, and TLasfile puts the total path (relative to the CD) in xxp_buff,
; ready for you to call TLOpenread or TLOpenwrite.


strings: dc.b 0
st_1: dc.b 'Test TLAslfile',0 ;1
 dc.b 'This is an TLAslfile requester',0 ;2
 dc.b 'You chose cancel',0 ;3
 dc.b 'Error: Can''t open  window - out of chip memory',0 ;4

 ds.w 0


dir: ds.b 128
fil: ds.b 32


* program to demonstrate TLaslfile
Program:
 TLwindow0                ;open window
 beq.s Pr_quit            ;go if can't
 bsr Test                 ;do test of Aslfile
 TLwclose                 ;close window & screen
 rts                      ;return ok

Pr_quit:
 TLbad #4                 ;return bad if out of chip memory
 rts
```

```
* test Reqinput
Test:
 clr.b dir                  ;directory here (null prompt)
 clr.b fil                  ;file here (null prompt)
 TLaslfile #dir,#fil,#2,ld
 bne.s Te_good
 TLstrbuf #3                ;if cancel, report in buffer, else send path

Te_good:
 TLreqchoose                ;report choice
 rts
```

Forward to program 38

1.33 p38 Demonstrate Program 38

Program 38

This program demonstrates menus.

Run Program 38 -> Menus

Source program 38
-> Teaching/38.asm

In your program you put up a series of TLnm MACRO's, 1 for each menu item, to specify the menu. Then, you call TLreqmenu to set the menu up. You then call TLreqmuset to attach it to any window, or TLreqmuclr to detach it. Individual menu items are disabled, or enabled, with TLmenuoff & TLmenuon.

Examine the source for this program to see how simple it is. Note especially how easy it is to attach hotkeys to menu items using TLnm.

Forward to Program 39

1.34 38asm

Forward to program 39

```
* 38.asm     TLreqmenu       version 0.01    8.6.99


 include 'Front.i'          ;*** change to 'Tandem.i' to step thru TL's ***


; This program is designed to demonstrate the use of TLreqmenu.
; To use TLreqmenu, you must first create a NewMenu structure, and then
```

```
; call TLreqmenu. This creates a value in xxp_Menu which can then be
; used by TLreqmuset to attach the menu strip to a window, or TLreqmuclr
; to detach it. While the menu strip is attached, if the user makes a menu
; selection, it will be returned by the next TLkeyboard, with the value
; $95 in D0; the menu num, item num and sub-item num will be in D1-D3.
; If any of these are null, they will be -1 (n.b. separator bars count
; as items, but cannot be selected).

; When you set up the NewMenu, you can if you like put string numbers in
; Label fields, which TLreqmenu will convert to pointers. You may also use
; one only string with some/all of the hotkey characters in it in order.
; Then, put that string number in the CommKey field, and Reqmenu will
; convert it to a pointer. (Although TLnm's change their contents when run,
; they can safely be rerun, or be part of a PURE program).

; the newmenu memory area below (an instance of a NewMenu structure) uses
; the TLnm MACRO.  TLnm requires \1 to be 1,2,3 or 4 for NM_TITLE,
; NM_ITEM, NM_SUB or NM_END. \2 is the string number of the label.
; \3 (if present) is the string number of the CommKeys string. If \2
; is -1, it is an NM_BARLABEL. (You can also use pointers instead of
; string numbers for \2 and \3 if you want). Refer also to
; libraries/gadtools.i for details of Amiga's NewMenu structure. \4 and
; \5 (if present) are explained in tandem.i's AUTODOC for TLnm.


* The NewMenu structure (the \2 and \3 values refer to string numbers)
newmenu:
 TLnm 1,3       ;Menu 0
 TLnm 2,4,13    ;  Item 0         A
 TLnm 2,5       ;  Item 0
 TLnm 3,6,13    ;    Sub-item 0   B
 TLnm 3,7       ;    Sub-item 1
 TLnm 2,8       ;  Item 2
 TLnm 2,-1      ;  (bar)
 TLnm 2,9       ;  Item 3
 TLnm 1,10      ;Menu 1
 TLnm 2,11,13   ;  Item 0         C
 TLnm 2,12      ;  Item 1
 TLnm 4,0       ;delimiter


strings: dc.b 0
st_1: dc.b 'Test TLReqmenu',0 ;1
 dc.b 'You chose menu '
st_2a: dc.b ' , item '
st_2b: dc.b ' , sub-item '
st_2c: dc.b ' .',0             ;2
 dc.b 'Menu 0',0               ;3
 dc.b 'Item 0',0               ;4 CommKey A
 dc.b 'Item 1',0               ;5
 dc.b 'Sub-item 0',0           ;6 CommKey B
 dc.b 'Sub-item 1',0           ;7
 dc.b 'Item 2',0               ;8
 dc.b 'Item 4 (bar=3)',0       ;9
 dc.b 'Menu 1',0               ;10
 dc.b 'Item 0',0               ;11 CommKey C
 dc.b 'Item 1',0               ;12
```

```
 dc.b 'ABC',0                ;13 (the CommKeys)
 dc.b '(Choose a menu item, or click close gadget)',0          ;14
 dc.b 'Error: can''t open screen/window. Out of chip memory',0  ;15
 dc.b 'Error: the gadtools.library could not set up the menu',0 ;16

 ds.w 0


* program to demonstrate  TLreqmenu
Program:
 TLwindow #0,#0,#0,#300,#120,#640,#256,#0,#st_1
 beq.s Pr_quit             ;go if can't
 bsr Test                  ;do test of Reqmenu
 rts

Pr_quit:
 TLbad #15
 rts


* test Reqmenu
Test:
 TLreqmenu #newmenu        ;set up xxp_Menu
 beq Te_bad                ;bad if can't
 TLreqmuset                ;attach menu to window
 TLstring #14,#20,#19      ;ask user to select an item

Te_wait:
 TLkeyboard                ;get response
 cmp.w #$93,d0
 beq.s Te_quit             ;done if close gadget
 cmp.w #$95,d0
 bne Te_wait               ;else, keep waiting until menu item selected
 add.b #'0',d1
 move.b d1,st_2a           ;report menu number ('0' if null)
 add.b #'0',d2
 move.b d2,st_2b           ;report item number ('0' if null)
 add.b #'0',d3
 move.b d3,st_2c           ;report sub-item no ('0' if null)
 TLstring #2,#20,#40       ;report choice, and continue
 bra Te_wait

Te_quit:
 rts

Te_bad:
 TLbad #16                 ;error condition if can't create menu
 rts
```

Forward to program 39

1.35 p39 Demonstrate Program 39

```
                Program 39

This program demonstrates the TLreqinfo requester. TLreqinfo requesters
can have just an OK button, or OK and Cancel buttons, or else a set of
1 or more custom buttons.

   Run Program 39    -> TLreqinfo requesters I

                Source program 39
                 -> Teaching/39.asm

The MACRO for this is just as easy as all the other tandem.library
requesters. To reiterate, you get everything layed out automatically,
font sensitive, with layout & pens following user prefs if appropriate,
and tandem.library even reduces the font size &c to make things fit if
necessary. All in just 1 line of source code written by you.


                Forward to Program 40
```

1.36 39asm

```
                Forward to program 40
                * 39.asm          TLreqinfo     version 0.01    8.6.99


 include 'Front.i'         ;*** change to 'Tandem.i' to step thru TL's ***


; This program tests the next of the requesters created by tandem.library,
; the TLreqinfo requester. This creates an information box, with an
; OK box, and an optional Cancel box, or a designer set of boxes. To
; use the TLreqinfo MACRO:
;
;  \1 = stringnum of header, followed by other strings for display
;  \2 = no. of strings to be displayed (1+)
;  \3 = 1:ok button 2:ok/canc buttons 3:designer boxes
;
; TLreqinfo returns choice in D0 (1+), or D0=0 if bad
;
; If \3=3, then \2 must be 2+. The last string is 1 or more strings for
; button contents, separated by \ characters.


strings: dc.b 0
st_1: dc.b 'Demonstrate TLreqinfo',0 ;1
 dc.b 'This is the first line of info,',0 ;2
 dc.b 'whereas this is the second line.',0 ;3
 dc.b 'Finally, the third line.',0 ;4
 dc.b 'You selected OK',0 ;5
 dc.b 'You selected Cancel',0 ;6
 dc.b 'Error: The requester won''t fit!!',0 ;7
```

```
 dc.b 'Error: Can''t open screen & window - out of chip memory',0 ;8
 dc.b 'Error: Can''t open the ASL font requester',0 ;9
 dc.b 'Error: Can''t load your selected font',0 ;10

 ds.w 0

* deomonstrate TLreqinfo
Program:                                ;Note carefully how this program
 TLwindow0                              ;gives error reports to the
 bne.s Pr_open                          ;user if something goes wrong in
 TLbad #8       ;quit if can't          ;the warm-up. This is an important
 rts                                    ;aspect of user friendliness.

Pr_open:
 TLaslfont #1               ;select font 1
 bne.s Pr_ofont             ;go if ok
 tst.l xxp_errn(a4)
 beq Pr_redi                ;if font selection cancelled, use default font
 TLbad #9                   ;if Asl font request failed, return bad
 bra.s Pr_quit

Pr_ofont:
 TLnewfont #1,#0,#1         ;put font 1, plain in req font
 bne.s Pr_redi
 TLbad #10                  ;bad if can't
 bra.s Pr_quit

Pr_redi:
 bsr Test                   ;Test TLReqinfo

Pr_quit:
 TLwclose                   ;close screen, window & return
 rts

* test TLreqinfo
Test:
 clr.w xxp_ReqNull(a4)      ;first get requester dimensions
 TLreqinfo #2,#3,#2
 bne.s Te_ok
 TLbad #7                   ;bad if requester won't fit
 rts

Te_ok:
 move.l xxp_AcWind(a4),a5  ;a5 points to WSuite for window
 move.w xxp_PWidth(a5),d0  ;center the requester horizontally in window
 sub.w xxp_reqw+2(a4),d0
 bmi.s Te_vert              ;go if too wide
 lsr.w #1,d0
 move.w d0,xxp_ReqLeft(a5)

Te_vert:
 move.w xxp_PHeight(a5),d0 ;center the requester viertically in window
 sub.w xxp_reqh+2(a4),d0
 bmi.s Te_redi              ;go if too high
 lsr.w #1,d0
 move.w d0,xxp_ReqTop(a5)
```

```
Te_redi:
 TLreqinfo #2,#3,#2        ;now, put up requester for real
 addq.w #4,d0              ;choice becomes 5/6
 TLreqinfo d0              ;report choice (default /2,/3=#1,#1)
 rts


                Forward to program 40
```

1.37 p40 Demonstrate Program 40

```
                Program 40

This is like 39, but with a custom set of buttons in the TLreqinfo
requester.

   Run Program 40    -> TLreqinfo II

                Source program 40
                 -> Teaching/40.asm



                Forward to Program 41
```

1.38 40asm

```
                Forward to program 41
                * 40.asm    more TLreqinfo          version 0.01    8.6.99


 include 'Front.i'         ;*** change to 'Tandem.i' to step thru TL's ***


; This is like Teaching/39.asm, but it uses a set of custom buttons.

; If \3=3, then \2 must be 2+. The last string is 1 or more strings for
; button contents, separated by \ characters.


strings: dc.b 0
st_1: dc.b 'Test Reqinfo',0 ;1
 dc.b 'This is the first line of info,',0 ;2
 dc.b 'whereas this is the second line.',0 ;3
 dc.b '.... Finally, the *third* line!!',0 ;4
 dc.b 'Press <Help> for even more info!',0 ;5
 dc.b '1st\2nd\3rd',0 ;6
 dc.b 'Error: The requester won''t fit!!',0 ;7
 dc.b 'Error: Can''t open screen & window - out of chip memory',0 ;8
```

```
 dc.b 'Error: Can''t open the ASL font requester',0 ;9
 dc.b 'Error: Can''t load your selected font',0 ;10
 dc.b 'You selected 1st',0 ;11
 dc.b 'You selected 2nd',0 ;12
 dc.b 'You selected 3rd',0 ;13
 dc.b 'Here is some help!!',0 ;14
 dc.b 'The 3 lines of info are presented for',0 ;15
 dc.b 'your delectation. When you''ve done',0 ;16
 dc.b 'enjoying it, click one of the buttons',0 ;17
 dc.b 'at the bottom.',0 ;18
 dc.b 'Eat healthy meals, avoid illegal drugs.',0 ;19

 ds.w 0


* test Reqinfo
Program:                                    ;Note carefully how this program
 TLwindow #0,#0,#0,#200,#100,#640,#300,#0,#st_1 ;gives error reports.
 bne.s Pr_open                              ;If something goes wrong in the
 TLbad #8       ;quit if can't              ;warm-up. This is an important
 rts                                        ;aspect of user friendliness. Also
Pr_open:                                    ;Help is always available
 TLaslfont #1   ;select font 1
 bne.s Pr_ofont ;go if ok
 tst.l xxp_errn(a4)
 beq Pr_redi    ;if font selection cancelled, use default font
 TLbad #9       ;if Asl font request failed, return bad
 bra Pr_quit

Pr_ofont:
 TLnewfont #1,#2,#1 ;put font 1 bold in req
 TLnewfont #1,#0,#2 ;put font 1 plain in help
 bne.s Pr_redi
 TLbad #10      ;bad if can't
 bra.s Pr_quit

Pr_redi:
 bsr Test       ;Test TLReqinfo

Pr_quit:
 rts


* test TLReqinfo
Test:
 move.w #14,xxp_Help(a4)  ;help from line 14
 move.w #5,xxp_Help+2(a4) ;5 lines thereof
 clr.w xxp_ReqNull(a4)    ;first get requester dimensions
 TLreqinfo #2,#4,#3
 bne.s Te_ok
 TLbad #7                 ;bad if requester won't fit
 rts

Te_ok:
 move.l xxp_AcWind(a4),a5 ;a5=WSuite of window
 move.l xxp_Width(a4),d0  ;center the requester on screen
 sub.l xxp_reqw(a4),d0
```

```
lsr.w #1,d0
move.w d0,xxp_ReqLeft(a5)
move.l xxp_Height(a4),d0
sub.l xxp_reqh(a4),d0
lsr.w #1,d0
move.w d0,xxp_ReqTop(a5)
TLreqinfo #2,#5,#3        ;now, put up requester for real
add.w #10,d0              ;choice becomes 11/12/13
move.w #19,xxp_Help(a4)   ;update help
move.w #1,xxp_Help+2(a4)
TLreqinfo d0              ;report choice (default /2,/3=#1,#1)
rts
```

Forward to program 41

1.39 p41 Demonstrate Program 41

Program 41

The TLreqshow requester:

Run Program 41 -> TLreqshow - display dynamic information

Source program 41
-> Teaching/41.asm

The TLreqshow requester allows the display of dynamically calculated strings. There are three TLreqshow demos in this Guide, to show different aspects of this very powerful feature.

Unlike the other requesters, it takes some programming skill to use the very extensive features of TLreqshow to the full. You can use it to show large amounts of info in a hierarchical structure, with searches, &c.

You can use the TLreqshow to ask the user to choose one of the dynamically calculated strings also.

Forward to Program 42

1.40 41asm

Forward to program 42
* 41.asm TLreqshow version 0.01 8.6.99

```
include 'Front.i'         ;*** change to 'Tandem.i' to step thru TL's ***
```

```
; This program demonstrates the use of TLreqshow, which allows the user to
; view a series of lines, while your program calculates their contents
; dynamically. It takes some programming skill to get the most from
; TLreqshow.


strings: dc.b 0
 dc.b 'Some strings for your delectation (n.b. press <Help> for info)',0 ;1
 dc.b 'String  ....',0 ;2
 dc.b 'You have seen a TLReqshow requester (sigh!)',0 ;3
 dc.b '(press <Help> for more help!)',0 ;4
 dc.b 'Alas! This wonderful TLReqshow demo is finished!',0 ;5
 dc.b '(Wasn''t it great!!! Applause! Applause!)',0 ;6

 ds.w 0


clikd: ds.l 1 ;line selected


* test program
Program:
 TLwindow #-1
 move.l #-1,clikd                ;flag no line is yet clicked
 TLreqshow #Hook,#1,#100,#17,#40,seek
 beq.s Pr_quit                   ;go if TLreqshow fails
 move.w #5,xxp_Help(a4)          ;help from line 5, 2 lines
 move.w #2,xxp_Help+2(a4)
 TLreqinfo #3,#2                 ;final message

Pr_quit:
 rts


* Act as hook for TLReqshow
Hook:
 tst.l d0             ;go if line clicked
 bmi.s Ho_clkd
 bsr Make             ;synthesize line d0, point a0 to it
 rts

Ho_clkd:
 cmp.l clikd,d0       ;line already highlighted?
 bne.s Ho_on          ;no, go
 move.l #-1,clikd
 moveq #1,d0          ;highlighting off
 rts

Ho_on:
 move.l d0,clikd      ;remember which line is being highlighted
 moveq #2,d0          ;turn highlighting on
 rts


* synthesize line d0
Make:
```

```
 TLstrbuf #2          ;string 2 to buffer
 move.l a4,a0
 addq.l #8,a0
 move.l #'    ',(a0)  ;blank num
 TLhexasc d0,a0       ;put num
 clr.b (a0)
 move.l a4,a0         ;point a0 to string as synthesized
 rts


                Forward to program 42
```

1.41 p42 Demonstrate Program 42

```
                Program 42

This demonstrates the simple TLassdev routine, which determines whether an
assign exists, and if so what it is asigned to, without putting up a
system requester (which can confuse the user) if the assign doesn't exist.

   Run Program 42    -> See if an assign exists

                Source program 42
                 -> Teaching/42.asm



                Forward to Program 43
```

1.42 42asm

```
                Forward to program 43
                * 42.asm     Demonstrate TLassdev       version 0.00      1.9.97


 INCLUDE 'Front.i'

; This tests if an assign exists, without putting up an annoying
; system requester if it doesn't. If it does exist, it tells you
; what it is (primarily) assigned to.


strings: dc.b 0
 dc.b 'LIBS',0 ;1   (note - do NOT put trailing :)
 dc.b 'EH?? LIBS: Doesn''t exist?? Impossible!!',0 ;2
 dc.b $0C,'LIBS: is assigned to:',0 ;3
 dc.b $0A,'(Press <return> to acknowledge)',0 ;4

 ds.w 0
```

```
* demonstrate TLAssdev
Program:
 TLoutstr #3     ;string 3 to CLI
 TLstrbuf #1     ;'LIBS' to buffer
 TLassdev        ;get its assign
 bne.s Pr_report ;go if ok
 TLstrbuf #2     ;else, report string 2 (can't happen?)

Pr_report:
 TLoutput        ;put assign
 TLoutstr #4     ;send string 4
 TLinput         ;get acknowledge
 rts
```

Forward to program 43

1.43 p43 Demonstrate Program 43

Program 43

This program demonstrates how to have several windows operating at once. You can resize & zoom them, and whenever you acticate a fresh window the "iteration" increases. The windows are all sensitive to resizing &c. you can close any of them & the others carry on, until such time as you've closed them all.

Run Program 43 -> Multiple windows

Sourcce program 43
-> Teaching/43.asm

This program demonstrates TLwpoll, which waits to see which window you activate, and also TLwsub, which closes windows.

Forward to Program 44

1.44 43asm

```
Forward to program 44
* 43.asm   TLwpoll, TLwsub       0.01     8.6.99


 include 'Front.i'         ;*** change to 'Tandem.i' to step thru TL's ***


; This program shows how to have several windows at once. In practice, each
```

```
; window would have its own subroutine to do whatever that window does.
; A window will keep calling TLKeyboard. If it returns resize, then re-draw
; the window. It it returns close, then close the window. If it returns
; inactive window, exit from the window, and call TLWpoll to wait for a
; window to become active. All other inputs would be dealt with within
; the context of whatever that window is for. Each window can have its own
; help, can call requesters, or be used for TLmultiline. Each can make use
; of the fonts in FSuite. There can be up to 10 windows, numbered 0-9.

; You should study this program carefully - Amiga programming relies
; heavily on mastering the difficult art of managing a suit of windows.


open: ds.w 1                ;number of open windows
spen: dc.l -1               ;pens for screen


strings: dc.b 0
 dc.b 'Iteration '
st_1a: dc.b 'A: Window '
st_1b: dc.b '2 is active',0 ;1
 dc.b 'Error: out of memory',0 ;2
 dc.b 'Error: can''t open windows - out of mem',0 ;3
 dc.b 'Error: can''t open window 2 - out of mem',0 ;4
 dc.b 'move windows about, click them, size them, & close them',0 ;5
st_6: dc.b 'A Private Screen',0 ;6
st_7: dc.b 'Window 0',0 ;7
st_8: dc.b 'Window 1',0 ;8
st_9: dc.b 'Window 2',0 ;9

 ds.w 0


Program:
 move.b #'Z',st_1a          ;initialise string 1
 TLscreen #2,#st_6,#spen    ;open screen
 beq Pr_bad
 TLwindow #0,#20,#10,#50,#25,#350,#75,#0,#st_7 ;open window 0
 beq Pr_bad
 TLwindow #1,#30,#15,#50,#25,#350,#75,#0,#st_8 ;open window 1
 beq Pr_bad
 TLwindow #2,#40,#20,#50,#25,#350,#75,#0,#st_9 ;open window 2
 beq Pr_bad
 move.w #3,open             ;set number of open windows
 bra.s Pr_cont              ;(window 2 is active)

Pr_cycl:
 TLwpoll                    ;wait for a window to be active

Pr_cont:
 move.w xxp_Active(a4),d7   ;d7 is active window
 bsr Win0                   ;do this window until it becomes inactive
 tst.w open                 ;any windows still open?
 bne Pr_cycl                ;yes, recycle
 rts                        ;exit ok

Pr_bad:
```

```
 TLbad #2                  ;report if out of mem
 rts


* window D7 is active
Win0:
 move.b d7,d0              ;put window num in string 1
 add.w #'0',d0
 move.b d0,st_1b
 addq.b #1,st_1a           ;bump A-Z in string 1
 cmp.b #'Z'+1,st_1a
 bne.s W0_draw
 move.b #'A',st_1a

W0_draw:
 TLwupdate                 ;update window size
 TLstring #1,#10,#20       ;print string 1

W0_wait:
 TLkeyboard                ;wait at keyboard
 cmp.b #$96,d0             ;resized?
 beq Win0                  ;redraw if resized
 cmp.b #$93,d0             ;close?
 beq.s W0_close            ;close if close gadget
 cmp.b #$97,d0             ;inactive?
 bne W0_wait               ;keep waiting until inactive
 rts

W0_close:
 TLwsub d7                 ;close window
 subq.w #1,open            ;dec no. of open windows
 rts


                Forward to program 44
```

1.45 p44 Demonstrate Program 44

```
                Program 44

This program demonstrates TLmultiline, which attaches a text viewer, or
a text editor, to a window.

   Run Program 44    -> TLmultiline - text viewing/editing

                Source program 44
                 -> Teaching/44.asm

TLmultiline is a large & complex routine, with many features. You will
notice it has built into it its own AmigaGuide, which you can peruse, as
well as lots of context sensitive help attached to the help button.

If you load a large text file (e.g. Tandem/Tandem.guide), you will see
that TLmultiline can handle quite large files without slowing down much.
```

```
(Tandem's sc and jotting windows use TLmultiline with some functions
 disabled).


                Forward to Program 45
```

1.46 44asm

```
                Forward to program 45
                * 44.asm  TLmultiline     version 0.01     8.6.99


 include 'Front.i'          ;*** change to 'Tandem.i' to step thru TL's ***


; TLmultiline is a very large and complex program. It allows the user to
; co-opt a window temporarily to display a set of lines, and perhaps to
; edit them. It is a full-blown text editor in its own right. TLmultiline
; also has its own Amiga.guide which the user of TMmultiline can peruse.
; The version of TLmultiline in release 2 of tandem.library is crippled, as
; it only edits plaintext, without character & line styling or graphics.

; TLmultiline when it takes over a window remembers its attributes, and
; restores those attributes to its host window when it exits. The program
; below demonstrates this.

; TLmultiline has extensive built-in online help.


strings: dc.b 0
 dc.b 'TLMultiline demonstration',0 ;1
 dc.b 'Done!!',0 ;2
st_3: dc.b 'a TLMultiline window!',0 ;3
 dc.b 'This window will become a TLmultiline window.',0 ;4
 dc.b 'TLmultiline is a built-in text editor.',0 ;5
 dc.b 'There is context sensitive help, via the',0 ;6
 dc.b '<Help> key, and it also has its own menu.',0 ;7

 ds.w 0


menu:
 TLnm 1,13
 TLnm 2,14
 TLnm 2,-1
 TLnm 2,15
 TLnm 4,0


* demonstrate TLMultiline
Program:
 TLwindow #0,#0,#0,#380,#120,#640,#256,#-1,#st_3 ;open window 0
 beq Pr_quit
```

```
 TLreqinfo #4,#4,#0        ;show preliminary info
 beq.s Pr_quit             ;quit if can't
 move.l xxp_AcWind(a4),a5       ;point to current window
 move.w #76,xxp_Mmxc(a5)        ;max 76 chrs/line    } override defaults
 move.l #1000000,xxp_Mmsz(a5)   ;mem size 1000000    } before calling

Pr_mult:
 TLmultiline #xxp_xmsty,#xxp_xesty ;* run TLMultiline (all styl forbidden)
 tst.l xxp_errn(a4)
 bne.s Pr_quit             ;quit if error
 cmp.b #$97,xxp_kybd+3(a4) ;redo if merely stopped because inactive window
 beq Pr_mult
```

Forward to program 45

1.47 p45 Demonstrate Program 45

Program 45

TLfloat allows the user to type in a floating-point number, in very free format, and processes it into a bcd string ready to load into an FPU register.

Run Program 45 -> input a floating point number

Source program 45
-> Teaching/45.asm

(This routine does not actually require an FPU (or 68040+), but of course its output would be used by an FPU).

Forward to Program 46

1.48 45asm

Forward to program 46
* 45.asm TLfloat version 0.01 8.6.99

```
 include 'Front.i'         ;*** change to 'Tandem.i' to step thru TL's ***


; TLfloat allows you to change an ASCII string to the format that an FPU
; uses for its input. Strangely, tandem.library does not use the FPU to
; do TLfloat, but of course the FPU would immediately be called to FMOVE
; TLfloat's output into one of its registers.
```

```
; TLfloat points A0 to the delimiter of the float, so for example if you
; input  123.45,  then TLfloat will return with A0 pointint to the "," which
; is the delimiter of the float. Thus, you can use TLfloat to evaluate
; 1 by 1 sets of floats separated by commas, spaces, or whatever. TLfloat
; is very flexible and hopefully clever in what it will accept as an input.


strings: dc.b 0
st_1: dc.b 'Test TLFloat',0 ;1
 dc.b 'Input a float (e.g. -123.45, 1.76E-5)',0 ;2
 dc.b 'bad: no mantissa digits ',0 ;3
 dc.b 'bad: abs(exponent)>999 after normalising ',0 ;4
 dc.b 'bad: no digits after E ',0 ;5

 ds.w 0


* demonstrate TLfloat
Program:
 TLwindow #-1              ;initialise
 beq Pr_quit

Pr_cyc:
 clr.b (a4)
 TLreqinput #2,str,#25     ;get next input
 beq Pr_quit               ;done if cancel

 move.l a4,a0              ;tfr input to buff+100
 move.l a4,a1
 move.l a1,a2
 add.l #100,a2             ;a2=buff+100
 move.l a2,a1
Pr_tfr:
 move.b (a0)+,(a1)+
 bne Pr_tfr

 move.l a4,a3              ;a3=buff+200
 add.l #200,a3

 TLfloat a2,a3             ;put float in buff+200 (12 bytes)
 bne.s Pr_good             ;go if good

 addq.w #2,d0              ;convert d0 to error string num
 TLstra0 d0                ;point to error string
 move.l a4,a1
Pr_bad:
 move.b (a0)+,(a1)+        ;error string to buffer
 bne Pr_bad
 bra.s Pr_pik              ;append for input

Pr_good:
 move.l a3,a0              ;convert ouput to ASCII
 move.l a4,a1
 moveq #5,d1               ;(6 words)

Pr_word:
```

```
 move.w (a0)+,d0           ;get word
 moveq #3,d2               ;(4 nybbles)

Pr_nybb:
 rol.w #4,d0               ;get next nybble
 move.w d0,d3              ;convert to hex
 and.w #15,d3
 add.b #'0',d3
 cmp.b #':',d3
 bcs.s Pr_asc
 add.b #'A'-':',d3         ;(only 1st byte should get here!)

Pr_asc:
 move.b d3,(a1)+           ;put ASCII of nybble
 dbra d2,Pr_nybb
 move.b #' ',(a1)+         ;spc between words
 dbra d1,Pr_word

Pr_pik:
 move.b #'}',(a1)+         ;append '}' then input
 move.b #' ',(a1)+
 move.l a2,a0

Pr_app:
 move.b (a0)+,(a1)+
 bne Pr_app

 TLreqchoose               ;show 'output } input', wait for acknowledge
 bra Pr_cyc                ;repeat until cancel

Pr_quit:
 rts
```

Forward to program 46

1.49 p46 Demonstrate Program 46

Program 46

This demonstrates TLbusy and TLunbusy, which make the pointer busy or unbusy, using the amigados 2.04 or 3.1 method as appropriate.

Run Program 46 -> busy & un-busy pointers

Source program 46
-> Teaching/46.asm

Forward to Program 49

1.50 46asm

```
                Forward to program 49
                * 46.asm    TLbusy  TLunbusy      version 0.0  1.9.97


; tandem.library has 2 routines for the busy pointer:
;
; TLbusy     implements the busy pointer (for this window only)
; TLunbusy   returns to the normal pointer


 include 'Front.i'


strings: dc.b 0
st_1: dc.b 'Teaching/46.asm',0 ;1
 dc.b 'Click me to make me busy',0 ;1
 dc.b 'Click me to make me un-busy',0 ;2
 dc.b 'Click me to quit',0 ;3
 dc.b 'Busysetup failed - out of mem',0
 dc.b 'Can''t start - out of mem',0 ;5
 dc.b 'Teaching/45.asm',0 ;6

 ds.w 0


* control overall execution
Program:
 TLwindow0                ;open standard window
 beq Pr_bad               ;bad if can't

 TLstring #2,#10,#5       ;wait for response
 TLkeyboard
 TLbusy                   ;busy pointer

 TLstring #3,#10,#18      ;wait for response
 TLkeyboard
 TLunbusy                 ;restore normal pointer

 TLstring #4,#10,#31      ;wait for final response
 TLkeyboard
 bra.s Pr_done            ;quit ok

Pr_bad:
 TLbad #6                 ;report if TLwindow failed
 rts

Pr_done:
 rts


                Forward to program 49
```

1.51 p49 Demonstrate Program 49

```
                Program 49

The second TLreqshow demo.

   Run Program 49    -> TLreqshow II

                Source program 49
                 -> Teaching/49.asm



                Forward to Program 50
```

1.52 49asm

```
                Forward to program 50
                * 49.asm     TLreqshow II    version 0.01   8.6.99


 include 'Front.i'


; This program demonstrates the use of TLreqshow to view classified data
; with dynamic contents


strings: dc.b 0
 dc.b 'View subset 0',0 ;1
 dc.b 'Some Catalogued data (n.b. press <Help> for assistance)',0 ;2
 dc.b '** Click one of the lines below to select which subset **',0 ;3
 dc.b '** Click this line to return to the subset catalogue **',0 ;4
 dc.b 'Subset 0 String     ',0 ;5
 dc.b 'Error: out of memory',0 ;6

 ds.w 0


subset: ds.w 1                   ;subset selected (-1 if in catalogue list)


* test program
Program:
 TLwindow #-1
 move.w #-1,subset               ;flag no subset is yet selected
 TLreqshow #Hook,#2,#20,#20      ;hook=Hook title=2 lines=20 shown=20
 rts

Pr_bad:
 TLbad #6
 rts
```

```
* Act as hook for TLreqshow
Hook:
 tst.w subset          ;go if we are in one of the subsets
 bpl.s Ho_sub          ;(if sub=-1, we are still in the catalogue list)
 tst.l d0              ;go if a line clicked
 bmi.s Ho_clkd
 bne.s Ho_cat          ;go if d0>0
 TLstrbuf #3           ;line 0: send string 3
 move.l a4,a0
 rts

Ho_cat:
 cmp.w #5,d0           ;if D0>4, send blank line (lines 5-19 blank)
 bcc.s Ho_null
 TLstrbuf #1           ;send string 1
 add.b d0,12(a4)       ;poke subset num in last chr
 move.l a4,a0          ;a0=string to display
 rts

Ho_null:
 clr.b (a4)            ;send blank line
 move.l a4,a0
 rts

Ho_clkd:               ;catalogue list has been clicked
 bclr #31,d0           ;d0=line clicked
 tst.w d0
 beq.s Ho_clkn         ;if line 0 was clicked, do nothing
 cmp.w #5,d0
 bcc.s Ho_clkn         ;if (blank) line 5-19 clicked, do nothing
 move.w d0,subset      ;remember which subset clicked
 moveq #3,d0           ;return from click: request redraw in subset
 moveq #100,d1         ;each subset has 100 strings
 moveq #0,d3           ;new topline
 rts

Ho_clkn:               ;return from click: do nothing
 moveq #-1,d0
 rts

Ho_sub:                ;* we are in a subset
 tst.l d0
 bmi.s Ho_sclk         ;go if a subset line was clicked
 bne.s Ho_some         ;go if string > 0
 TLstrbuf #4
 move.l a4,a0          ;string 0: send string 4
 rts

Ho_some:
 TLstrbuf #5           ;string 5 to buffer
 move.w subset,d1
 add.b d1,7(a4)        ;poke subset num into string
 move.l a4,a0
 add.l #16,a0          ;point to item num in subset
 TLhexasc d0,a0        ;poke line num into string
 move.l a4,a0
```

```
 rts

Ho_sclk:              ;subset has been clicked
 tst.w d0
 bne Ho_clkn          ;do nothing unless 1st line clicked
 move.w #-1,subset
 moveq #3,d0          ;request redraw (back to catalogue window)
 moveq #20,d1         ;items=20
 moveq #0,d3          ;new topline
 rts


                Forward to program 50
```

1.53 p50 Demonstrate Program 50

```
                Program 50

Introduces TLreqcolor for choosing a pen, or adjusting the colour palette.
The Amiga Colour Wheel feature is very messy to program, though of course
more sophisticated than TLreqcolor.

   Run Program 50    -> Choose a pen / Set the palette

                Source program 50
                 -> Teaching/50.asm

TLreqcolor works ok on ECS machines. (Note: I plan to do an upgrade to
TLreqcolor which is more user friendly, but the same to program).



                Forward to Program 51
```

1.54 50asm

```
                Forward to program 51
                * 50.asm     TLreqcolor    version 0.01   8.6.99

 include 'Front.i'


; This program demonstrates the use of TLreqcolor to choose a colour/palette


strings: dc.b 0
 dc.b 'You selected  ',0 ;1
 dc.b 'Error: our of memory',0 ;2
 dc.b 'You selected cancel',0 ;3
```

```
 ds.w 0


* test program
Program:
 TLwindow #-1              ;set things up
 beq Pr_quit               ;quit if can't

 TLreqcolor #2             ;2=pen+palette+ld/sv
 subq.w #1,d0              ;d0 -> -1 if bad/cancel, or 0+ for pen chosen
 bmi.s Pr_bad2             ;go if bad/cancel

 TLstrbuf #1               ;string 1 to buff
 move.l a4,a0
 add.l #13,a0              ;point to after str 1 in buff
 TLhexasc d0,a0            ;put pen num selected
 clr.b (a0)                ;delimit
 bra.s Pr_rep              ;go report choice

Pr_bad1:                   ;here if TLwindow failes
 TLbad #2                  ;report error in monitor
 bra.s Pr_quit

Pr_bad2:                   ;here if TLreqcolor bad/cancel
 tst.l xxp_errn(a4)
 bne.s Pr_err              ;go if bad
 TLstrbuf #3               ;str 3 to buff
 bra.s Pr_rep              ;go report cancelled

Pr_err:                    ;here if TLreqcolor bad
 TLerror                   ;error report to buff

Pr_rep:                    ;report pen chosen / cancelled / error
 TLreqchoose

Pr_quit:
 rts
```

Forward to program 51

1.55 p51 Demonstrate Program 51

Program 51

It is a simple matter to modify the appearance of tandem.library requesters.
e.g. to add a logo, or attach some info, &c.

This program takes a normal TLreqchoose requester (with 1 choice only)
and puts a bevelled box with a checkmark on it.

Run Program 51 -> modifying tandem.library requesters.

Source program 51

```
                -> Teaching/51.asm

I should stress that this is just a beginning - you can and should do all
kinds of jazzy things to requesters to pretty them up, in accordance with
the Amiga tradition of making attractive user interfaces.


               Forward to Program 52
               Note: A basic utility like Tandem should be fast, compact, elegant ↩
                   , reliable
and intuitive - these are the primary aesthetic considerations. But to most
programs should be added beauty, style, flair and configurability.
Tandem.library gives you the first list simply and easily. It's up to you
to add the second list. Go for it...
```

1.56 51asm

```
               Forward to program 52
               * 51.asm      Requester hooks      0.01   8.6.99


; This program shows you how to use xxp_hook0,xxp_hook1 and xxp_hook2 to
; modify the appearance of requesters. This program puts a picture
; at the right of an otherwise minimal TLreqchoose requester.


 include 'Front.i'


strings: dc.b 0
 ds.w 0


* entry point
Program:
 TLwindow #-1              ;intialise everything (hooks get zapped)
 beq.s Pr_quit
 move.l #Hook0,xxp_hook0(a4) ;attach hook0
 move.l #Hook1,xxp_hook1(a4) ;attach hook1
 move.l #'OK! ',(a4)       ;text for requester
 clr.b 4(a4)
 TLreqchoose               ;put up minimal reqchoose requester

Pr_quit:
 rts


* make requester wider
Hook0:
 add.l #100,xxp_reqw(a4)   ;make requester 100 dots wider before drawing
 rts
```

```
* put bev on requester
Hook1:
 move.l xxp_reqw(a4),d0    ;draw a bev box at the rhs of the window
 sub.w #80,d0
 moveq #60,d2
 moveq #4,d1
 move.l xxp_reqh(a4),d3
 subq.l #8,d3
 TLreqbev d0,d1,d2,d3,rec

 add.w #26,d0              ;draw a tick in the bev
 subq.w #8,d3
 lsr.w #1,d3
 add.w d3,d1
 move.l d1,d2
 move.l d0,d1
 TLpict #10,d1,d2
 rts
```

Forward to program 52

1.57 p52 Demonstrate Program 52

Program 52

This demonstration is of TLgetilbm, allows you to load an IFF picture into a bitmap, from where you can use it e.g. to copy it to a window or a requester.

If you don't know what to load, then point the ASL requester to /logo.iff

Run Program 52 -> TLgetilbm - load an ILBM file

Source program 52
-> Teaching/52.asm

Of course, TLgetilbm is less sophisticated than using data types.

Forward to Program 53

1.58 52asm

Forward to program 53

```
* 52.asm  demonstrate TLGetilbm     version 0.01     2.3.98


 include 'Front.i'
```

```
strings: dc.b 0
st_1: dc.b 'Demonstrate TLgetilbm',0 ;1
 dc.b 'Filename of ILBM IFF file',0 ;2
 ds.w 0


fil: ds.b 34                ;ilbm fil, dir
dir: ds.b 130
bmhd: ds.l 1                ;buffer for bmhd data
bmap: ds.l 1                ;bitmap created by TLgetilbm call
rprt: ds.l 1                ;rastport used to hold bmap for ClipBlit


*>>>> demonstrate  TLGetilbm
Program:
 TLwindow #-1
 TLwindow #0,#0,#0,#640,xxp_Height(a4),#640,xxp_Height(a4),#0,#st_1
 beq Pr_bad
 TLpublic #790
 move.l d0,bmhd             ;mem for bmhd
 beq Pr_bad
 TLpublic #rp_SIZEOF
 move.l d0,rprt             ;mem for rport
 beq Pr_bad
 clr.b fil                  ;init fil,dir
 clr.b dir

Pr_cyc:                     ;view next picture
 TLreqcls
 TLaslfile #fil,#dir,#2,ld ;get filename of ilbm
 beq Pr_bad                 ;bad if can't

 TLgetilbm #2,bmhd          ;load ilbm into bmap
 beq Pr_bad                 ;bad if can't
 move.l a0,bmap             ;save bmap address

 bsr Blit                   ;blit the ilbm from the bitmap to the window
 TLfreebmap bmap            ;free the bitmap

 TLkeyboard                 ;wait for acknowledge
 cmp.b #$93,d0
 bne Pr_cyc                 ;recycle unless close window
 bra.s Pr_done

Pr_bad:                     ;report error if bad
 TLerror
 TLreqchoose

Pr_done:                    ;quit
 rts


*>>>> blit the ilbm from bmap to window 0
Blit:
 move.l xxp_gfxb(a4),a6     ;initialise rastport
```

```
 move.l rprt,a1
 jsr _LVOInitRastPort(a6)
 move.l rprt,a0            ;a0 = rprt
 move.l bmap,a2            ;a2 = bmap, attach to rport
 move.l a2,rp_BitMap(a0)

 move.l xxp_AcWind(a4),a5  ;a1 = window's rastport
 move.l xxp_WPort(a5),a1

 moveq #0,d0               ;from 0,0 to top left
 moveq #0,d1
 moveq #0,d2
 move.w xxp_LeftEdge(a5),d2
 moveq #0,d3
 move.w xxp_TopEdge(a5),d3

 moveq #0,d4               ;greater of bmap width, window width to d4
 move.w (a2),d4
 lsl.w #3,d4
 cmp.w xxp_PWidth(a5),d4
 bcs.s Bl_xlim
 move.w xxp_PWidth(a5),d4
Bl_xlim:

 moveq #0,d5               ;greater of bmap height, window height to d5
 move.w 2(a2),d5
 cmp.w xxp_PHeight(a5),d5
 bcs.s Bl_ylim
 move.w xxp_PHeight(a5),d5 ;trim to fit window
Bl_ylim:

 move.l #$C0,d6            ;minterm: $C0=vanilla
 jsr _LVOClipBlit(a6)      ;do the blit
 rts


               Forward to program 53
```

1.59 p53 Demonstrate Program 53

```
               Program 53

   Run Program 53    -> Save to an ILBM file.

               Source program 53
                -> Teaching/53.asm
```

This program demonstrates TLputilbm, which allows you to save all or part of a rastport (e.g. the screen) to a file. The above puts a little bit of the screen into a file called RAM:Temp.iff

In order to see what was saved after you clicked the above, go back to the previous node ("Browse <"), and run program 52, and input RAM:Temp.iff

```
                Forward to Program 54
```

1.60 53asm

```
                Forward to program 54
                * 53.asm  demonstrate TLPutilbm     version 0.01     8.6.99


 include 'Front.i'


; TLPutilbm can save a region from any bitmap as an IFF file.
; This program saves the workbench screen image as an IFF file. The program
; saves the workbench screen image to a file named "RAM:Temp.iff". You
; can use Multiview, or a paint program, to load RAM:Temp.iff to check
; that this program worked ok.


strings: dc.b 0
 dc.b 'RAM:Temp.iff',0     ;1
 dc.b 'Saved - from the CLI, enter  Multiview RAM:Temp.iff  to view it',0 ;2
 dc.b '(Or, load, assemble & run 53.asm, and input RAM:Temp.iff)',0 ;3

 ds.w 0


* demonstrate  TLPutilbm
Program:
 TLwindow #-1              ;initilise everything
 beq Pr_quit

 TLstrbuf #1               ;filename to buffer
 move.l xxp_Screen(a4),a1  ;a1 = the workbench screen
 move.l sc_RastPort+rp_BitMap(a1),a0 ;a0 = the workbench screen's bitmap
 TLputilbm #0,#0,#100,#50,a0  ;save the screen (0,0)-(99,49) in RAM:Temp.iff
 beq.s Pr_bad              ;go if bad

 TLreqinfo #2,#2           ;tell user how to view it
 bra.s Pr_quit

Pr_bad:                    ;report error if bad
 TLerror
 TLreqchoose

Pr_quit:
 rts


                Forward to program 54
```

1.61 p54 Demonstrate Program 54

```
                Program 54

This program demonstrates the TLprogress routine, which puts up a thermometer
showing progress of a task towards completion. You can specify either
no text, a ratio, or a %. This program shows the 3 options side by side:

   Run Program 54    -> TLprogress

                Source program 54
                 -> Teaching/54.asm

Once again, it is very easy to program. You need merely poke the position
& dimensions of the thermometer into A4 offsets, & then call TLprogress with
3 paramters for total, progress & option each time you want to (re)draw it.


                Forward to program 55
```

1.62 54asm

```
                Forward to program 55
                * 54.asm  TLprogress     version 0.01     8.6.99


 include 'Front.i'


prog: ds.l 1                ;keeps progress


strings: dc.b 0
st_1: dc.b 'TLProgress demonstration',0 ;1
 dc.b 'Error: out of chip memory',0 ;2
 dc.b 'Wait until I''m full',0 ;3
 dc.b 'Click to clear      ',0 ;4
 dc.b 'Click to quit       ',0 ;5

 ds.w 0


* demonstrate TLprogress
Program:
 TLwindow #0,#0,#0,#100,#100,#500,#200,#0,#st_1
 beq Pr_bad

 TLstring #3,#2,#1          ;print 'wait until full'

 move.l #10,xxp_prgd+4(a4)  ;thermometer ypos   (xpos set at Pr_wait+1,3)
 move.l #100,xxp_prgd+8(a4) ;            width
 move.l #10,xxp_prgd+12(a4) ;            height
```

```
 clr.l prog

Pr_wait:                    ;report progress
 move.l #20,xxp_prgd(a4)
 TLprogress prog,#50        ;draw thermometer without text

 move.l #130,xxp_prgd(a4)
 TLprogress prog,#50,txt    ;draw thermometer with text

 move.l #240,xxp_prgd(a4)
 TLprogress prog,#50,%      ;draw thermometer with %

 TLbusy

 moveq #4,d7                ;d7 = delay factor (make d7 bigger for slower)
 move.l xxp_gfxb(a4),a6
Pr_paus:
 jsr _LVOWaitTOF(a6)
 dbra d7,Pr_paus

 addq.l #1,prog             ;bump total while prog < 51
 cmpi.l #51,prog
 bne Pr_wait

 TLunbusy

 TLstring #4,#2,#1          ;wait for acknowledge & quit ok
 TLkeyboard
 bra.s Pr_quit

Pr_bad:                     ;here if out of mem
 TLbad #2

Pr_quit:
 rts
```

```
                Forward to program 55
```

1.63 p55 Demonstrate Program 55

```
                Program 55

This program shows TLData, which unlike tandem.library's other requesters,
runs asynchronously.

   Run Program 55    -> TLdata - an asynchronous requester

                Source program 55
                 -> Teaching/55.asm

The idea is, you call TLData with the dimensions you want, and text on it
as required. Then your program continues to run - you could get your program
```

to write things on it, or get IDCMPs from it, as required. Finally, you call TLreqoff to remove it. (This is called "asynchronous", unlike other requesters which take over your program until they return, & are hence "synchronous").

Forward to program 56

1.64 55asm

```
                Forward to program 56
                * 55.asm  TLdata     version 0.01     8.6.99


 include 'Front.i'


strings: dc.b 0
 dc.b '     I''m a TLdata demonstration...',0 ;1
 dc.b 'I will wait on the screen for a few seconds.',0 ;2
 dc.b 'Then, I''ll disappear. So, parden me a few',0 ;3
 dc.b 'seconds while I contemplate the relationship',0 ;4
 dc.b 'between mind and matter.',0 ;5
 dc.b 'Error: out of memory',0 ;6

 ds.w 0

* demonstrate TLdata
Program:
 TLwindow #-1               ;set things up
 beq Pr_bad

 TLdata #2,#4               ;draw data window

 move.l xxp_dosb(a4),a6     ;delay 7 seconds
 move.l #7*50,d1
 jsr _LVODelay(a6)

 TLreqoff                   ;remove data window
 bra.s Pr_quit              ;quit ok

Pr_bad:                     ;here if out of mem
 TLbad #6

Pr_quit:
 rts


                Forward to program 56
```

1.65 p56 Demonstrate Program 56

```
                Program 56

This is the third demo of TLreqshow

   Run Program 56    -> TLreqshow III

                Source program 56
                 -> Teaching/56.asm

These three demos you have seen of TLreqshow are just the begining though;
you can do all sorts of jazzy things.

For example, you could put up several lines, and the next call to your
hook could also put a picture near the three lines. It all depends on your
programming skill and creativity. Another example, the hook could play
different sounds when particular lines are clicked.


                Forward to program 57
```

1.66 56asm

```
                Forward to program 57
                * 56.asm     TLreqshow with smart search    version 0.01    8.6.99


 include 'Front.i'


; This program demonstrates the use of TLReqshow, which allows the user to
; view a series of lines, while your program calculates their contents
; dynamically. It includes a smart search. Sometimes, as here, the calling
; program knows how to find strings quickly, which saves TLReqshow from
; looking through the strings one by one, a slow process if there are many
; strings. In such cases, use smart search.

; TLreqshow is a powerful requester, and though writing a hook is admittedly
; rather difficult, they can save you a lot of work. See, for example, the
; TLreqshow that comes up in TLMultiline when you request "Print" from the
; menu. TLreqshow there saves a lot of coding.


strings: dc.b 0
 dc.b 'Some strings for your delectation (n.b. press <Help> for info)',0 ;1
st_2: dc.b 'String   ',0 ;2
 dc.b 'You have seen a TLReqshow requester (sigh!)',0 ;3
 dc.b '(press <Help> for more help!)',0 ;4
 dc.b 'Alas! This wonderful TLReqshow demo is finished!',0 ;5
 dc.b '(Wasn''t it great!!! Applause! Applause!)',0 ;6
 dc.b 'This program (i.e. 56.asm) will put up a TLreqshow requester.',0 ;7
```

```
 dc.b ' ',0 ;8
 dc.b '56.asm is programmed to do "smart search". So when you press',0 ;9
 dc.b 'any of the "Seek" buttons on the TLReqshow, the search will be',0 ;10
 dc.b 'very fast.',0 ;11
 dc.b 'Error: out of mem',0 ;12

 ds.w 0

clikd: ds.l 1 ;line selected
maxi: ds.w 1  ;no. of lines

* test program
Program:
 TLwindow #-1              ;set things up
 beq Pr_bad1

 TLreqinfo #7,#5           ;preliminary info

 move.l #100000,maxi       ;set number of lines
 move.l #-1,clikd          ;flag no line is yet clicked
 TLreqshow #Hook,#1,maxi,#17,#0,smart
 beq.s Pr_bad2             ;go if TLreqshow fails

 move.w #5,xxp_Help(a4)    ;attach help
 move.w #2,xxp_Help+2(a4)
 TLreqinfo #3,#2           ;final message
 bra.s Pr_quit

Pr_bad1:                   ;here if TLwindow failed
 TLbad #12                 ;report to CLI
 bra.s Pr_quit

Pr_bad2:                   ;here if TLreqshow failed
 TLerror                   ;report cause of error
 TLreqchoose

Pr_quit:
 rts


* Act as hook for TLReqshow
Hook:
 tst.l d0                  ;go if line clicked
 bmi.s Ho_clkd
 bsr Make                  ;synthesize line d0, point a0 to it
 rts
Ho_clkd:
 btst #30,d0               ;go if smart search
 bne.s Ho_smrt
 cmp.l clikd,d0            ;line already highlighted?
 bne.s Ho_on               ;no, go
 move.l #-1,clikd
 moveq #1,d0               ;highlighting off
 rts
Ho_on:
 move.l d0,clikd           ;remember which line is being highlighted
 moveq #2,d0               ;turn highlighting on
```

```
 rts
Ho_smrt:
 and.l #$3FFFFFFF,d0       ;d0 = start from, d1 = 3/4/5 = fore/back/left
 bsr Smart                 ;do a smart search
 rts


* synthesize line d0
Make:
 TLstrbuf #2               ;string 2 to buffer
 move.l a4,a0
 addq.l #8,a0
 move.l #'    ',(a0)       ;blank num
 TLhexasc d0,a0            ;put num
 clr.b (a0)
 move.l a4,a0              ;point a0 to string as synthesized
 rts


* do a smart search   D0 = start from,  D1 = 3/4/5 = fore/back/left
Smart:
 move.l d0,d2              ;cache d0 in d2
 move.l a4,a0              ;point a5 to sought
 add.w #xxp_patt,a0
 move.b (a0),d3            ;d3 is the first chr of sought
 beq.s Sm_1                ;(match anything if null string - can't happen?)
 lea st_2,a1
Sm_st2:                    ;see if 1st chr in st_2
 tst.b (a1)
 beq.s Sm_num              ;no, assume all numbers
 cmp.b (a1)+,d3
 bne Sm_st2
 subq.l #1,a1              ;a1 is 1st chr of st_2 that matches sought
 cmp.b #' ',d3
 bne.s Sm_cmp              ;go if not blank
 cmp.b #' ',1(a0)
 beq.s Sm_cmp              ;ok if 2nd chr of sought also blank
 addq.l #1,a1              ;else, match w. 2nd blank of st_2
Sm_cmp:
 cmpm.b (a0)+,(a1)+        ;match each chr of sought w. st_2
 bne.s Sm_no               ;no if mismatch
 tst.b (a0)
 beq.s Sm_1                ;always matches if all match w. st_2
 cmp.b #'0',(a0)
 bcs Sm_cmp                ;stop matching with st_2 if 0-9 found
 cmp.b #'9'+1,(a0)
 bcc Sm_cmp
Sm_num:                    ;seek value of number at a0
 TLaschex a0               ;d0 = value
 tst.b (a0)                ;did all chrs get bypassed?
 bne.s Sm_no               ;no, garbage after number
 cmp.w #4,d1
 bcs.s Sm_fore             ;go if forward
 bne.s Sm_maxi             ;go if left
 cmp.l d2,d0
 bcc.s Sm_no               ;backward: no if d0 not backward
Sm_yes:
```

```
 rts
Sm_fore:
 cmp.l d0,d2               ;forward: no if d0 not forward
 bcc.s Sm_no
Sm_maxi:
 cmp.l maxi,d0             ;forward: no if d0 out of range
 bcs Sm_yes
Sm_no:
 moveq #-1,d0              ;d0 = -1 if unfound
 rts
Sm_1:                      ;here if always matches
 moveq #0,d0
 cmp.w #5,d1
 beq Sm_yes                ;string 0 if left
 move.l d2,d0
 addq.l #1,d0
 cmp.w #3,d1               ;string +1 if forward
 beq Sm_maxi
 subq.l #2,d0              ;string -1 if backward
 rts
```

Forward to program 57

1.67 p57 Demonstrate Program 57

Program 57

This program demonstrates TLresize which resizes portions of a window or rastport.

Run Program 57 -> TLresize

Source program 57
-> Teaching/57.asm

TLresize operates on horizontal & vertical dimensions independently, as you can see from the demonstration.

Forward to program 58

1.68 57asm

Forward to program 58

```
* 57.asm     TLResize     version 0.01    8.6.99


 include 'Front.i'
```

```
strings: dc.b 0
st_1: dc.b 'TLResize demo',0 ;1
 dc.b 'Error: Out of chip memory',0 ;2
 dc.b 'Hello, world',0 ;3
 dc.b 'TLresize demo...',0 ;4
 dc.b ' ',0 ;5
 dc.b 'I am about to use TLresize to resize 2 of the 3 strings',0 ;6
 dc.b 'on the window.',0 ;7
 dc.b 'TLresize demo...',0 ;8
 dc.b ' ',0 ;9
 dc.b 'As you can see, the strings are now resized.',0 ;10
 dc.b '<- Narrower, taller',0 ;11
 dc.b '<- wider, shorter',0 ;12

 ds.w 0


* test program
Program:
 TLwindow #0,#0,#0,#640,#200,#640,#200,#0,#st_1 ;open window 0
 bne.s Pr_cont              ;go if ok
 TLbad #2                   ;report if can't open window
 rts

Pr_cont:
 TLstring #3,#0,#0          ;normal size at 0,0
 TLstring #3,#0,#12         ;again at 0,12 - will be resized
 TLstring #3,#0,#24         ;again at 0,24 - will be resized

 move.l xxp_AcWind(a4),a5  ;reposition requesters so they don't hide strings
 move.w #70,xxp_ReqTop(a5)
 TLreqinfo #4,#4            ;threaten to resize the strings

 TLresize #0,#12,#96,#8,#48,#12   ;resize the 2nd string
 TLresize #0,#24,#96,#8,#192,#6   ;resize the 3rd string

 TLstring #11,#220,#12
 TLstring #12,#220,#24
 TLreqinfo #8,#3            ;confess
 rts


                Forward to program 58
```

1.69 p58 Demonstrate Program 58

```
                Program 58

This program demonstrates TLellipse which draws an ellipse on a window or
rastport. TLellipse clips the edges of the ellipse if it doesn't fit.

   Run Program 58    -> TLellipse
```

```
Source program 58
 -> Teaching/58.asm


Forward to program 59
```

1.70 58asm

```
Forward to program 59
* 58.asm     TLellipse I    version 0.01    8.6.99


 include 'Front.i'


strings: dc.b 0
st_1: dc.b 'For connaisseurs of ellipses!',0 ;1
 dc.b 'Error: out of chip memory',0 ;2
 dc.b 'An unclipped Ellipse        An Ellipse clipped 1 pixel all around',0
 dc.b 'A Hollow Ellipse            2 Hollow Ellipses, clipped 1 pixel, '
 dc.b 'lapped',0 ;4
 dc.b 'A bevelled ellipse...',0 ;5

 ds.w 0


* test program
Program:
 TLwindow #0,#0,#0,#640,#200,#640,#200,#0,#st_1 ;open window 0
 bne.s Pr_cont             ;go if ok
 TLbad #2                  ;report if can't open window
 rts

Pr_cont:
 TLstring #3,#2,#2
 move.l xxp_AcWind(a4),a5
 move.b #2,xxp_FrontPen(a5)
 TLellipse #100,#40,#90,#30,#0,#0,#640,#200,,solid    ;solid, untrimmed
 TLellipse #400,#40,#90,#30,#311,#11,#489,#69,,solid  ;solid, trimmed
 move.b #1,xxp_FrontPen(a5)
 TLstring #4,#2,#80
 move.b #2,xxp_FrontPen(a5)
 TLellipse #100,#120,#90,#30,#0,#0,#640,#200          ;outline, untrimmed
 TLellipse #400,#120,#90,#30,#311,#91,#489,#149       ;} outline, trimmed &
 TLellipse #401,#120,#90,#30,#312,#91,#490,#149       ;}           lapped
 TLkeyboard

 TLreqcls                                        ;do "bevelled" ellipse
 TLstring #5,#2,#2
 TLellipse #316,#90,#150,#75,#167,#16,#465,#164,,solid   ;white
 move.b #1,xxp_FrontPen(a5)
```

```
 TLellipse #324,#94,#150,#75,#175,#20,#474,#168,,solid   ;black
 move.b #3,xxp_FrontPen(a5)
 TLellipse #320,#92,#150,#75,#171,#18,#471,#166,,solid   ;blue
 TLkeyboard

 rts
```

Forward to program 59

1.71 p59 Demonstrate Program 59

Program 59

This program demonstrates TLgetarea which allows the user to select a region on a window, like in paint programs.

Run Program 59 -> TLgetarea

Source program 59
-> Teaching/59.asm

Once again, this routine is very simple to program. You merely put in the TLgetarea MACRO with parameters for the bounds of the area on the window from which the area is to be chosen. (or 4 0's for the whole area within the window borders, as in the above demo).

Forward to program 60

1.72 59asm

Forward to program 60

```
* 59.asm     TLgetarea and some bevs   version 0.01    8.6.99


 include 'Front.i'        ;*** replace by 'Tandem.i' to step thru TL's ***


strings: dc.b 0
st_1: dc.b 'TLgetarea demo + some bevs',0 ;1
 dc.b 'Error: out of chip memory',0 ;2
 dc.b 'Select any region on this window ....',0 ;3
 dc.b ' ',0 ;4
 dc.b '1. Move the mouse pointer to the top left of the region.',0 ;5
 dc.b '2. Press the left mouse button down.',0 ;6
 dc.b '3. Move the mouse pointer to the bottom right of the region.',0 ;7
 dc.b '4. Release the left mouse button.',0 ;8
 dc.b ' ',0 ;9
```

```
 dc.b 'The region you select will then be highlighted.',0 ;10
 dc.b '(Alternately, you can press Esc to cancel)',0 ;11
 dc.b 'You chose to cancel',0 ;12
 dc.b 'Chosen region highlighted',0 ;13

 ds.w 0


* test program
Program:
 TLwindow #0,#0,#0,#640,#200,#640,#200,#0,#st_1 ;open window 0
 bne.s Pr_cont             ;go if ok
 TLbad #2                  ;report if can't open window
 rts

Pr_cont:
 TLreqbev #20,#100,#70,#20      ;* do some bevs (just for novelty value)
 TLreqbev #120,#100,#70,#20,rec ;bev 2
 TLreqbev #218,#99,#74,#22      ;bev 3
 TLreqbev #220,#100,#70,#20,rec
 TLreqbev #318,#99,#74,#22,rec  ;bev 4
 TLreqbev #320,#100,#70,#20
 TLreqbev #419,#100,#72,#20     ;bev 5
 TLreqbev #420,#100,#70,#20
 TLreqbev #519,#100,#72,#20,rec ;bev 6
 TLreqbev #520,#100,#70,#20,rec
 TLreqbev #80,#135,#70,#20      ;bev 7
 TLreqbev #70,#130,#70,#20
 TLreqarea #72,#131,#66,#18
 TLreqbev #216,#128,#78,#24     ;bev 8
 TLreqbev #220,#130,#70,#20
 TLreqbev #316,#128,#78,#24     ;bev 9
 TLreqbev #320,#130,#70,#20,rec
 TLreqbev #416,#128,#78,#24,rec ;bev 10
 TLreqbev #420,#130,#70,#20,rec
 TLreqbev #516,#128,#78,#24,rec ;bev 11
 TLreqbev #520,#130,#70,#20
 moveq #3,d0               ;print instructions (string 3-10)
 moveq #8,d2
 moveq #8,d3

Pr_info:
 TLstring d0,#16,d2
 addq.w #1,d0
 addq.w #8,d2
 dbra d3,Pr_info
 move.l xxp_AcWind(a4),a5
 move.l xxp_Window(a5),a0
 move.w xxp_LeftEdge(a0),d0 ;set d0-d3 to limits of printable area
 move.w xxp_TopEdge(a0),d1
 move.w xxp_PWidth(a0),d2
 sub.w d0,d2
 subq.w #1,d2
 move.w xxp_PHeight(a0),d3
 sub.w d1,d3
 subq.w #1,d3
```

```
 TLgetarea d0,d1,d2,d3,a4  ;select in printable area
 beq.s Pr_canc             ;go if cancelled

 move.l xxp_gfxb(a4),a6    ;set window draw mode to complement
 move.l xxp_WPort(a5),a1
 moveq #RP_COMPLEMENT,d0
 jsr _LVOSetDrMd(a6)

 move.l (a4),d0            ;complement area selected
 move.l 4(a4),d1
 move.l 8(a4),d2
 sub.l d0,d2
 addq.l #1,d2
 move.l 12(a4),d3
 sub.l d1,d3
 addq.l #1,d3
 subq.w #4,d0              ;make rel to printable area
 sub.w #11,d1
 TLreqarea d0,d1,d2,d3
 TLreqinfo #13             ;highlighted: wait for acknowledge
 rts

Pr_canc:
 TLreqinfo #12             ;cancel: wait for acknowledge
 rts
```

Forward to program 60

1.73 p60 Demonstrate Program 60

Program 60

This program demonstrates TLreqedit, which displays or edits a line of text. Be sure to press <Help> when it is running, so you can see all the things you can do, like italics, double width font, etc.

Run Program 60 -> TLreqedit

Source program 60
-> Teaching/60.asm

TLreqedit has many options. If the text deosn't fit within its tablet, it will scroll the text sideways. It will trim if the tablet doesn't fit in the current window size, and can continue (if required) if the window is resized while it is operating, adjusting its trimming as required.

Click this button, or the button at the end, when you've read the text below

Forward to Program 61
One of TLreqedit's specialties is the ability to spread text ↩ evenly across
the line, to full justify it like in professional printing (or it can

```
left justify or right justify or centre).

Many of the options (such as jam1, small caps) are really applicable to
using TLreqedit as a text displayer rather than text editor. See TLreqedit
in the optional supplement of Tandem.guide to see an explanation of all its
options.

Although TLreqedit is quite simple to use, the source for program 60 is
fairly large, as it has many options for choosing the tags for TLreqedit.


                Forward to program 61
```

1.74 60asm

```
                Forward to program 61
                * 60.asm     Demonstrate TLreqedit   version 0.01    8.6.99


Sorry!

 60.asm is large, and it would be wasteful to duplicate here.

 You'll find it in  Tandem/Teaching/60.asm


                Forward to program 61
```

1.75 p61 Demonstrate Program 61

```
                Program 61

   Run Program 61    -> Manage a set of fonts

                Source program 61
                 -> Teaching/61.asm

TLreqfont allows the user to manage and/or choose from a set of up to 10
fonts.


                Forward to program 62
                Note: I plan to make a prettier & more user friendly version of  ↩
                   this
requester in the next release, programmed the same way.
```

1.76 61asm

```
                Forward to program 62
                * 61.asm  TLReqfont     version 0.01     8.6.99


 include 'Front.i'


strings: dc.b 0
st_1: dc.b 'Demonstrate TLReqfont',0 ;1
 dc.b 'You chose /',0 ;2
 dc.b 'You chose cancel',0 ;3
st_4: dc.b 'Times.font',0 ;4

 ds.w 0


* demonstrate TLReqfont
Program:
 TLwindow #0,#0,#0,#380,#120,#640,#256,#-1,#st_1 ;open window 0
 beq Pr_bad
 TLgetfont #st_4,#1,#24

 TLreqfont #0            ;show TLreqfont
 beq.s Pr_bad             ;go if error
 move.w d0,d1
 bmi.s Pr_canc            ;go if cancel
 TLstrbuf #2
 add.b d1,10(a4)
 TLreqchoose              ;report choice
 bra.s Pr_quit

Pr_canc:                  ;report cancelled
 TLreqinfo #3,#1
 bra.s Pr_quit

Pr_bad:                   ;report error
 TLerror
 TLreqchoose

Pr_quit:
 rts

Pr_quit:
 rts


                Forward to program 62
```

1.77 p62 Demonstrate Program 62

```
                Program 62

This program is the first of 2 demonstrations of TLwscroll, which
manages window scrollers.

   Run Program 62    -> TLscroller I

                Source program 62
                 -> Teaching/62.asm

All you have to do to add scrollers to a window is to put #-1 for \8 of
a TLwindow call.

TLwscroll allows you to re-position the scrolers, and the above program
shows you the inputs that come from TLkeyboard when you move the scrollers,
or click their arrow boxes.

Be sure to resize the window as part of the demo. The program wil help you
to write programs to manage TLkeyboard input from scrollers.


                Forward to program 63
```

1.78 62asm

```
                Forward to program 63
                * 62.asm   Demonstrate Scrollers    0.01   8.6.99


 include 'Front.i'

; This program is useful, as it shows the IDCMP's that come in
; as you fiddle with the scrollers.


* text strings
strings: dc.b 0
st_1: dc.b 'Set up Window with Scrollers',0 ;1
 dc.b 'Out of Chip memory',0 ;2
 dc.b 'Message  IAddress Code Class    Qual MsX  MsY',0 ;3
 dc.b 'This program is about to open a window with scrollers.',0 ;4
 dc.b 'As you fiddle with the scrollers, resize the window, &c,',0 ;5
 dc.b 'you will see on the window the results of each call to',0 ;6
 dc.b 'TLwindow. This should help you to understand how to make',0 ;7
 dc.b 'your programs responsive to scrollers.',0 ;8

 ds.w 0


* set up window with scrollers
Program:
 TLwindow #-1              ;preliminary info
```

```
 TLreqinfo #4,#5

 TLwindow #0,#0,#0,#600,#180,#640,#200,#-1,#st_1 ;\8=-1 for scrollers
 bne.s Pr_cont

Pr_bad:
 TLbad #2
 bra Pr_quit

Pr_cont:
 move.l xxp_AcWind(a4),a5
 move.l xxp_scrl(a5),a3
 move.l #32,xxp_hztp(a3)
 move.l #64,xxp_hzvs(a3)
 move.l #256,xxp_hztt(a3)
 move.l #32,xxp_vttp(a3)
 move.l #64,xxp_vtvs(a3)
 move.l #256,xxp_vttt(a3)
 TLwscroll set

 TLstring #3,#0,#0
 moveq #0,d7               ;message count
 bsr Wait                  ;wait until close window

Pr_quit:
 nop
 nop
 rts


* wait for close window, show mousemoves
Wait:
 TLwcheck
 beq.s Wt_cu
 TLwupdate
Wt_cu:
 TLkeyboard                ;get response

 movem.l d0-d7/a0-a6,-(a7) ;scroll previous messages
 move.l xxp_gfxb(a4),a6
 move.l xxp_AcWind(a4),a5
 move.l xxp_Window(a5),a0
 move.l wd_RPort(a0),a1
 moveq #0,d2
 move.w xxp_LeftEdge(a5),d2
 move.l d2,d4
 add.w #479,d4
 moveq #0,d3
 move.w xxp_TopEdge(a5),d3
 addq.w #8,d3
 move.l d3,d5
 add.w #127,d5
 moveq #0,d0
 moveq #8,d1
 jsr _LVOScrollRaster(a6)
 movem.l (a7)+,d0-d7/a0-a6
```

```
 move.l xxp_mesg(a4),a1    ;collect message data    d0 = ascii / TL code
 move.l im_IAddress(a1),d5 ;d5 = IAdress            d3 = modified qualifier
 move.l im_Class(a1),d4    ;d4 = class              d1,d2 = mouse / scroller

 cmp.b #$93,d0             ;qui if close window
 beq Wt_done

 move.l a4,a0              ;show message content
 addq.l #1,d7
 TLhexasc16 d7,#8,a0       ;message count
 move.b #' ',(a0)+
 TLhexasc16 d5,#8,a0       ;IAddress
 move.b #' ',(a0)+
 TLhexasc16 d0,#4,a0       ;ascii / TL code
 move.b #' ',(a0)+
 TLhexasc16 d4,#8,a0       ;class (IDCMP)
 move.b #' ',(a0)+
 TLhexasc16 d3,#4,a0       ;qualifier
 move.b #' ',(a0)+

 TLhexasc16 d1,#4,a0       ;show mouse/scroller posn
 move.b #' ',(a0)+
 TLhexasc16 d2,#4,a0
 move.b #' ',(a0)+

 clr.b (a0)                ;show data
 TLtext #0,#120

 bra Wait                  ;go wait for next message

Wt_done:
 rts
```

Forward to program 63

1.79 p63 Demonstrate Program 63

Program 63

This is the second TLscroller demo. TLellipse fills a rastport with ellipses, and TLscroller lets you view it by moving the scrollers. You can resize the window as you go.

Run Program 63 -> TLscroller II

Source program 63
-> Teaching/63.asm

Forward to program 64

1.80 63asm

```
               Forward to program 64
               * 63.asm   Scrollers II, View Rastport    Version 0.01    8.6.99


 include 'Front.i'


port: ds.l 1               ;area to hold rastport
bmap: ds.l 1               ;area to hold bitmap

rwid: dc.l 1024            ;width of rastport
rhgt: dc.l 1024            ;height of rastport

rpxp: dc.l 0               ;region of rastport to be viewed
rpyp: dc.l 0
rpwd: dc.l 1024
rpht: dc.l 1024

wdxp: ds.l 1               ;region of window where rastport is shown
wdyp: ds.l 1
wdwd: ds.l 1
wdht: ds.l 1


* text strings
strings: dc.b 0
st_1: dc.b 'View a rastport by using window scrollers',0 ;1
 dc.b 'I have made a rastport, & filled it with ellipses.',0 ;2
 dc.b 'After you click "OK" on this requester, use the window scrollers',0
 dc.b 'to view the rastport. Also, resize, zoom & move the window.',0 ;4
 dc.b 'Finally, click the window to quit.',0 ;5
 dc.b 'Error: out of public memory',0 ;6

 ds.w 0


* control program execution
Program:
 TLwindow #-1
 beq Pr_quit

 TLwindow #0,#0,#0,#320,#100,#640,xxp_Height(a4),#-1,#st_1
 beq Pr_quit

 bsr Make                  ;draw ellipses on rastport
 beq Pr_quit               ;go if out of mem
 TLreqinfo #2,#4,#0        ;instructions

Pr_resz:                   ;here if window resized
 TLwupdate
```

```
 move.l xxp_AcWind(a4),a5  ;set sliders to top left
 move.l xxp_scrl(a5),a3
 moveq #0,d0
 move.w xxp_PWidth(a5),d0
 clr.l xxp_hztp(a3)
 move.l d0,xxp_hzvs(a3)
 move.l #1024,xxp_hztt(a3)
 move.w xxp_PHeight(a5),d0
 clr.l xxp_vttp(a3)
 move.l d0,xxp_vtvs(a3)
 move.l #1024,xxp_vttt(a3)
 TLwscroll set

Pr_blit:                   ;here for next blit
 move.l xxp_gfxb(a4),a6
 move.l port,a0
 move.l xxp_hztp(a3),d0
 move.l xxp_vttp(a3),d1
 move.l xxp_AcWind(a4),a5
 move.l xxp_WPort(a5),a1
 moveq #0,d2
 move.w xxp_LeftEdge(a5),d2
 moveq #0,d3
 move.w xxp_TopEdge(a5),d3
 moveq #0,d4
 move.w xxp_PWidth(a5),d4
 moveq #0,d5
 move.w xxp_PHeight(a5),d5
 move.w #$C0,d6
 TLwcheck                  ;go if window resized
 bne Pr_resz
 jsr _LVOClipBlit(a6)

Pr_wait:                   ;wait for response
 TLkeyboard
 cmp.b #$98,d0
 beq Pr_blit               ;update window if scroller
 TLwcheck
 bne Pr_resz               ;go if resized
 cmp.b #$93,d0
 beq.s Pr_quit             ;quit if close window
 cmp.b #$80,d0
 bne Pr_wait               ;continue until clicked

Pr_quit:                   ;quit, with error report if any
 TLerror
 rts


* make the rastport
Make:
 TLbusy
 clr.l xxp_errn(a4)        ;no error so far

 TLpublic #rp_SIZEOF       ;memory for rastport
 move.l d0,port
```

```
 beq Mk_bad1

 move.l xxp_gfxb(a4),a6    ;init rastport
 move.l d0,a1
 jsr _LVOInitRastPort(a6)

 TLpublic #bm_SIZEOF       ;memory for bitmap struct
 move.l d0,bmap
 beq Mk_bad1

 move.l d0,a0              ;init bitmap (only 1 plane)
 moveq #1,d0
 move.l rwid,d1
 move.l rhgt,d2
 jsr _LVOInitBitMap(a6)

 move.l bmap,a0            ;point rastport to bitmap
 move.l port,a1
 move.l a0,rp_BitMap(a1)

 TLchip #128*1024          ;memory for bitplane  (bytes per row 1024/8=128)
 move.l d0,bm_Planes(a0)
 beq Mk_bad2

 move.l d0,a1
 move.l #128*1024,d0
 moveq #1,d1
 jsr _LVOBltClear(a6)

 moveq #64,d0              ;draw ellipses:   D0 = x centre
 moveq #8,d2               ;d2 = x radius
 bset #31,d0               ;flag solid
 move.l port,a0            ;a0 = rastport
 moveq #0,d4               ;use all of rastport
 moveq #0,d5
 move.l #1024,d6
 move.l #1024,d7
Mk_col:                    ;for each column
 moveq #32,d1              ;d1 = y centre
 bset #31,d1               ;flag use rastport
 moveq #2,d3               ;d3 = y radius
Mk_elps:
 TLellipse d0,d1,d2,d3,d5,d5,d6,d7,a0 ;draw ellipse
Mk_fwd:
 addq.w #2,d3              ;bump y radius
 add.w #64,d1              ;bump y centre
 cmp.w #1024,d1
 bcs Mk_elps               ;until column done
 addq.w #8,d2              ;bump d radius
 add.w #128,d0             ;bump x centre
 cmp.w #1024,d0
 bcs Mk_col                ;until all columns done

 bra.s Mk_done

Mk_bad1:                   ;bad 1 - out of public mem
 move.w #1,xxp_errn+2(a4)
```

```
 bra.s Mk_done

Mk_bad2:                    ;bad 2 - out of chip mem
 move.w #2,xxp_errn+2(a4)

Mk_done:
 TLunbusy
 tst.l xxp_errn(a4)
 eori.w #-1,CCR
 rts


                Forward to program 64
```

1.81 p64 Demonstrate Program 64

```
                Program 64

   Run Program 64    -> refreshing

                Source program 64
                 -> Teaching/64.asm

This program uses TLellipse to draw an ellipse so big it doesn't fit
entirely in the window. It is interesting to continually resize the window,
watching the indomitable efforts of TLellipse to finish the job.

This program is really a demonstration of TLwupdate & TLwcheck, which
allow you to do easy (but unsophisticated) refreshing.


                Forward to program 65
```

1.82 64asm

```
                Forward to program 65
                * 64.asm     TLellipse II    version 0.01    8.6.99


 include 'Front.i'


strings: dc.b 0
st_1: dc.b 'TLEllipse demo',0 ;1
 dc.b 'Error: out of chip memory',0 ;2
 dc.b 'I''m an ellipse!!',0 ;3
 dc.b 'This is version II of TLEllipse (Version I was Teaching/58.asm)',0 ;4
 dc.b 'Version I drew to a rastport, whereas Version II draws direct',0 ;5
 dc.b 'to a window. As this version draws, try resizing the window',0 ;6
```

```
 dc.b 'to see it stop and retry until such times as you stop resizing',0
 dc.b 'the window. Then, inspect the program to see how this works -',0 ;8
 dc.b 'TLellipse fails (returns EQ) if the window is resized before it',0 ;9
 dc.b 'finishes. (\9 of TLellipse is null, so uses the popped window)',0

 ds.w 0


* test TLellipse
Program:
 TLwindow #-1              ;set up
 TLreqinfo #4,#7           ;tell user what is happening
 TLwindow #0,#0,#0,#320,#100,#640,#200,#0,#st_1 ;open window 0
 bne.s Pr_cont             ;go if ok
 TLbad #2                  ;report if can't open window
 rts

Pr_cont:
 TLreqcls                  ;n.b. this includes a call to TLWupdate.
 move.l xxp_AcWind(a4),a5  ;a5 = window's xxp_wsuw
 move.l #320,d0
 moveq #100,d1             ;centre 320,100
 move.l d0,d2
 move.l d1,d3              ;radii 320,100
 moveq #0,d4               ;trim at current limit of window display area
 moveq #0,d5
 moveq #0,d6
 moveq #0,d7
 move.w xxp_PWidth(a5),d6
 subq.w #1,d6
 move.w xxp_PHeight(a5),d7
 subq.w #1,d7

 move.b #2,xxp_FrontPen(a5)
 TLellipse #320,#100,#320,#100,d4,d5,d6,d7,,solid ;draw ellipse

 move.w #$0100,xxp_FrontPen(a5)
 TLstring #3,#260,#85      ;write message

Pr_wait:
 TLwcheck                  ;re-draw if window resized
 bne Pr_cont
 TLwslof

 TLkeyboard                ;wait for response & quit
 rts


               Forward to program 65
```

1.83 p65 Demonstrate Program 65

```
               Program 65
```

```
This program demonstrats TLtabs.

   Run Program 65    -> TLtabs

                Source program 65
                 -> Teaching/65.asm

TLtabs allows you to easily put a set of thumbtabs on a window, & bring
other thumbtabs to the front if you click them.

If the window is too small to fit the tab cards in, tandem.library will
trim them as required.

Program 65 is large, as it does a lot of things. But if you study it
carefully you will see it is conceptually simple.


                Forward to program 66
```

1.84 65asm

```
                Sorry!

Program 65.asm is large, and it would be wasteful to duplicate it here.

You'll find it in:


  Tandem/Teaching/65.asm


                Forward to program 66
```

1.85 p66 Demonstrate Program 66

```
                Program 66

   Run Program 66    -> TLslider, TLslimon

                Source program 66
                 -> Teaching/66.asm

This program demonstrates TLslider & TLslimon, which allow you to easily
put sliders on a window, & monitor them.

If the sliders don't fit on the window, tandem.library clips them as
required.

Click the button below, or the one at the end, after reading the text below
```

```
                Forward to Program 67
                Incidentally, tandem.library does not use messages to monitor  ↩
                   things like
sliders, tabcards & buttons. Instead, you call TLkeyboard, and then call
a monitor routine for each type of thing, to see if it wants the TLkeyboard
output. This is less sophisticated than using Amiga's gadgets & objects, but
much easier to understand & program.

After you get lots of experience with tandem.library you will be ready
to graduate to object oriented programming. But really the tandem.library
GUI allows you to do just about anything you can do with BOOPSI objects
anyway, well, er, at least within the context of a single task.

In future I will bring out ARexx and messaging TL routines to add to
tandem.library's capabilities.


                Forward to program 67
```

1.86 66asm

```
                Forward to program 67
                * 66.asm     TLslimon      0.01    8.6.99)


 include 'Front.i'


; *** Important!!! ***
;
; If assembling this program for demonstration/77, make the
; address of  Logo.iff  to be  /Logo.iff  in string 12


; See the remarks on 65.asm, which are also pertinent to this program


vert: ds.l 1                ;xxp_tops cache for vert,horz sliders
horz: ds.l 1

logo: ds.l 1                ;pointer to bitmap where Logo.iff stored


strings: dc.b 0
st_1: dc.b  'TLSlimon demonstration',0 ;1
 dc.b 'Out of chip ram',0 ;2
 dc.b 'totl 298',0 ;3
 dc.b 'strs 64',0 ;4
 dc.b 'tops',0 ;5
 dc.b 'Horz',0 ;6
 dc.b 'Vert',0 ;7
```

```
 dc.b 'totl 149',0 ;8
 dc.b 'strs 48',0 ;9
 dc.b 'Error: can''t find/load Tandem/Logo.iff',0 ;10
 dc.b 'Use the sliders to move the logo around...',0 ;11
 dc.b 'Logo.iff',0 ;12

 ds.w 0


Program:
 TLwindow #0,#0,#0,#200,#100,#640,#256,#0,#st_1
 beq Pr_bad1

 TLstrbuf #12
 move.l a4,a1
 add.w #20,a1
 TLgetilbm #2,a1
 beq Pr_bad2
 move.l a0,logo

 clr.l vert                ;init vert,horz
 clr.l horz

Pr_draw:
 bsr Logo                  ;update the window, display the logo

 move.l xxp_AcWind(a4),a5
 move.w #$0200,xxp_FrontPen(a5)
 TLstring #11,#20,#6
 subq.b #1,xxp_FrontPen(a5)

 TLreqbev #20,#20,#320,#161     ;outline bev: scrolling area 22-319 X 21-169

 TLreqarea #320,#170,#18,#10,#3 ;draw icon at bottom right
 TLreqbev #320,#170,#20,#11
 TLpict #11,#326,#171

 TLstring #6,#400,#20      ;print horz data: totl, strs, tops stub
 TLstring #3,#448,#20
 TLstring #4,#448,#28
 TLstring #5,#448,#36

 TLstring #7,#400,#52      ;print vert data:  totl, strs, tops stub
 TLstring #8,#448,#52
 TLstring #9,#448,#60
 TLstring #5,#448,#68

 move.l #20,xxp_slix(a4)   ;draw horz slider
 move.l #170,xxp_sliy(a4)
 move.l #300,xxp_sliw(a4)
 move.l #11,xxp_slih(a4)
 move.l #298,xxp_totl(a4)
 move.l #64,xxp_strs(a4)
 move.l horz,xxp_tops(a4)
 TLslider xxp_AcWind(a4),a5

 move.l #320,xxp_slix(a4)  ;draw vert slider
```

```
 move.l #20,xxp_sliy(a4)
 move.l #20,xxp_sliw(a4)
 move.l #150,xxp_slih(a4)
 move.l #149,xxp_totl(a4)
 move.l #48,xxp_strs(a4)
 move.l vert,xxp_tops(a4)
 clr.l xxp_hook(a4)
 TLslider xxp_AcWind(a4),a5

 bsr Htops                ;print horz tops value
 bsr Vtops                ;print vert tops value

Pr_wait:                  ;* wait for keyboard input
 TLwcheck
 bne Pr_draw              ;go redraw all if window resized

 TLkeyboard
 cmp.b #$93,d0            ;quit if close window
 beq Pr_quit
 cmp.b #$1B,d0            ;quit if Esc
 beq Pr_quit

 cmp.b #$80,d0            ;ignore keyboard input unless lmb click
 bne Pr_wait

 move.l horz,xxp_tops(a4)  ;monitor horz slider
 move.l #20,xxp_slix(a4)
 move.l #170,xxp_sliy(a4)
 move.l #300,xxp_sliw(a4)
 move.l #11,xxp_slih(a4)
 move.l #298,xxp_totl(a4)
 move.l #64,xxp_strs(a4)
 move.l #Hhook,xxp_hook(a4)
 TLslimon d1,d2,d3
 beq.s Pr_vert            ;go if horz slider inactive
 move.l xxp_tops(a4),horz  ;else update horz
 bra Pr_wait              ;& get next keyboard input

Pr_vert:
 move.l vert,xxp_tops(a4)  ;monitor vert slider
 move.l #320,xxp_slix(a4)
 move.l #20,xxp_sliy(a4)
 move.l #20,xxp_sliw(a4)
 move.l #150,xxp_slih(a4)
 move.l #149,xxp_totl(a4)
 move.l #48,xxp_strs(a4)
 move.l #Vhook,xxp_hook(a4)
 TLslimon d1,d2,d3
 beq Pr_wait              ;go if vert slider inactive
 move.l xxp_tops(a4),vert  ;else update vert
 bra Pr_wait              ;& get keyboard input

Pr_bad1:                  ;report out of mem if bad
 TLbad #2
 bra.s Pr_quit

Pr_bad2:
```

```
 TLbad #10

Pr_quit:                    ;exit from Program
 tst.l logo
 beq.s Pr_exit
 TLfreebmap logo            ;free logo memory
Pr_exit:
 rts


* hook for horizontal slider
Hhook:
 bsr Htops
 bsr Hlogo
 rts


* show horz tops
Htops:
 move.l #'    ',(a4)
 TLhexasc xxp_tops(a4),a4
 clr.b 3(a4)
 TLtrim #488,#36
 rts


* hook for vertical slider
Vhook:
 bsr Vtops
 bsr Vlogo
 rts


* move the logo horizontally
Hlogo:
 move.l xxp_gfxb(a4),a6
 move.l xxp_AcWind(a4),a5

 move.l xxp_WPort(a5),a1
 moveq #3,d0
 jsr _LVOSetBPen(a6)

 move.l xxp_WPort(a5),a1
 moveq #22,d2
 moveq #21,d3
 move.l #319,d4
 cmp.w xxp_PWidth(a5),d4
 ble.s Hl_vtrm
 move.w xxp_PWidth(a5),d4
Hl_vtrm:
 move.l #169,d5
 cmp.w xxp_PHeight(a5),d5
 ble.s Hl_redi
 move.w xxp_PHeight(a5),d5
Hl_redi:
 move.l horz,d0
 sub.l xxp_tops(a4),d0
```

```
 move.l xxp_tops(a4),horz
 moveq #0,d1
 add.w xxp_LeftEdge(a5),d2
 add.w xxp_TopEdge(a5),d3
 add.w xxp_LeftEdge(a5),d4
 add.w xxp_TopEdge(a5),d5
 TLwcheck
 bne.s Hl_quit
 jsr _LVOScrollRaster(a6)
Hl_quit:
 rts


* move the logo vetically
Vlogo:
 move.l xxp_gfxb(a4),a6
 move.l xxp_AcWind(a4),a5

 move.l xxp_WPort(a5),a1
 moveq #3,d0
 jsr _LVOSetBPen(a6)

 move.l xxp_WPort(a5),a1
  moveq #22,d2
 moveq #21,d3
 move.l #319,d4
 cmp.w xxp_PWidth(a5),d4
 ble.s Vl_vtrm
 move.w xxp_PWidth(a5),d4
Vl_vtrm:
 move.l #169,d5
 cmp.w xxp_PHeight(a5),d5
 ble.s Vl_redi
 move.w xxp_PHeight(a5),d5
Vl_redi:
 moveq #0,d0
 move.l vert,d1
 sub.l xxp_tops(a4),d1
 move.l xxp_tops(a4),vert
 add.w xxp_LeftEdge(a5),d2
 add.w xxp_TopEdge(a5),d3
 add.w xxp_LeftEdge(a5),d4
 add.w xxp_TopEdge(a5),d5
 TLwcheck
 bne.s Vl_quit
 jsr _LVOScrollRaster(a6)
Vl_quit:
 rts


* show vert tops
Vtops:
 move.l #'    ',(a4)
 TLhexasc xxp_tops(a4),a4
 clr.b 3(a4)
 TLtrim #488,#68
 rts
```

```
* draw logo on the window (calls Wupdate)
; caution: the minumum size of the window must allow logo to fit at topleft

Logo:
 TLwupdate
 TLreqarea #22,#21,#298,#149,#3 ;fill scrolling area with pen 3
 move.l xxp_AcWind(a4),a5  ;a5 = currently active window

 move.w xxp_PWidth(a5),d0  ;move horz if it doesn't fit on the window
 sub.w #22,d0
 sub.w #64,d0
 cmp.w horz+2,d0
 bcc.s Lo_vert
 move.w d0,horz+2

Lo_vert:                   ;move vert if it doesn't fit on the window
 move.w xxp_PHeight(a5),d0
 sub.w #21,d0
 sub.w #48,d0
 cmp.w vert+2,d0
 bcc.s Lo_redi
 move.w d0,vert+2

Lo_redi:                   ;ready the blit
 move.l logo,a0
 move.l xxp_WPort(a5),a1
 move.l xxp_gfxb(a4),a6
 moveq #0,d0
 moveq #0,d1
 moveq #22,d2
 add.w horz+2,d2
 add.w xxp_LeftEdge(a5),d2
 moveq #21,d3
 add.w vert+2,d3
 add.w xxp_TopEdge(a5),d3
 moveq #64,d4
 moveq #48,d5
 move.w #$C0,d6

 TLwcheck                  ;recyle if window resized
 bne Logo

 jsr _LVOBltBitMapRastPort(a6) ;go the blit
 rts


                Forward to program 67
```

1.87 p67 Demonstrate Program 67

```
                Program 67
```

```
This program demonstrates TLbutmon which allows you easily draw and monitor
sets of buttons.

   Run Program 67    -> TLbutmon

                Source program 67
                 -> Teaching/67.asm



                Forward to program 68
```

1.88 67asm

```
                Forward to program 68
                * 67.asm     TLbutmon        0.00        8.6.99)


 include 'Front.i'


;box size & contents data
xwid: ds.l 1                ;width of 2nd box set
radi: ds.l 1                ;which button of 1st box set selected
prev: ds.w 1                ;previous box clicked (ASCII of set,box)


strings: dc.b 0
st_1: dc.b  'TLbutmon demonstration',0 ;1
 dc.b 'Error: out of chip ram',0 ;2
 dc.b 'A box\Another\Yet another\Yes! Another\The Last non-blank box',0 ;3
 dc.b 'Click me!',0 ;4
 dc.b 'No - click me!',0 ;5
 dc.b 'Or, click me',0 ;6
 dc.b 'Click any box from either set of buttons',0 ;7
 dc.b '(Click the close window gadget to quit)',0 ;8
 dc.b 'Set 1 - radio buttons',0 ;9
 dc.b 'Set 2 - some boxes with text',0 ;10
 dc.b 'Your last click was set '
st_11a: dc.b ' , box '
st_11b: dc.b ' ',0 ;11

 ds.w 0


Program:
 TLwindow #-1
 beq Pr_bad
 TLwindow #0,#0,#0,#200,#100,xxp_Width(a4),xxp_Height(a4),#0,#st_1
 beq Pr_bad

 clr.w prev                ;no input yet
```

```
Pr_draw:                    ;draw the window contents
 TLreqcls                   ;clear window, update window size

 move.l xxp_AcWind(a4),a5  ;print instructions
 move.w #$0200,xxp_FrontPen(a5)
 TLstring #7,#10,#5
 TLstring #8,#10,#13
 subq.b #1,xxp_FrontPen(a5)

 TLstring #9,#10,#43       ;print 1st set
 TLstring #4,#30,#54
 TLstring #5,#30,#69
 TLstring #5,#30,#84
 move.l #10,xxp_butx(a4)   ;posn = 10,53
 move.l #53,xxp_buty(a4)
 move.l #12,xxp_butw(a4)   ;size = 12 X 10
 move.l #10,xxp_buth(a4)
 move.l #15,xxp_btdy(a4)   ;dy = 15   (no dx since butk=1)
 move.l #1,xxp_butk(a4)    ;columns = 1
 move.l #3,xxp_butl(a4)    ;rows = 3
 TLbutprt                  ;print buttons

 TLstring #10,#10,#106     ;print 2nd set
 move.l #116,xxp_buty(a4)
 TLstra0 #3                ;point to button contents
 TLbutstr a0               ;set butx,buty for them
 move.l xxp_butw(a4),xwid  ;cache button width
 move.l xxp_butw(a4),xxp_btdx(a4) ;buttons touch horz & vert
 move.l xxp_buth(a4),xxp_btdy(a4)
 move.l #3,xxp_butk(a4)    ;columns = 3
 move.l #2,xxp_butl(a4)    ;rows = 2
 TLbuttxt a0               ;print text in them
 TLbutprt                  ;print buttons

Pr_chek:
 move.l radi,d2            ;print checkmark
 mulu #15,d2
 add.w #54,d2              ;ypos = radi * btdy + 54
 TLpict #10,#12,d2         ;pict 10 at 12,d2

Pr_prev:
 tst.w prev                ;go if not input yet
 beq.s Pr_wait
 move.b prev,st_11a        ;else, show last input
 move.b prev+1,st_11b
 TLstring #11,#10,#150

Pr_wait:                   ; wait for user response
 TLwcheck
 bne Pr_draw               ;redraw if window resized
 TLkeyboard                ;get input
 cmp.b #$93,d0
 beq Pr_quit               ;quit if close window
 cmp.b #$80,d0
 bne Pr_wait               ;else reject unless lmb

 move.l #10,xxp_butx(a4)   ;set up to monitor 1st button set
```

```
 move.l #53,xxp_buty(a4)
 move.l #12,xxp_butw(a4)
 move.l #10,xxp_buth(a4)
 move.l #15,xxp_btdy(a4)
 move.l #1,xxp_butk(a4)
 move.l #3,xxp_butl(a4)

 TLbutmon d1,d2            ;was 1st set clicked?
 beq.s Pr_set2             ;no, to second set
 move.b #'1',prev
 move.b #'0',prev+1        ;record lst keyboard input
 add.b d0,prev+1

 move.l radi,d1            ;d1 = old checkmark
 subq.l #1,d0              ;d0 = new chweckmark, 0-2
 move.l d0,radi            ;which save in radi
 mulu #15,d1
 add.w #54,d1
 TLreqarea #12,d1,#8,#8,#0 ;clear old checkmark
 bra Pr_chek               ;print new checkmark, to next input

Pr_set2:                   ;set up to monitor 2nd button set
 move.l #116,xxp_buty(a4)
 move.l xwid,xxp_butw(a4)
 move.l xxp_butw(a4),xxp_btdx(a4)
 move.l #10,xxp_buth(a4)
 move.l #10,xxp_btdy(a4)
 move.l #3,xxp_butk(a4)
 move.l #2,xxp_butl(a4)

 TLbutmon d1,d2            ;monitor 2nd button set
 beq Pr_wait               ;go if neither set
 move.b #'2',prev          ;else record which clicked
 move.b #'0',prev+1
 add.b d0,prev+1
 bra Pr_prev               ;& await next input

Pr_bad:                    ;report out of mem if bad
 TLbad #2

Pr_quit:                   ;exit from Program
 rts
```

Forward to program 68

1.89 p68 Demonstrate Program 68

Program 68

Run Program 68 -> TLprefs

Source program 68
-> Teaching/68.asm

```
This program demonstrates TLprefs, which allows the user to set
tandem.library preferences, and optionally to specify the colour palette.


                Forward to program 69
```

1.90 68asm

```
                Forward to program 69
                * 68.asm  TLprefs     version 0.01     8.6.99


 include 'Front.i'


; TLprefs puts up a requester with lots of built-in help &c to allow the
; user to set prefernces for the tandem.lubrary GUI. You can choose whether
; to also allow the user to change the colour palette. Note that Multiline
; has "GUI Preferences" in its Project menu, which calls TLprefs.


strings: dc.b 0
 dc.b 'Error: out of memory',0 ;1
 ds.w 0


* demonstrate TLprefs
Program:
 TLwindow #-1
 beq Pr_bad

 TLprefs                    ;or "TLprefs color" to allow palette select
 bra.s Pr_quit

Pr_bad:                     ;here if out of mem
 TLbad #1

Pr_quit:
 rts


                Forward to program 69
```

1.91 p69 Demonstrate Program 69

```
                Program 69

This program is not very thrilling.
```

```
   Run Program 69    -> Custom requesters

                Source program 69
                 -> Teaching/69.asm
```

Program 69.asm is really meant to have its source examined. It is the skeleton of a custom requester, on which the programmer can put wonderfully creative decorations & active areas. Amiga requesters are something of an art form. As a programmer, note carefully all the requesters you see, and consider their aesthetics and intuitiveness. Then, do even better.

On your custom requester, you can put sliders, drop down menus, strings, buttons and tabcards, as well as drawing boxes, areas, and adding pictures. All these things can also be done on windows just as on requesters.

tandem.library's set of requesters is very suitable for all the basic things you do with requesters, & don't forget you can modify them with hooks to jazz them up a little.

```
                Forward to program 70
```

1.92 69asm

```
                Forward to program 70
                * 69.asm  Custom Requester     version 0.00     1.9.97


 include 'Front.i'

        ************        Important !!!        **************
  *
  *     Tandem/Teaching/69.asm has a version of this program with
  *     very detailed comments - be sure to look at it. I have
  *     chopped them out here, as it would be wasteful to
  *     duplicate them.
  *




width: dc.l 100  ; dummy width & height for the
height: dc.l 50  ; requester below


strings: dc.b 0
 dc.b 'Error: out of memory',0 ;1
 dc.b 'Click me!',0 ;2
 ds.w 0


* here is the "skeleton" of a custom requester.
```

```
Program:
 TLwindow #-1
 beq Pr_bad

 bsr Custom
 bra.s Pr_quit

Pr_bad:                      ;here if out of mem
 TLbad #1

Pr_quit:
 rts


* make a custom requester (with nothing on it)
Custom:
 movem.l d0-d7/a0-a6,-(a7) ;save all
 sub.w #xxp_WPort+4,a7     ;create dummy part xxp_wsuw

 move.l a7,a5              ;a5 points to dummy part xxp_wsuw
 TLreqredi a5              ;set pop window, default values to xxp_prfp
 beq .bad                  ;go if TLReqredi fails - unlikely

 move.l xxp_pref(a4),a0    ;prefs to prfp  } Optional:
 move.l xxp_yinf(a0),xxp_prfp(a4)          } Use prefs for TLReqinfo-type
 move.l xxp_yinf+4(a0),xxp_prfp+4(a4)      } requesters (but must set prfp)


                           ;calculate requester size: set width,height
                           ;(adjust prfp & font if necessary to make it fit)


 TLreqchek width,height    ;check req size & position
 beq .bad                  ;go if won't fit

 tst.w xxp_ReqNull(a4)     ;quit ok if ReqNull=0
 beq .wrap

 TLreqon a5                ;open requester window
 beq .bad                  ;go if can't

 move.b xxp_prfp+2(a4),xxp_FrontPen(a5)  ;} Set pens to TLReqinfo
 move.b xxp_prfp(a4),xxp_BackPen(a5)     ;} prefernces (see above)

                           ;attach help

.draw:                     ;draw requester

 TLstring #2,#5,#6

.wait:                     ;wait for user response
 TLwfront
 TLkeyboard

 cmp.b #$80,d0             ;process user response, branching to
 bne .wait                 ;  draw - if requester to be re-drawn
                           ;  wait - if requester not to be re-drawn
```

```
                          ;  clos - if user requested OK/Cancel &c


.clos:
 TLreqoff                 ;close requester window
 moveq #-1,d0             ;signal ok
 bra.s .wrap              ;return ok

.bad:
 moveq #0,d0              ;too big/can't open window

.wrap:
 move.w #-1,xxp_ReqNull(a4) ;leave ReqNull<>0
 TLwslof
 tst.l d0                 ;EQ if bad
 add.w #xxp_WPort+4,a7
 movem.l (a7)+,d0-d7/a0-a6
 rts


               Forward to program 70
```

1.93 p70 Demonstrate Program 70

```
               Program 70

This program demonstrate built-in help, and how you can attach an AmigaGuide
to the help button also, which comes up with a spcified node showing.

   Run Program 70    -> built-in help

               Source program 70
                -> Teaching/70.asm

The idea is, you put the lines of help in "strings", & then at each part
of your program, you poke the number of the first string & the number of
strings into an A4 offset. From then on, TLkeyboard does the rest. It is
scarcely less simple to also add a Guide node to be looked up.

(I will add to these capabilities to have a table of active areas in a
 window for the <Help> button, in a later edition of tandem.library).


               Forward to program 71
```

1.94 70asm

```
               Forward to program 71
               * 70.asm  Attach Guide     version 0.01     8.6.99
```

```
 include 'Front.i'


; The Help requester that comes up may also contain a "Guide" button that
; invokes an AmigaGuide to supplement the help you give. To do that:
;   1. set xxp_guid(a4) to point to the path of the guide
;      (if your program CD's to its progdir:, that can be the guide's
;       simple filename if it is in the Progdir:)
;   2. set xxp_node(a4) to point to the nodename (or 0 for contents)
;   3. set bit 7 of xxp_Help+2(a4)
;
; Thus, as well as setting xxp_Help with context sensitive help from time
; to time, you can at the same time set xxp_node(a4) to point to the
; relevent node name.


guid: dc.b 'Tandem.guide',0 ;path of AmigaGuide
node: dc.b 'help',0         ;node of Tandem.guide
 ds.w 0


strings: dc.b 0
 dc.b 'Error: out of memory',0 ;1
 dc.b 'This is just a dummy requester...',0 ;2
 dc.b 'The idea, is for you to press the <Help> key.',0 ;3
 dc.b 'When you do, Help will have 2 buttons:',0 ;4
 dc.b ' ',0 ;5
 dc.b '1. Guide - which should bring up the "help" node of Tandem.guide',0
 dc.b '2. OK - which should close the Help requester',0 ;7
 dc.b 'Data about Tandem/Teaching/70.asm',0 ;8
 dc.b ' ',0 ;9
 dc.b 'The program you are running is a demonstration of',0 ;10
 dc.b 'tandem.library''s ability to display online help. To find out',0 ;11
 dc.b 'more about how to attach context-sensitive online help, press',0 ;12
 dc.b 'the "Guide" key below, which should display the "help" node',0 ;13
 dc.b 'of Tandem.guide',0 ;14

 ds.w 0


* demonstrate context-sensitive online help
Program:
 TLwindow #-1
 beq Pr_bad

 move.w #8,xxp_Help(a4)     ;create context sensitive online help
 move.w #7,xxp_Help+2(a4)

 move.l #guid,xxp_guid(a4)  ;cause Help to offer Tandem.guide
 move.l #node,xxp_node(a4)
 bset #7,xxp_Help+2(a4)

 TLreqinfo #2,#6            ;put up requester with online help
 bra.s Pr_quit
```

```
Pr_bad:                     ;here if out of mem
 TLbad #1

Pr_quit:
 rts
```

Forward to program 71

1.95 p71 Demonstrate Program 71

Program 71

Run Program 71 -> Typing a password

Source program 71
-> Teaching/71.asm

This puts up a tiny requester to type in a password. You can position the requester (as you can with all requesters) to come up where you want it relative to its calling window. having a little area suddenly appear, & disappear when to password is entered, adds to the warm feeling of security, I hope. (Of course, as everyone knows, there's no such thing as security).

Forward to program 72

1.96 71asm

Forward to program 72

```
* 71.asm  TLpassword     version 0.01     8.6.99


 include 'Front.i'


strings: dc.b 0
 dc.b 'Cancelled',0 ;1
 dc.b 'Error: out of memory',0 ;2
 ds.w 0


; TLpassword puts up a little requester to get a password. You can adjust
; its position (with xxp_ReqTop & xxp_ReqLeft) to place itself on a
; window where it's required. This program does not encrypt the input
; internally, so it is a fairly naive sort of routine.


* demonstrate TLPassword
```

```
Program:
 TLwindow #-1               ;set up
 beq Pr_quit                ;bad if can't

 TLpassword #6              ;get password (6 characters maximum)
 beq.s Pr_bad2              ;go report if error
 cmp.b #$1B,d0
 bne.s Pr_rept              ;go report password if ok
 TLstrbuf #1
 bra.s Pr_rept              ;go report if cancel

Pr_bad1:                    ;error 1: out of mem (report in monitor)
 TLbad #2
 bra.s Pr_quit

Pr_bad2:                    ;error 2: TLpassword failed - error to buff
 TLerror

Pr_rept:                    ;report password / cancel / error
 TLreqchoose

Pr_quit:
 rts
```

Forward to program 72

1.97 p72 Demonstrate Program 72

Program 72

This program demonstrates TLdropdown

Run Program 72 -> Drop down menus

Source program 72
-> Teaching/72.asm

TLdropdown draws & monitors dropdown menus. Try resizing the window little to see how TLdropdown clips the menu to fit, & repositions the dropdown if required, at least up to a point. Unlike most other requesters, TLdropdown is not font sensitive. Actually, TLdropdown is not a requester, but an active area on a window.

1.98 72asm

```
* 72.asm     TLdropdown    version 0.00    8.9.99


 include 'Front.i'
```

```
; The program draws & monitors a drop down menu. If you wanted it to be
; a cycle gadet, you'd put 'cycle' for \8, like this:
;
; To draw:
;
;     TLdropdown draw,#4,#6,mood,#20,#25,,cycle  ;draw
;
; To monitor;
;
;     TLdropdown monitor,#4,#6,mood,#20,#25,,cycle   ;monitor


mood: ds.l 1              ;holds the operative mood (1+)


strings: dc.b 0
st_1: dc.b 'TLdropdown demo',0 ;1
 dc.b 'Error: out of chip memory',0 ;2
 dc.b 'Choose a mood, or click the window close gadget',0 ;3
 dc.b 'Happiness is',0 ;4
 dc.b 'Sadness is',0 ;5
 dc.b 'Boredom is',0 ;6
 dc.b 'Fascination is',0 ;7
 dc.b 'Fear is',0 ;8
 dc.b 'Relief is',0 ;9
 dc.b 'Someone gets JAVA working on Amiga',0 ;10
 dc.b 'People are buying PCs             ',0 ;11
 dc.b 'Downloading a newsgroup           ',0 ;12
 dc.b 'The Amiga workbench               ',0 ;13
 dc.b 'Programming a printer driver      ',0 ;14
 dc.b 'I didn''t crash                    ',0 ;15
 dc.b '(I should reappear when the drop down disappears)',0 ;16

 ds.w 0


* demonstrate TLdoprdown
Program:
 TLwindow #0,#0,#0,#200,#40,#640,#200,#0,#st_1
 beq Pr_bad

Pr_resize:                ;here if window resized
 TLreqcls                 ;clear window
 TLstring #3,#20,#10      ;print instructions
 TLstring #16,#20,#58     ;writing underneath
 move.l #1,mood           ;initialise mood

 TLdropdown draw,#4,#6,mood,#20,#25    ;draw with initial mood

Pr_report:                ;report mood so far
 moveq #9,d0
 add.l mood,d0
 TLstring d0,#154,#26

Pr_wait:                  ;wait for user response
```

```
 TLwcheck                  ;go if window resized
 bne Pr_resize
 TLkeyboard                ;get input
 cmp.b #$93,d0
 beq Pr_quit               ;quit if close window
 cmp.b #$1B,d0
 beq Pr_quit               ;quit if Esc
 cmp.b #$80,d0
 bne Pr_wait               ;else ignore unless lmb

 move.l d1,xxp_kybd+4(a4)
 move.l d2,xxp_kybd+8(a4)
 TLdropdown monitor,#4,#6,mood,#20,#25,,#5  ;monitor dropdown
 beq Pr_wait

 move.l d0,mood            ;set new mood
 bra Pr_report             ;report response, wait for next

Pr_bad:
 TLbad #2

Pr_quit:
 rts
```