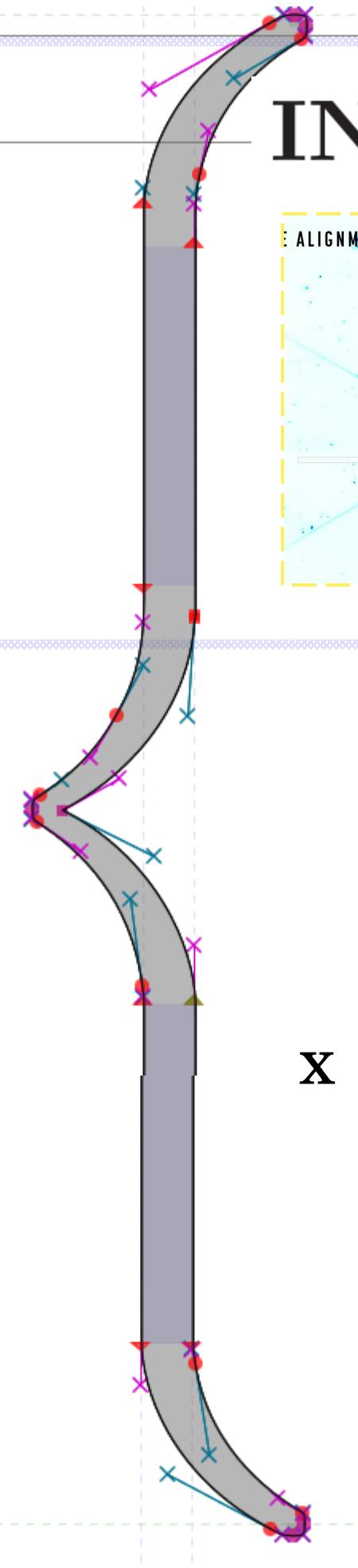
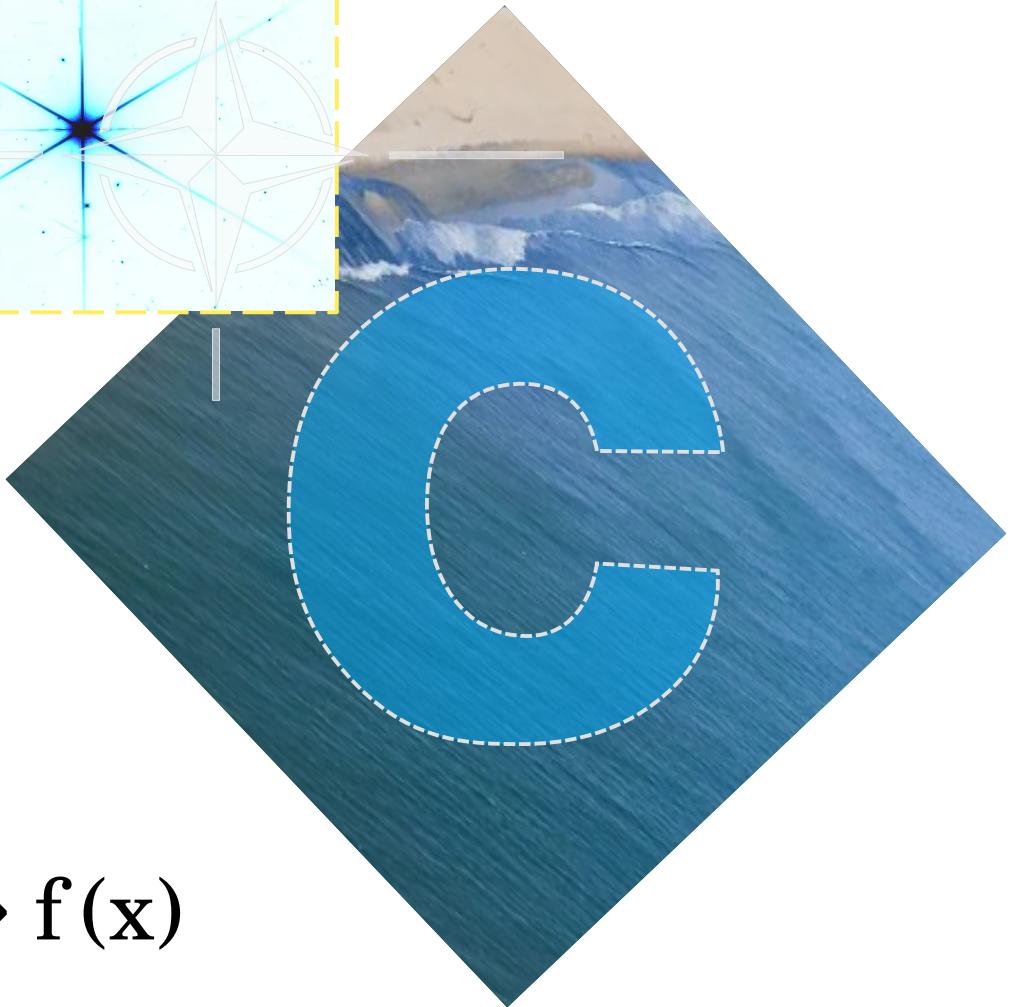
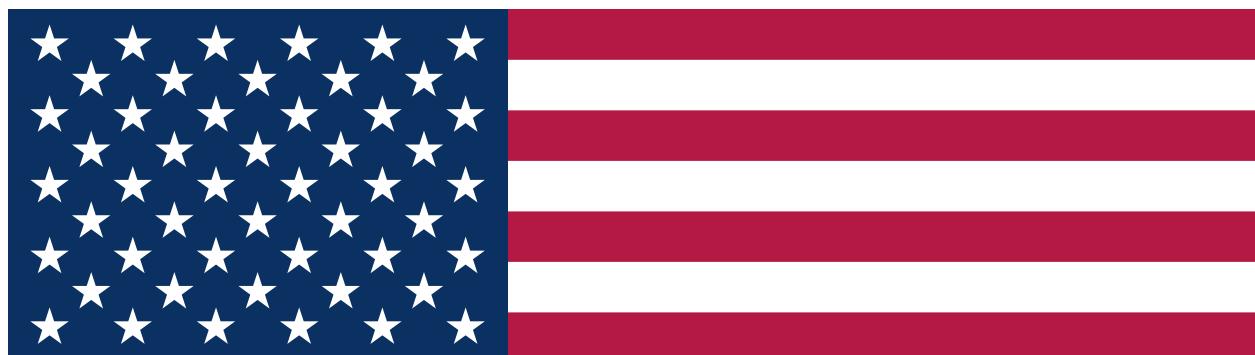
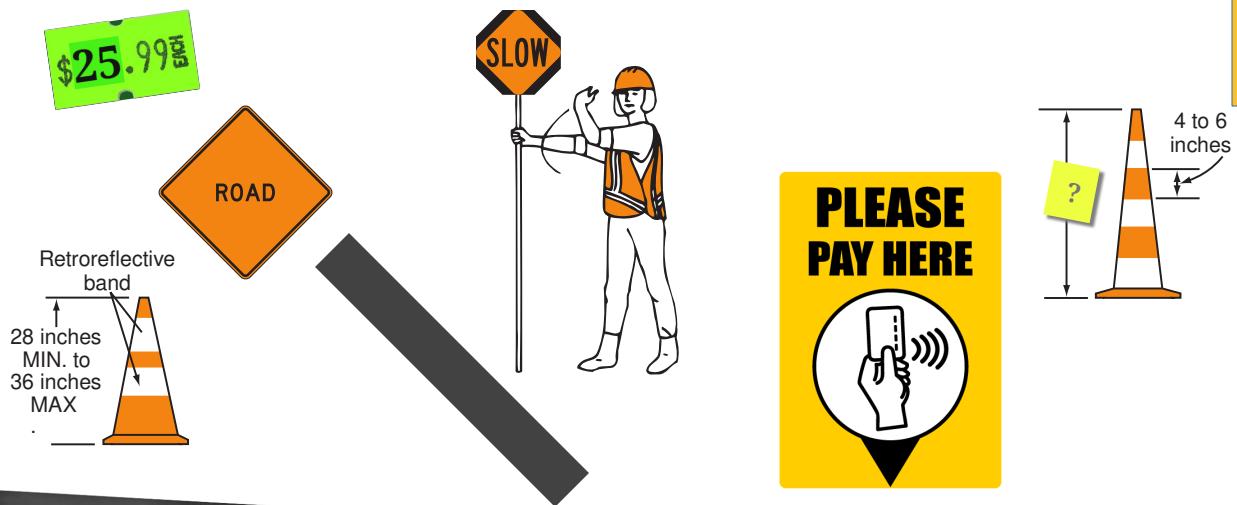


INTER-UNIVERSAL LANGUAGE



$$x \mapsto f(x)$$







the result of post-composing φΘ
with the “poly-action”

Chapter 1 - A Tutorial Introduction

Let us begin with a quick introduction in C. Our aim is to show the essential elements of the language in real programs, but without getting bogged down in details, rules, and exceptions. At this point, we are not trying to be complete or even precise (save that the examples are meant to be correct). We want to get you as quickly as possible to the point where you can write useful programs, and to do that we have to concentrate on the basics: variables and constants, arithmetic, control flow, functions, and the rudiments of input and output. We are intentionally leaving out of this chapter features of C that are important for writing bigger programs. These include pointers, structures, most of C's rich set of operators, several control-flow statements, and the standard library.

This approach and its drawbacks. Most notable is that the complete story on any particular feature is not found here, and the tutorial, by being brief, may also be misleading. And because the examples do not use the full power of C, they are not as concise and elegant as they might be. We have tried to minimize these effects, but be warned. Another drawback is that later chapters will necessarily repeat some of this chapter. We hope that the repetition will help you more than it annoys.

In any case, experienced programmers should be able to extrapolate from the material in this chapter to their own programming needs. Beginners should supplement it by writing small, similar programs of their own. Both groups can use it as a framework on which to hang the more detailed descriptions that begin in [Chapter 2](#).

Chapter 1 - A Tutorial Introduction

Let us begin with a quick introduction in C. Our aim is to show the essential elements of the language in real programs, but without getting bogged down in details, rules, and exceptions. At this point, we are not trying to be complete or even precise (save that the examples are meant to be correct). We want to get you as quickly as possible to the point where you can write useful programs, and to do that we have to concentrate on the basics: variables and constants, arithmetic, control flow, functions, and the rudiments of input and output. We are intentionally leaving out of this chapter features of C that are important for writing bigger programs. These include pointers, structures, most of C's rich set of operators, several control-flow statements, and the standard library.

This approach and its drawbacks. Most notable is that the complete story on any particular feature is not found here, and the tutorial, by being brief, may also be misleading. And because the examples do not use the full power of C, they are not as concise and elegant as they might be. We have tried to minimize these effects, but be warned. Another drawback is that later chapters will necessarily repeat some of this chapter. We hope that the repetition will help you more than it annoys.

In any case, experienced programmers should be able to extrapolate from the material in this chapter to their own programming needs. Beginners should supplement it by writing small, similar programs of their own. Both groups can use it as a framework on which to hang the more detailed descriptions that begin in [Chapter 2](#).

Chapter 1 - A Tutorial Introduction

Let us begin with a quick introduction in C. Our aim is to show the essential elements of the language in real programs, but without getting bogged down in details, rules, and exceptions. At this point, we are not trying to be complete or even precise (save that the examples are meant to be correct). We want to get you as quickly as possible to the point where you can write useful programs, and to do that we have to concentrate on the basics: variables and constants, arithmetic, control flow, functions, and the rudiments of input and output. We are intentionally leaving out of this chapter features of C that are important for writing bigger programs. These include pointers, structures, most of C's rich set of operators, several control-flow statements, and the standard library.

This approach and its drawbacks. Most notable is that the complete story on any particular feature is not found here, and the tutorial, by being brief, may also be misleading. And because the examples do not use the full power of C, they are not as concise and elegant as they might be. We have tried to minimize these effects, but be warned. Another drawback is that later chapters will necessarily repeat some of this chapter. We hope that the repetition will help you more than it annoys.

In any case, experienced programmers should be able to extrapolate from the material in this chapter to their own programming needs. Beginners should supplement it by writing small, similar programs of their own. Both groups can use it as a framework on which to hang the more detailed descriptions that begin in [Chapter 2](#).

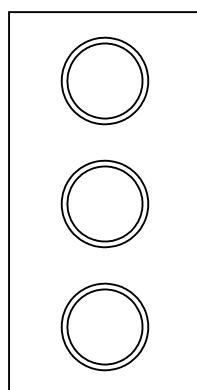
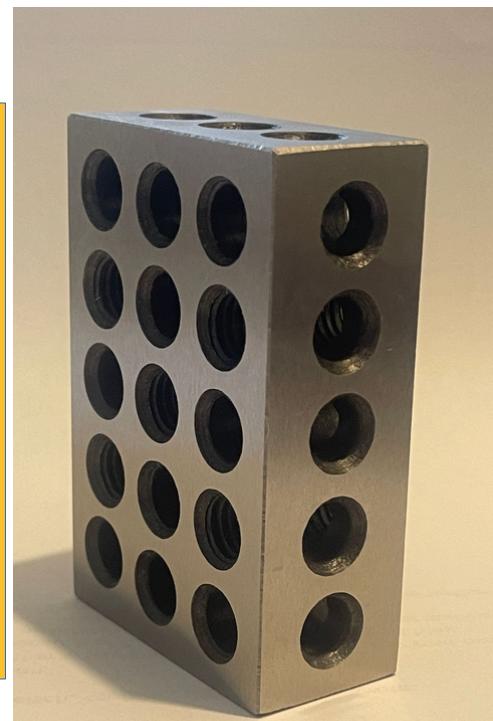
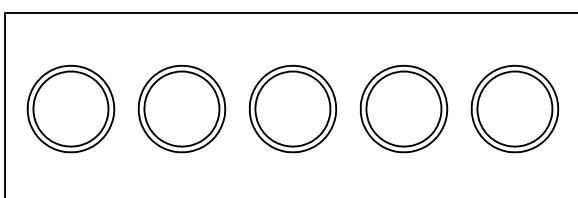
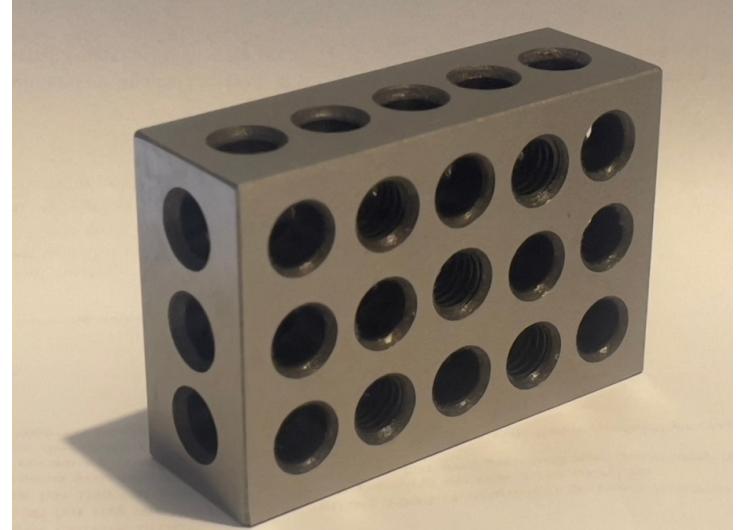
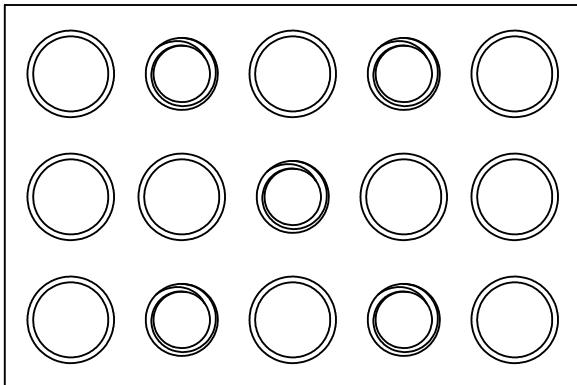


Figure 6H-6. Shoulder Work with Minor Encroachment (TA-6)

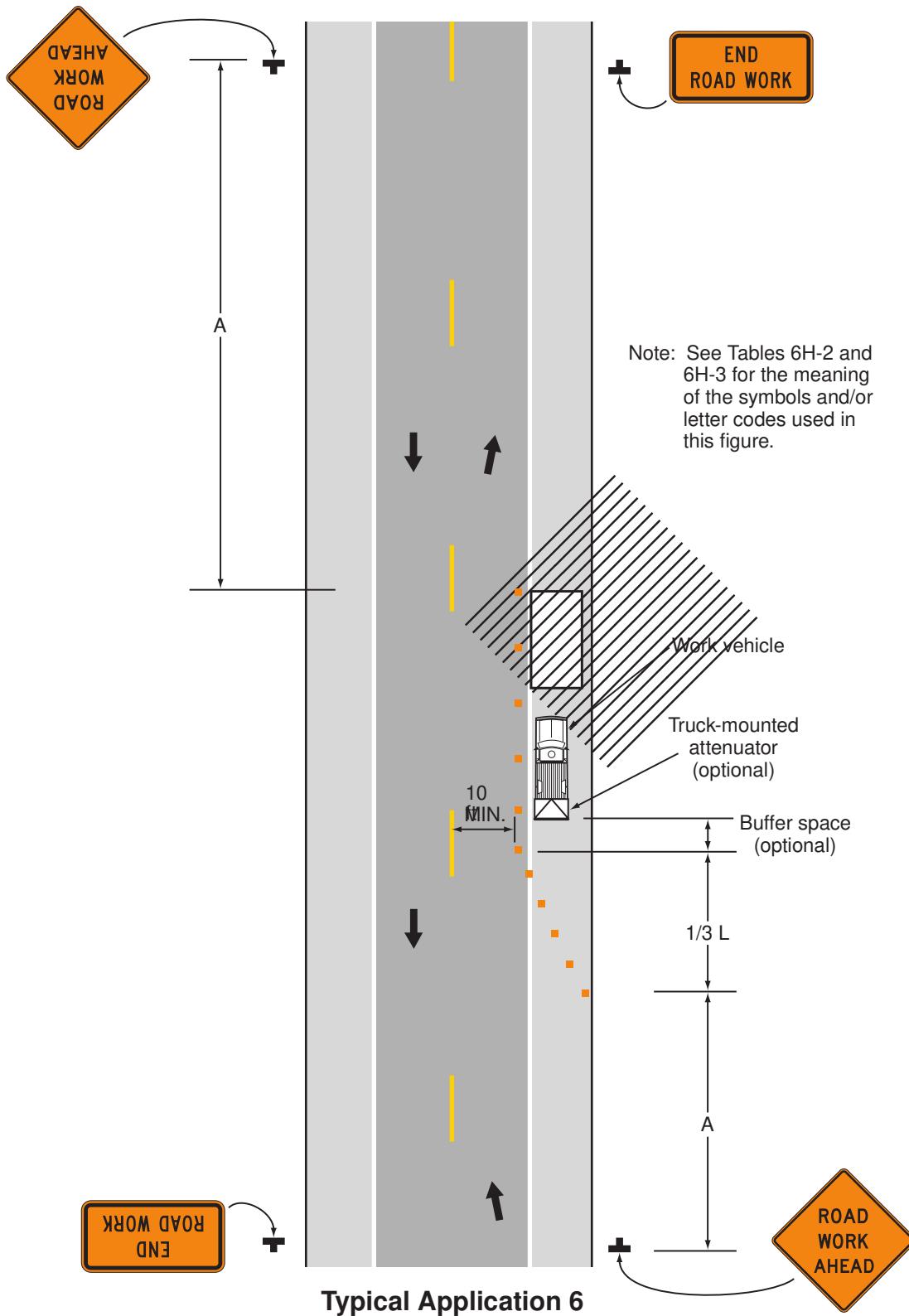


Figure 6C-3. Example of a One-Lane, Two-Way Traffic Taper

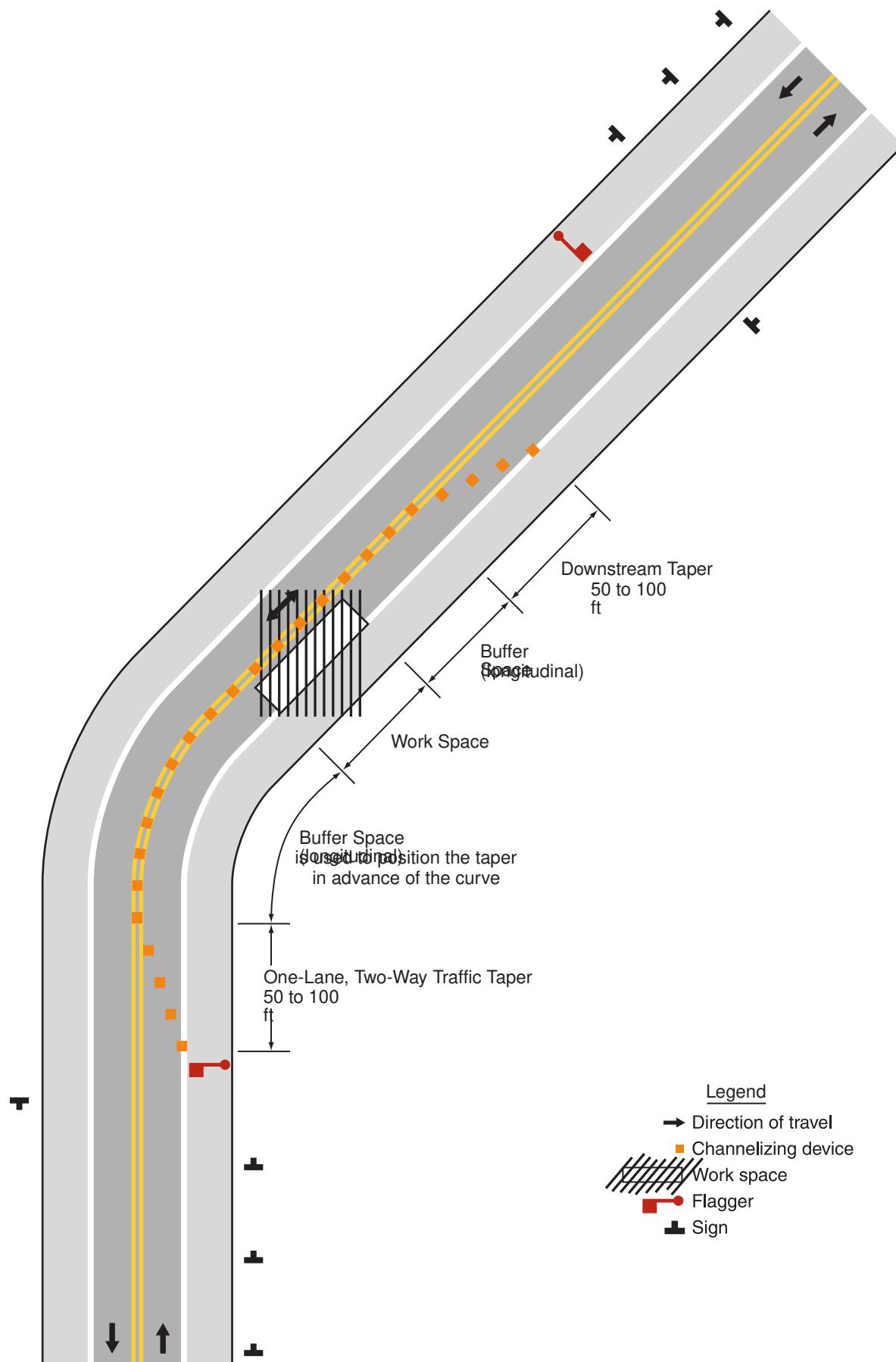


Figure 6F-3. Regulatory Signs and Plaques in Temporary Traffic Control Zones
 (Sheet 1 of 2)



Figure 6F-3. Regulatory Signs and Plaques in Temporary Traffic Control Zones
(Sheet 2 of 2)



R9-9



R9-10



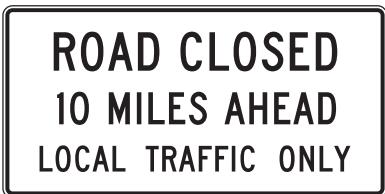
R9-11



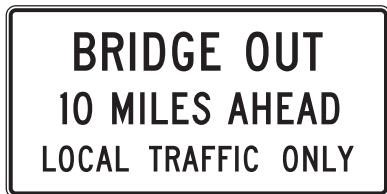
R9-11a



R11-2



R11-3a



R11-3b



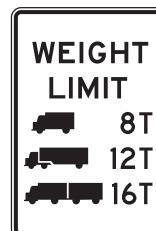
R11-4



R12-1



R12-2



R12-5

Figure 6F-4. Warning Signs and Plaques in Temporary Traffic Control Zones
(Sheet 1 of 3)



Figure 6F-4. Warning Signs and Plaques in Temporary Traffic Control Zones
 (Sheet 2 of 3)

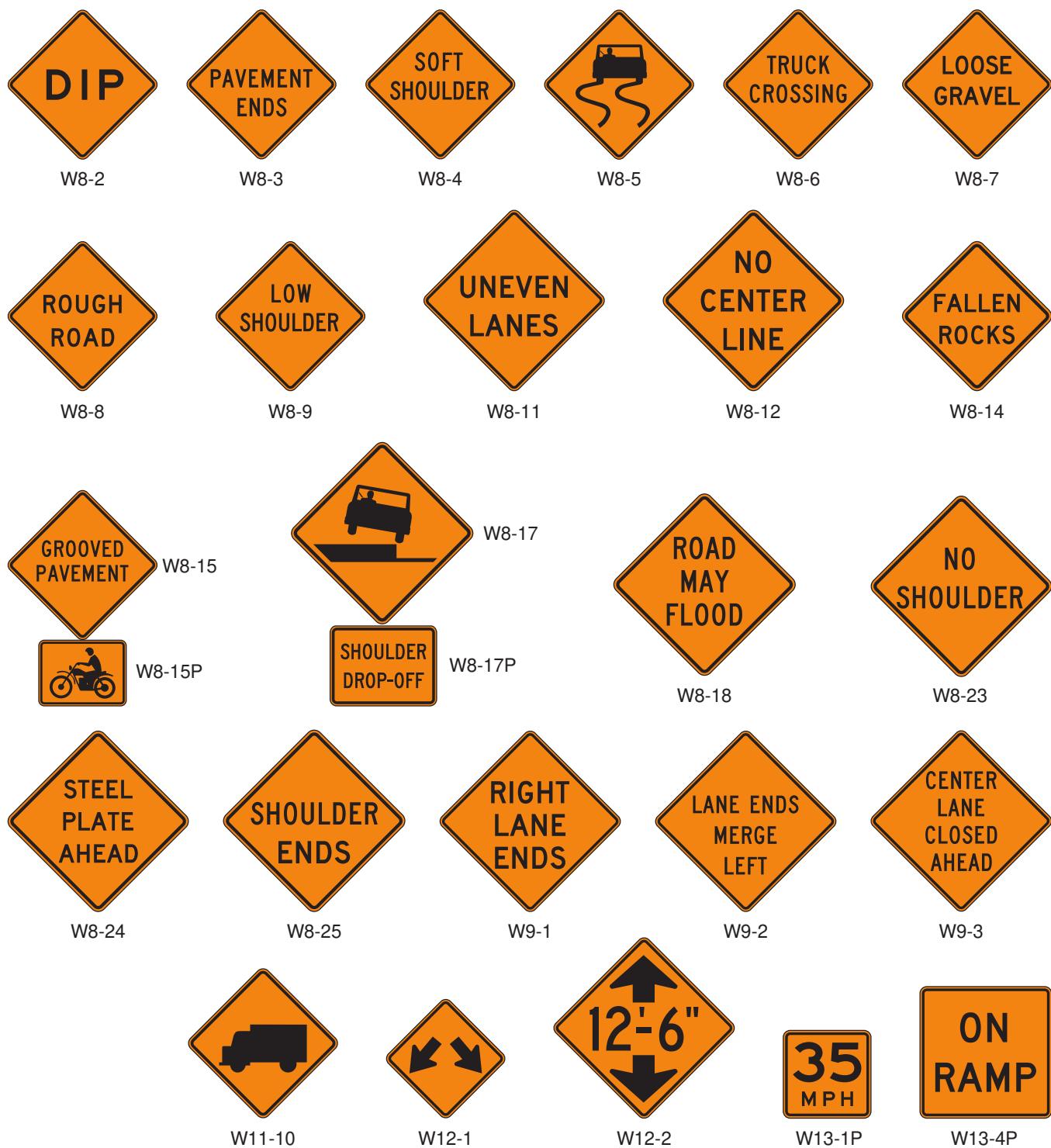


Figure 6F-4. Warning Signs and Plaques in Temporary Traffic Control Zones (Sheet 3 of 3)

* An optional STREET WORK word message sign is shown in the "Standard Highway Signs and Markings" book.

** An optional STREET CLOSED word message sign is shown in the "Standard Highway Signs and Markings" book.

*** An optional FLAGGER (W20-7a) word message sign is shown in the "Standard Highway Signs and Markings" book.

**** An optional FRESH TAR word message sign is shown in the "Standard Highway Signs and Markings" book.

$\mathfrak{Z}(\dagger G_{\underline{v}})^0$ for the portion of $\dagger \mathfrak{D}^\perp$ indexed by \underline{v} [cf. the notation Corollary 4.5], then the algorithms for constructing “ $k^\sim(G)$ ”, “ $\mathcal{I}(G)$ ” [opIII], Proposition 5.8, (ii), yield a **functorial algorithm** in the process of constructing an ind-topological module equipped with a continuous

$$\underline{\log}(\dagger \mathcal{D}_{\underline{v}}^\perp) \stackrel{\text{def}}{=} \left\{ \dagger G_{\underline{v}} \curvearrowright k^\sim(\dagger G_{\underline{v}}) \right\}$$

local submodule — i.e., a “**mono-analytic log-shell**” —

$$\mathcal{I}_{\dagger \mathcal{D}_{\underline{v}}^\perp} \stackrel{\text{def}}{=} \mathcal{I}(\dagger G_{\underline{v}}) \subseteq k^\sim(\dagger G_{\underline{v}})$$

its $p_{\underline{v}}$ -adic log-volume [cf. [AbsTopIII], Corollary 5.10, (iv)]. Moreover, there is a natural **functorial algorithm** [cf. the second display of [IUTchI], (ii)] in the collection of data $\dagger \mathcal{F}_{\underline{v}}^{+ \times \mu}$ [i.e., the portion of $\dagger \mathfrak{F}^{+ \times \mu}$ labeling an **Ism-orbit of isomorphisms** [cf. [IUTchII], Example 1.1, Definition 4.9, (i), (vii)]]

Figure 6H-25. Multiple Lane Closures at an Intersection (TA-25)

