Meta-commands used in this document:

italic text
‹*placeholder*›
\command
[‹*bar*›]{‹*foo*›}
[*bar*]{*foo*}
\ic_funca:n
function
FUNCTION

```
\textit{ italic  text}\par
\meta{placeholder}\par
   \cmd{command}\par
\oarg{bar}\marg{foo}\par
\oarg*{bar}\marg*{foo}\par
   \cmd{ic_funca:n}\par
\term{function}\par
\term*{Function}\par
```

1 Introduction

VARIABLES

\‹*scope*›_‹*module*›_‹*name*›_‹*type*›

Scope	Description
l	local
g	global

Type	Description
bool	boolean
box	box
cctab	category code table
clist	clist
dim	dimension
flag	flag
fp	floating point
fparray	floating point array
int	integer
intarry	integer array
msg	message
muskip	muskip
prop	property list
seq	sequence
skip	skip
str	string
tl	token list

FUNCTIONS

\‹*name*›‹*signature marker*›‹*signature*›

2 Basics

[2.1]

```
\ExplSyntaxOn  % <------
Some code.
```

```
Some code.
Some code.
\ExplSyntaxOff  % <------
Some code.
\group _begin:  % <------
Some code.
\group _end:  % <------
Some code.
```

[2.2]

[2.3] – Grouping

```
\ExplSyntaxOn
Text ~ text ~
\group_begin:
\color{red}
\large
text ~ text ~
\group_end:
text ~ text.
\ExplSyntaxOff
```

Text text text text text text.

[2.4] – Defining a function

```
\ExplSyntaxOn
\cs_new:Npn \ic_funca:n #1 { {\color{blue} \bfseries #1 } }

Text ~ \ic_funca:n { ~ This ~ is ~ blue ~ bold. } ~ Text.
\ExplSyntaxOff
```

Text **This is blue bold.** Text.

[2.5] – Defining a user command

```
\ExplSyntaxOn
\NewDocumentCommand { \mycmdb } { m } {
\cs_new:Npn \ic_funcb:n ##1 { {\color{blue} \bfseries ##1 } }
 \ic_funcb:n { #1 }
 }
\ExplSyntaxOff
Text \mycmdb{This is blue bold.} Text.
```

Text **This is blue bold.** Text.

[2.6] – Using a token list

```
\ExplSyntaxOn
\tl_new:N \l_myicc_tl
\NewDocumentCommand { \mycmdc } { m } {
\tl_set:Nn \l_myicc_tl { #1 }
\cs_new:Npn \ic_funcc:N ##1 { {\color{blue} \bfseries ##1 } }
 \ic_funcc:N \l_myicc_tl
 }
\ExplSyntaxOff
Text \mycmdc{This is blue bold.} Text.
```

Text **This is blue bold.** Text.

[2.7] – Appending to a token list

```
\ExplSyntaxOn
\tl_new:N \l_myicd_tl
\NewDocumentCommand { \mycmdd } { m } {
\tl_set:Nn \l_myicd_tl { #1 }
\tl_put_right:Nn \l_myicd_tl { $\leftarrow$ ! ! }
\tl_put_left:Nn \l_myicd_tl { ! ! $\rightarrow$ }
\cs_new:Npn \ic_funcd:N ##1 { {\color{blue} \bfseries ##1 } }
 \ic_funcd:N \l_myicd_tl
 }
\ExplSyntaxOff
Text \mycmdd{This is blue bold.} Text.
```

Text **!!→This is blue bold.←!!** Text.

[2.8] – Modifying a token list (\tl_replace)

```
\ExplSyntaxOn
\tl_new:N \l_myice_tl
\NewDocumentCommand { \mycmde } { m } {
\tl_set:Nn \l_myice_tl { #1 }
\tl_replace_all:Nnn \l_myice_tl { blue } { not ~ red }
\tl_replace_all:Nnn \l_myice_tl { bold } { italic }
\cs_new:Npn \ic_funce:N ##1 { {\color{blue} \bfseries ##1 } }
 \ic_funce:N \l_myice_tl
 }
\ExplSyntaxOff
Text \mycmde{This is blue bold.} Text.
```

Text **This is not red italic.** Text.

[2.9] – Modifying a token list (\regex_replace)

```
\ExplSyntaxOn
\tl_new:N \l_myicez_tl
\NewDocumentCommand { \mycmdez } { m } {
\tl_set:Nn \l_myicez_tl { #1 }
\regex_replace_all:nnN
{ (\c{color} \cB\{ [^\cE\}]* \cE\}) }
{ not \  }
\l_myicez_tl
\regex_replace_all:nnN
{ \c{textbf}(.+)(bold) }
{ is\ \c{textsc}\1 small\ caps }
\l_myicez_tl
\tl_use:N \l_myicez_tl
 }
\ExplSyntaxOff
Text \mycmdez{This is {\color{blue}blue} and \textbf{bold}.} Text.
```

Text This is not blue and is SMALL CAPS. Text.

[2.10] – Copying a control sequence

```
\ExplSyntaxOn
\cs_new:Npn \ic_funcf:N #1 { {\color{blue} \bfseries #1 } }
\cs_new_eq:NN \mybbcmd \ic_funcf:N
\ExplSyntaxOff
Text \mybbcmd{This is blue bold.} Text.
```

Text **This is blue bold.** Text.

[2.11] – Looping

```
\ExplSyntaxOn
\cs_set:Npn \ic_funcg:n #1 { \symbol{#1} }
\cs_set:Nn \ic_funcgb:  {
\int_step_function:nnnN { 97 } { 1 } { 122 } \ic_funcg:n
\
}
\cs_new_eq:NN \myloopcmd \ic_funcgb:
\ExplSyntaxOff
Text \myloopcmd Text.
```

Text abcdefghijklmnopqrstuvwxyz Text.

COMMENTARY:
(1) Open the expl3 environment.
(2) Define a 1-parameter function, *g*, that will print a glyph, given the glyph's slot number, in the current font using the **\symbol** command.
(3) Define a no-parameter function, *gb*, that will
(4) step through values 97 to 122 (inclusive) and pass each value to the *g* function.
(5) Add a space (replacing the one gobbled after the command in the user code).
(6) –
(7) Create a user-command, **\myloopcmd**, as a copy of the *gb* function.
(8) Close the expl3 environment.
(9) Use the user-command
RESULT: The letters a..z are printed, followed by a space.

[2.12] – Mapping function

```
\ExplSyntaxOn
\cs_set:Npn \ic_funch:n #1 { \fbox{#1}. }
\tl_new:N \l_myich_tl
\NewDocumentCommand { \mycmdh } { m } {
\tl_set:Nn \l_myich_tl { #1 }
\tl_map_function:NN \l_myich_tl \ic_funch:n
\par tl ~ = ~ >> \tl_use:N \l_myich_tl <<
}
\ExplSyntaxOff
Text \mycmdh{abc{de}fgh} Text.
```

Text a.b.c.de.f.g.h.
tl = »abcdefgh« Text.

[2.13] – Mapping inline function

```
\ExplSyntaxOn
\cs_set:Npn \ic_funci:n #1 { \fbox{#1}. }
\tl_new:N \l_myici_tl
\NewDocumentCommand { \mycmdi } { m } {
\tl_set:Nn \l_myici_tl { #1 }
\tl_map_inline:Nn \l_myici_tl { \ic_funci:n {##1} }
\par tl ~ = ~ >> \tl_use:N \l_myici_tl <<
}
\ExplSyntaxOff
Text \mycmdi{abc{de}fgh} Text.
```

Text a.b.c.de.f.g.h.
tl = »abcdefgh« Text.

[2.14] – Contents of a token list (1)

```
\ExplSyntaxOn
\tl_new:N \l_myicj_tl
\NewDocumentCommand { \mycmdj } { m } {
\tl_set:Nn \l_myicj_tl { #1 }
\par (a) ~ \tl_count:N \l_myicj_tl  \ token ~ groups.
\par (b) ~ \tl_count_tokens:n { \l_myicj_tl } ~ token
    \int_compare:nNnTF {\tl_count_tokens:n { \l_myicj_tl }} = { 1
    } { } { s }.
\par (c) ~ \exp_args:No \tl_count_tokens:n { \l_myicj_tl } ~
    token \int_compare:nNnTF {\exp_args:No \tl_count_tokens:n {
    \l_myicj_tl }} = { 1 } { } { s } : ~
{ \color{blue} \tl_to_str:N \l_myicj_tl }.
}
\ExplSyntaxOff
\mycmdj{abc{de}fgh}
```

(a) 7 token groups.
(b) 1 token.
(c) 10 tokens: abc{de}fgh.

`\l_myicj_tl` will always be one token.

[2.15] – Contents of a token list (2)

```
\ExplSyntaxOn
\tl_new:N \l_myick_tl
\NewDocumentCommand { \mycmdk } { m } {
\tl_set:Nn \l_myick_tl { #1 }
\par head: ~ \tl_head:N \l_myick_tl
\par tail: ~ \tl_tail:N \l_myick_tl
\par reverse: ~ \tl_reverse:N \l_myick_tl   \tl_use:N \l_myick_tl
    ~<~ \tl_to_str:N \l_myick_tl
\par 5th ~ item : ~ \tl_item:Nn \l_myick_tl { 5 }
\par reverse: ~ \tl_reverse:N \l_myick_tl   \tl_use:N \l_myick_tl
    ~<~ \tl_to_str:N \l_myick_tl
\par reverse ~ items: ~ \exp_args:No \tl_reverse_items:n {
    \l_myick_tl } ~<~ \tl_to_str:N \l_myick_tl
\par 5th ~ item : ~ \tl_item:Nn \l_myick_tl { 5 }
}
\ExplSyntaxOff
\mycmdk{abc{de}fgh}
```

head: a
tail: bcdefgh
reverse: hgfdecba< hgf{de}cba
5th item: c
reverse: abcdefgh< abc{de}fgh
reverse items: hgfdecba < abc{de}fgh
5th item: f

[2.16] – Tokens

```
\ExplSyntaxOn
\tl_new:N \l_myiclz_tl
\cs_set:Npn \ic_funcl:n #1 {
\tl_set:Nn \l_myiclz_tl { #1 }
\fbox{ \strut \tl_to_str:N \l_myiclz_tl } $^ \exp_args:No
    \tl_count_tokens:n { \l_myiclz_tl }$
\int_compare:nNnT
{ \exp_args:No \tl_count_tokens:n { \l_myiclz_tl } }
>
{ 1 }
 { >> \mycmdl{#1} << }
}
\tl_new:N \l_myicl_tl
\NewDocumentCommand { \mycmdl } { m } {
\tl_set:Nn \l_myicl_tl { #1 }
\tl_map_function:NN \l_myicl_tl \ic_funcl:n
}
\ExplSyntaxOff
 \mycmdl{abc{d\textit{e}}fgh}
```

a[1] b[1] c[1] d\textit {e}[5] » d[1] \textit [1] e[1] « f[1] g[1] h[1]

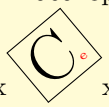
[2.17] – Variants

```
\ExplSyntaxOn
\cs_generate_variant:Nn \tl_count_tokens:n { V }
\tl_new:N \l_myicmz_tl
\cs_set:Npn \ic_funcm:n #1 {
\tl_set:Nn \l_myicmz_tl { #1 }
\fbox{ \strut \tl_to_str:N \l_myicmz_tl } $^ \tl_count_tokens:V
    \l_myicmz_tl $
\int_compare:nNnT
{ \tl_count_tokens:V \l_myicmz_tl }
>
{ 1 }
 { >> \mycmdm{#1} << }
}
\tl_new:N \l_myicm_tl
\NewDocumentCommand { \mycmdm } { m } {
\tl_set:Nn \l_myicm_tl { #1 }
\tl_map_function:NN \l_myicm_tl \ic_funcm:n
}
\ExplSyntaxOff
 \mycmdm{abc{d\textit{e}}fgh}
```

a[1] b[1] c[1] d\textit {e}[5] » d[1] \textit [1] e[1] « f[1] g[1] h[1]

[2.18] – Box

```
\ExplSyntaxOn
\box_new:N \l_myicn_box
\hbox_set:Nn \l_myicn_box { \fbox { \Huge C \color{red}\tiny e } }
wd = \dim_use:N \box_wd:N \l_myicn_box \par
ht = \dim_use:N \box_ht:N \l_myicn_box \par
dp = \dim_use:N \box_dp:N \l_myicn_box \par
xxx \box_use:N \l_myicn_box xxx \par
\box_rotate:Nn \l_myicn_box { 40 }
wd = \dim_use:N \box_wd:N \l_myicn_box \par
ht = \dim_use:N \box_ht:N \l_myicn_box \par
dp = \dim_use:N \box_dp:N \l_myicn_box \par
xxx \box_use:N \l_myicn_box xxx
\ExplSyntaxOff
```

wd=26.47482pt
ht=20.79112pt
dp=3.79807pt
xxx  xxx
wd=36.08652pt
ht=32.94461pt
dp=2.90948pt
xxx  xxx

[2.19] – Box 2

```
\ExplSyntaxOn
\box_new:N \l_myico_box
\box_new:N \l_myicov_box
\cs_set:Npn \ic_funco: { 0.72 }
\cs_set:Npn \ic_funcoz:nn #1#2 {
\hbox_set:Nn \l_myico_box {  { \Huge #1 \color{red}\tiny #2 } }
\hbox_set:Nn \l_myicov_box {  \color{red}\Huge #1 }
\box_use:N \l_myico_box\llap {
\box_rotate:Nn \l_myico_box { 40 } \box_scale:Nnn \l_myico_box
    {\ic_funco:}{\ic_funco:}
\box_use:N \l_myico_box\llap {
\box_rotate:Nn \l_myico_box { 40 } \box_scale:Nnn \l_myico_box
    {\ic_funco:}{\ic_funco:}
\box_use:N \l_myico_box\llap {
\box_rotate:Nn \l_myico_box { 40 } \box_scale:Nnn \l_myico_box
    {\ic_funco:}{\ic_funco:}
\box_use:N \l_myico_box\llap {
\box_rotate:Nn \l_myico_box { 40 } \box_scale:Nnn \l_myico_box
    {\ic_funco:}{\ic_funco:}
\box_use:N \l_myico_box
}}}} \hspace{-\box_wd:N \l_myico_box} \box_use:N \l_myicov_box}
xxx \ic_funcoz:nn {A}{d} ~xxx \ic_funcoz:nn {C}{e} ~xxx
    \ic_funcoz:nn {X}{y} ~xxx \ic_funcoz:nn {O}{p} ~xxx
    \ic_funcoz:nn {B}{w} ~xxx \ic_funcoz:nn {I}{o}
\par xxx \ic_funcoz:nn {CAT}{dog} ~xxx \ic_funcoz:nn
    {Caterpillar}{} ~xxx \ic_funcoz:nn {C~~}{--} ~ \ic_funcoz:nn
    {O~~}{--}
\ExplSyntaxOff
```

[2.20] – Box 3

```
\ExplSyntaxOn
\box_new:N \l_myicpz_box
\box_new:N \l_myicpy_box
\cs_set:Npn \ic_funcp:nn #1#2 {
\hbox_set:Nn \l_myicpz_box {  { #1 } }
\hbox_set:Nn \l_myicpy_box {  { \color{#2}#1 } }
 \box_use:N \l_myicpz_box
 { \hspace{-\box_wd:N \l_myicpy_box} \hspace{0.43pt} \box_use:N
    \l_myicpy_box }
}
 xxx\Huge \ic_funcp:nn { C } { green }
\NewDocumentCommand { \mycmdp } { m m } { \ic_funcp:nn { #1 } {
    #2 } }
\ExplSyntaxOff
  ~ and \mycmdp{cat}{red}  and \mycmdp{dog}{brown}
```

xxxC and cat and dog

[2.21] – Sequence

```
\ExplSyntaxOn
\cs_generate_variant:Nn \tl_count_tokens:n { V }
\seq_new:N \l_myicq_seq
\seq_new:N \l_myicqz_seq
\seq_new:N \l_myicqy_seq
\tl_new:N \l_myicqz_tl
\seq_new:N \l_myicqyy_seq
\int_new:N \l_myicqz_int
\int_new:N \l_myicqy_int
\NewDocumentCommand { \mycmdq } { m } {
\seq_set_split:Nnn \l_myicq_seq { , } { #1 }
\seq_use:Nnnn \l_myicq_seq { - } { - } { - } ~ unsorted\par
\seq_sort:Nn \l_myicq_seq
{
\seq_clear:N \l_myicqz_seq
\seq_clear:N \l_myicqy_seq
\seq_set_split:Nnn \l_myicqz_seq { ; } { ##1 }
\seq_set_split:Nnn \l_myicqy_seq { ; } { ##2 }
%\seq_show:N \l_myicqz_seq
\tl_set:Nx \l_myicqz_tl { \seq_item:Nn \l_myicqz_seq {1} }
\tl_set:Nx \l_myicqy_tl { \seq_item:Nn \l_myicqy_seq {1} }
%\tl_show:N \l_myicqz_tl  \tl_show:N \l_myicqy_tl
\int_set:Nn \l_myicqz_int { \tl_count_tokens:V \l_myicqz_tl }
\int_set:Nn \l_myicqy_int { \tl_count_tokens:V \l_myicqy_tl }
%\int_show:N \l_myicqz_int
%%>>\seq_item:Nn \l_myicqz_seq {1}<<
\int_compare:nNnTF { \l_myicqz_int } < { \l_myicqy_int }
{ \sort_return_swapped: }
{ \sort_return_same: }
}
\seq_use:Nnnn \l_myicq_seq { - } { - } { - } ~ sorted
}
\ExplSyntaxOff
  \mycmdq{a;x,abv;y,ab;z,b;xyz,bb;i}
```

a;x-abv;y-ab;z-b;xyz-bb;i unsorted
abv;y-ab;z-bb;i-a;x-b;xyz sorted

[2.22] – Sequence 2

```
\ExplSyntaxOn
\cs_generate_variant:Nn \tl_replace_all:Nnn { Noo }
\seq_new:N \l_myicr_seq
\seq_new:N \l_myicrz_seq
\tl_new:N \l_myicrz_tl
\tl_new:N \l_myicrx_tl
\seq_set_split:Nnn \l_myicr_seq { , } { abv;y,ab;z,bb;i,
    a;x,b;xyz }
\cs_set:Npn \ic_funcrz:n #1 { %replace
\seq_set_split:Nnn \l_myicrz_seq { ; } { #1 }
\tl_set:Nx \l_myicrz_tl { \seq_item:Nn \l_myicrz_seq {1} }
\tl_set:Nx \l_myicrx_tl { \seq_item:Nn \l_myicrz_seq {2} }
\tl_replace_all:Noo \l_myicr_tl  { \l_myicrz_tl } { \l_myicrx_tl
    . }
}
\tl_new:N \l_myicr_tl
\NewDocumentCommand { \mycmdr } { m } {
\tl_set:Nn \l_myicr_tl { #1 }
\tl_map_function:NN \l_myicr_seq \ic_funcrz:n
\tl_use:N \l_myicr_tl
}
\ExplSyntaxOff
  \mycmdr{aabvabbbba{}b}
```

x.y.z.i.xyz.x.xyz.

*=====

3 regexpatch

[3.1] – testca

```
\newcommand{\testca}{\textit{label}}
Before: \testca
\par \regexpatchcmd{\testca}{\c{textit}}{\c{textbf}}{S}{F}
\par \xpatchcmd{\testca}{label}{babble}{S}{F}
\par After: \testca
```

Before: *label*
S
S
After: **babble**

[3.2] – ph: Too many brace levels

```
\newcommand{\ph}[1]{
\textbf{\textsc{{\color{blue}#1}}}\ \ }
Before: {\testfont\ph{Snail in the Bottle}}
\regexpatchcmd{\ph}{\c{color}\cB\{blue\cE\}}{red}{}{F}
\par After: {\testfont\ph{Snail in the Bottle}}
```

Before: **SNAIL IN THE BOTTLE** F
After: **SNAIL IN THE BOTTLE**

[3.3] – ph2: Two levels of braces

```
\newcommand{\phb}[1]{\textsc{{\color{blue}#1}}\ \ }
Before: {\testfont\phb{Snail in the Bottle}}
\regexpatchcmd{\phb}{\c{color}\cB\{blue\cE\}}{red}{}{F}
\par After: {\testfont\phb{Snail in the Bottle}}
```

Before: SNAIL IN THE BOTTLE
After: REDSNAIL IN THE BOTTLE

[3.4] – ph3: Entire \color command replaced

```
\newcommand{\phc}[1]{{
\bfseries\scshape\color{blue}#1\ \ }}
Before: {\testfont\phc{Snail in the Bottle}}
\regexpatchcmd{\phc}{\c{color}\cB\{blue\cE\}}
    {\c{color}\cB\{red\cE\}}{}{F}
\par After: {\testfont\phc{Snail in the Bottle}}
```

Before: **SNAIL IN THE BOTTLE**
After: **SNAIL IN THE BOTTLE**

[3.5] – ph4: Text replaced: 'blue' > 'red'

```
\newcommand{\phd}[1]{{
\bfseries\scshape\color{blue}#1\ \ }}
Before: {\testfont\phd{Snail in the Bottle}}
\xpatchcmd{\phd}{blue}{red}{}{F}
\par After: {\testfont\phd{Snail in the Bottle}}
```

Before: **SNAIL IN THE BOTTLE**
After: **SNAIL IN THE BOTTLE**

[3.6] – ph5: Text ('blue') replaced by a macro ('\mycolour')

```
\newcommand{\mycolour}{green}
\newcommand{\phe}[1]{{
\bfseries\scshape\color{blue}#1\ \ }}
Before: {\testfont\phe{Snail in the Bottle}}
\regexpatchcmd{\phe}{blue}{\c{mycolour}}{}{F}
\par After: {\testfont\phe{Snail in the Bottle}}
```

Before: **SNAIL IN THE BOTTLE**
After: **SNAIL IN THE BOTTLE**

[3.7] – ph6: Multi-level grouping without braces[a]

[a] "patchable" = it can be reconstructed from its decomposition under the current category code egime. – Manual, §7.1 (2018/05/02)

```
\newcommand{\mycolour}{brown}
\newcommand{\phf}[1]{\begingroup
\bfseries\begingroup\scshape\begingroup\color{blue}#1\endgroup\
    smallcaps\endgroup \ bold\endgroup\ normal \ \ }
Before: {\testfont\phf{Snail in the Bottle}}
\regexpatchcmd{\phf}{blue}{\c{mycolour}}{}{F}
\par After: {\testfont\phf{Snail in the Bottle}}
```

Before: **SNAIL IN THE BOTTLE SMALLCAPS bold** normal
After: **SNAIL IN THE BOTTLE SMALLCAPS bold** normal

\dca{x} : nostar; noopt=-NoValue-; MArg=x.
\dca*{y} : star; noopt=-NoValue-; MArg=y.
\dca[abc]{z} : nostar; OArg=abc; MArg=z.
\dca[xyz]{zz} : nostar; OArg=xyz; MArg=zz.

\dca*[xyzz]{zzz} : star; OArg=xyzz; MArg=zzz.

[3.8] – ph4a: Text replaced: 'blue' > 'red'

```
1 \newcommand{\phda}[2]{{
2 \bfseries\scshape\color{blue}#1\normalcolor\ in the
      \color{blue}#2}}
3 Before: {\testfont\phda{Snail}{Bottle}}
4 \xpatchcmd{\phda}{blue}{red}{}{F}
5 \par After: {\testfont\phda{Snail}{Bottle}}
```

Before: **SNAIL IN THE BOTTLE**
After: **SNAIL IN THE BOTTLE**

[3.9] – ph4b: Text replaced: all 'blue' > 'red'

```
1 \newcommand{\phdb}[2]{{
2 \bfseries\scshape\color{blue}#1\normalcolor\ in the
      \color{blue}#2}}
3 Before: {\testfont\phdb{Snail}{Bottle}}
4 \xpatchcmd*{\phdb}{blue}{red}{}{F}
5 \par After: {\testfont\phdb{Snail}{Bottle}}
```

Before: **SNAIL IN THE BOTTLE**
After: **SNAIL IN THE BOTTLE**