

# Typesetting Sequential Function Chart (SFC) with L<sup>A</sup>T<sub>E</sub>X and TikZ

Luis Paulo Laus  
e-mail: laus@utfpr.edu.br

Version: 1.0, Version date: 2018-01-16

## 1 Abstract

Sequential Function Chart (SFC) describes the behaviour of a control program. It is derived from Petri Nets and GRAFCET. This package contains a set of symbols that simplifies the drawing of SFC diagrams with emphasis in readability and in agreement with the international standard IEC 61131-3. It extends the TikZ-library `circuits` which defines general keys for creating circuits. All graphic elements of SFC are supported by `tikz-sfc`: steps (normal and initial), transitions, actions and actions qualifiers (with and without time duration) are supported. The placement of all actions associated with a step is done automatically. Comments can also be placed automatically in a suitable location near the element they refer to. The final appearance of each element can be fine-tuned individually or in group.

## 2 Introduction

IEC 61131-3 Sequential Function Chart (SFC) describes the behaviour of a control program. It is derived from Petri Nets and IEC 60848 GRAFCET, with the changes necessary to convert the representation from a documentation standard to a set of execution control elements.

The GRAFCET language, described by IEC 60848, has served as foundation of SFC language of IEC standard 61131-3, but the syntax and the semantics defined by each of the two standards are nevertheless distinct because their scopes are different. GRAFCET is regarded as a *specification language* and SFC as an *implementation of the language for programming*. Many features of GRAFCET are not supported in SFC, but all GRAFCET functionalities can be implemented in SFC, although not directly. Particularly, assignation condition, time dependent assignation condition, delayed action, time limited action, stored actions, action on activation, action on deactivation, action at the clearing and action on event are all done, one way or another, in SFC using action qualifiers<sup>1</sup> (mainly). Consequently, the graphical representation of those features is fairly different. If you are interested in GRAFCET, refer to Robert Papanicola's `grafcet` package<sup>2</sup>.

SFC structures the internal organization of a program, and helps to decompose a control problem into manageable parts, while maintaining the overview.

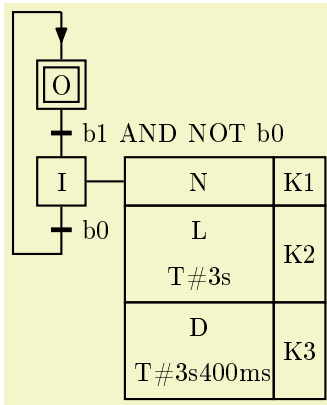
SFC consists of steps and transitions interconnected by directed links. Associated with each step is a set of actions, and with each transition is associated a transition condition. Each step represents a particular state of the systems being controlled. A transition is associated with a condition, which, when true, causes the step before the transition to be deactivated, and the next step to be activated. Steps are linked to actions, performing a certain control action. An action qualifier can be used in the association of a step with an action. Each element can be programmed in any of the standard languages defined in IEC 61131-3, including SFC itself.

Steps (normal and initial), transitions, actions and actions qualifiers (with and without time) are shown in the example below for a star-delta motor starter with dead time.

---

<sup>1</sup>This package typesets qualifiers in the agreement with the international standard IEC 61131-3/2013. The lack of qualifier support in GRAFCET and in particular in the `grafcet` package is the main reason to have a different package for SFC.

<sup>2</sup>The `grafcet` package uses a very different approach in the user interface. It is based in writing down a sequence of macro calls. The `tikz-sfc` uses the TikZ-library `circuits` interface in which you place symbols along a path.



```

\begin{tikzpicture}[circuit plc sfc,thick,x=2.6\tikzcircuitssizeunit,
y=1.3\tikzcircuitssizeunit,sfcaqw=8\tikzcircuitssizeunit]
\draw (0,0)
to [flow direction={pos=0.95}] ++(0,-1)
to [sfcestepi={info=0}] ++(0,-4)
node [sfctransition={info=b1 AND NOT b0}] {}
to [sfcestep={info=I,
sfcaction={info=K1,qualifier=N},
sfcaction={info=K2,qualifier=L,time=T\#3s},
sfcaction={info=K3,qualifier=D,time=T\#3s400ms}}] ++(0,-4)
node [sfctransition={info=b0}] {}
|- ++(-1,-1) |- (0,0);
\end{tikzpicture}

```

### 3 SFC Library

#### TikZ Library `circuits.plc.sfc`

```

\usepgflibrary{circuits.plc.sfc} %  $\TeX$  and plain  $\TeX$  and pure pgf
\usepgflibrary[circuits.plc.sfc] % Con $\TeX$ t and pure pgf
\usetikzlibrary{circuits.plc.sfc} %  $\TeX$  and plain  $\TeX$  when using TikZ
\usetikzlibrary[circuits.plc.sfc] % Con $\TeX$ t when using TikZ

```

This library provides graphics for Sequential Function Chart (SFC) related to programmable controllers (PLC) and according to the international standard IEC-61131-3. The library was written to extend the standard TikZlibrary `circuit`. The reader is urged to read the Section “Circuit Libraries” of TikZ manual. This library defines the following keys:

`/tikz/circuit plc sfc` (no value)

This key should be passed as an option to a picture or a scope that contains a SFC. It will do some internal setups.

`/tikz/sfcaqw= $\langle width \rangle$`  (no default, initially  $0 \backslash tikzcircuitssizeunit$ )

This key sets the qualifier width. See Section 9 for details.

`/tikz/sfcanw= $\langle width \rangle$`  (no default, initially  $2.6 \backslash tikzcircuitssizeunit$ )

This key sets the action name width. See Section 9 for details.

`/tikz/sfcah= $\langle height \rangle$`  (no default, initially  $2.6 \backslash tikzcircuitssizeunit$ )

This key sets the action height. See Section 9 for details.

`/tikz/time= $\langle time \rangle$`  (no default)

This key sets the action duration time to be used with qualifier D, L, DS, SD, LS. See Section 9 for details.

`/tikz/qualifier= $\langle qualifier \rangle$`  (no default)

This key sets the action qualifier. See Section 9 for details.

`/tikz/flow direction arrow` (style, no value)

The value of this key will be used for the arrow tip for flow direction as depicted in `direction ee` shape in TikZ manual.

### 4 Convention

In this package, the main information about the language element is written with `info= $\langle text \rangle$`  and comments with `info'= $\langle comment \rangle$` . The `tikz-sfc` package takes care of placing those texts in the appropriated position. For steps and actions, `info` places the  $\langle text \rangle$  inside the block that represents the element. For transitions, `info` places the  $\langle text \rangle$  at the right side of the transition symbol. Comments are placed at left side of steps and transitions and right side of actions. This behaviour can be override using the  $\langle angle \rangle$

option. For instance, if you want to place a comment at the right side of a step, you need to type something like `info'={right:(* first step *)}`. See Section “The Label Option” in TikZ manual for details.

## 5 Steps

A step represents a situation in which the behaviour of a program organization unit (POU) with respect to its inputs and outputs follows a set of rules defined by the associated actions of the step. A step is either active or inactive. At any given moment, the state of the program organization unit is defined by the set of active steps and the values of its internal and output variables.

A step shall be represented graphically by a block containing a step name in the form of an identifier. The directed link(s) into the step can be represented graphically by a vertical line attached to the top of the step. The directed link(s) out of the step can be represented by a vertical line attached to the bottom of the step.

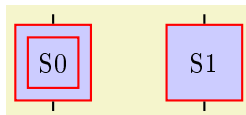
The initial state of the program organization unit is represented by the initial values of its internal and output variables, and by its set of initial steps, i.e., the steps which are initially active. Each SFC network shall have exactly one initial step.

The graphical form with directed links of an initial step and a step (normal<sup>3</sup>) are shown below:



One or more actions (actually, action blocks) can be associated with a step. See Section 9 for details.

The initial size of all steps is  $2.6 \times 2.6 \text{tikzcircuitssizeunit}$ . This size and other features can be changed globally using the style `every sfcstepi/.style` for initial steps and `every sfcstep/.style` for steps. For example, to change the size of all steps to  $1 \times 1 \text{cm}$ , the line colour to red, fill them in light blue and change the distance<sup>4</sup> between the lines of the initial step to 1.8mm, you can place `every sfcstep/.style={red, fill=blue!20, minimum width=1cm, minimum height=1cm, inner sep=0.3mm}, every sfcstepi/.style=every sfcstep` as an option to your TikZ command. You can also use the command `\tikzset` to set it document wide.



```
\begin{tikzpicture}[circuit plc sfc,thick,y=1.3|tikzcircuitssizeunit,
every sfcstep/.style={red, fill=blue!20, minimum width=1cm,
minimum height=1cm, inner sep=0.3mm},
every sfcstepi/.style=every sfcstep]
\draw (0,0)
to [sfcstepi={info=S0}] (0,-4);
\draw (2,0)
to [sfcstep={info=S1}] (2,-4);
\end{tikzpicture}
```

Every step need a step name and it is placed inside the block by the `info` key.

`/tikz/info=[(options)](angle):(text)` (no default)

This key has nearly the same effect as the `label` key, only the placement position is change to the block centre and following style is used additionally and automatically:

`/tikz/every info` (style, no value)

Set this style to configure the styling of info labels. Since this key is *not* used with normal labels, it provides an easy way of changing the way info labels look without changing other labels.

The `(options)` and `(angle)` are passed directly to the `label` command.



```
\begin{tikzpicture}[circuit plc sfc,thick,y=1.3|tikzcircuitssizeunit,
every info/.style=red]
\draw (0,0)
to [sfcstepi={info=S0}] (0,-4);
\draw (2,0)
to [sfcstep={info=S1}] ++(0,-4);
\end{tikzpicture}
```

<sup>3</sup>A step which is not initial is just called “step”, the “normal” is unappropriated, that is why is was written between parenthesis. This document is full of pedantic comments like this.

<sup>4</sup>The default value is six times the inner separation which, by its time, is half the line width. Thus, just divide the desire among by six.

For a detailed discussion of the `label` option refer to the TikZ manual.


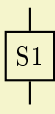
`/tikz/info'=[<options>]<angle>:<text>` (no default)

This key is meant to place a comment at the *left* side of the step<sup>5</sup>. It has nearly the same effect as the `label` key, only the placement position is change and following style is used additionally and automatically:

`/tikz/every info'` (style, no value)

Set this style to configure the styling of info' labels. Since this key is *not* used with normal labels, it provides an easy way of changing the way info labels look without changing other labels.

The *<options>* and *<angle>* are passed directly to the `label` command.

<p>(* initial step *)</p> 	<p>(* step or "normal" step *)</p> 	<pre>\begin{tikzpicture}[circuit plc sfc,thick,y=1.3 tikzcircuitssizeunit,   every info'/.style={red,font=\ttfamily,text width=2cm}]   \draw (0,0)     to [sfcstepi={info=S0,info'={(* initial step *)}}] (0,-4);   \draw (0,-5)     to [sfcstep={info=S1,info'={(* step or "normal" step *)}}] ++(0,-4); \end{tikzpicture}</pre>
---	--	---

For a detailed discussion of the `label` option refer to the TikZ manual.

## 6 Transitions

A transition represents the condition whereby control passes from one or more steps preceding the transition to one or more successor steps along the corresponding directed link. The transition shall be represented by a horizontal line across the vertical directed link.

The direction of evolution following the directed links shall be from the bottom of the predecessor step(s) to the top of the successor step(s).

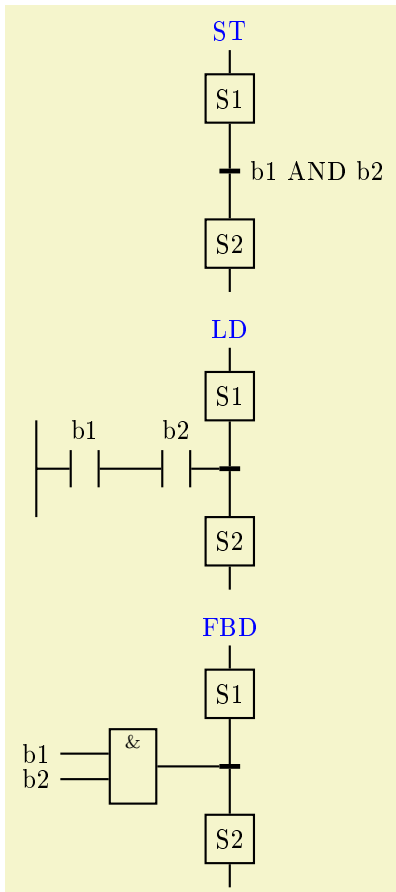
Each transition shall have an associated transition condition which is the result of the evaluation of a single Boolean expression. A transition condition which is always true shall be represented by the symbol 1 or the keyword TRUE.

A transition condition can be associated with a transition by one of the following means, as shown in example bellow<sup>6</sup>:

1. By placing the appropriate Boolean expression in the Structured Text (ST) language physically or logically adjacent to the vertical directed link.
2. By a ladder diagram network in the Ladder Diagram (LD) language physically or logically adjacent to the vertical directed link.
3. By a network in the Function Block Diagram (FBD) language, physically or logically adjacent to the vertical directed link.

<sup>5</sup>Note that, IEC 61131 does allow comments, but it is not clear where they are allowed.

<sup>6</sup>Textual representation of transition is not covered herein. Connector, though possible, is not covered herein.



```
\begin{tikzpicture}[circuit plc ladder, circuit logic
IEC, circuit plc sfc, thick,
y=1.3\tikzcircuitssizeunit, x=2.6\tikzcircuitssizeunit,
sfcaqw=2.6\tikzcircuitssizeunit]
\draw (0,0) node[above,blue]{ST}
to [sfctest={info=S1}] ++(0,-4)
to [sfctransition={name=t1,info=b1 AND b2}] ++(0,-2)
to [sfctest={info=S2}] ++(0,-4);
\pgftransformyshift{-16\tikzcircuitssizeunit}
\draw (0,0) node[above,blue]{LD}
to [sfctest={info=S1}] ++(0,-4)
to [sfctransition={name=t1}] ++(0,-2)
to [sfctest={info=S2}] ++(0,-4);
\draw (t1)
to [contact NO={info'=b2}] ++(-2,0)
to [contact NO={info'=b1}] ++(-2,0) ++(0,2) -- +(0,-4);
\pgftransformyshift{-16\tikzcircuitssizeunit}
\draw (0,0) node[above,blue]{FBD}
to [sfctest={info=S1}] ++(0,-4)
to [sfctransition={name=t1}] ++(0,-2)
to [sfctest={info=S2}] ++(0,-4);
\node[and gate={name=a1}] at ($(t1)-(2,0)$) {};
\draw (a1.input 1) -- ++(-1,0) node[left]{b1}
(a1.input 2) -- ++(-1,0) node[left]{b2}
(a1.output) -- (t1);
\end{tikzpicture}
```

For Ladder Diagram (LD) language typesetting refer to, for instance, package `tikz-ladder` and its documentation. For Function Block Diagram (FBD) language typesetting refer to TikZ-library `circuits.logic.IEC` in the TikZ manual. In this case, in order to conform TikZ-library `circuits.logic.IEC` with IEC 61 131-3 you need to place the following command in the document preamble:

```
\tikzset{every circuit symbol/.style={
logic gate inverted radius=0.8ex,
and gate IEC symbol = {$|char'|&$},
nand gate IEC symbol = {$|char'|&$},
or gate IEC symbol = {$>=1$},
nor gate IEC symbol = {$>=1$},
xor gate IEC symbol = {$=2k+1$},
xnor gate IEC symbol = {$=2k+1$},
not gate IEC symbol = {},
buffer gate IEC symbol = {}
}}
```

or, if the textual Boolean function names are preferred:

```
\tikzset{every circuit symbol/.style={
logic gate inverted radius=0.8ex,
and gate IEC symbol = {AND},
nand gate IEC symbol = {AND},
or gate IEC symbol = {OR},
nor gate IEC symbol = {OR},
xor gate IEC symbol = {XOR},
xnor gate IEC symbol = {XOR},
not gate IEC symbol = {},
buffer gate IEC symbol = {NOT}
}}
```

Note that, in this case, `buffer gate` is meant for NOT function (a block with the word “NOT” inside of it). The `not gate` produces an empty rectangle with a circle in the output to represent the negation.

Transitions are affected by the following keys:

`/tikz/every sfctransition`

(style, no value)

Set this style to configure the styling of transitions.

```
+ b1 \begin{tikzpicture}[circuit plc sfc,thick,y=1.3|tikzcircuitssizeunit,
every sfctransition/.style={red}]
\draw (0,0)
to [sfctransition={info=b1}] ++(0,-1);
\end{tikzpicture}
```

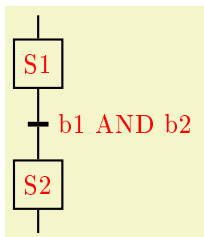
`/tikz/info=[(options)](angle):(text)` (no default)

This key is meant to place a Boolean expression in the Structured Text (ST) language at the *right* side of the transition to depict the transition condition. It has nearly the same effect as the `label` key, only the placement position is change and following style is used additionally and automatically:

`/tikz/every info` (style, no value)

Set this style to configure the styling of info labels. Since this key is *not* used with normal labels, it provides an easy way of changing the way info labels look without changing other labels.

The *(options)* and *(angle)* are passed directly to the `label` command.



```
\begin{tikzpicture}[circuit plc sfc,thick,y=1.3|tikzcircuitssizeunit,
every info/.style={red}]
\draw (0,0)
to [sfcstep={info=S1}] ++(0,-4)
to [sfctransition={info=b1 AND b2}] ++(0,-1)
to [sfcstep={info=S2}] ++(0,-4);
\end{tikzpicture}
```

Note that this key affects all `info` keys, including those used in steps. For a detailed discussion of the `label` option refer to the TikZ manual.

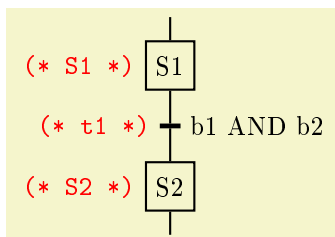
`/tikz/info'=[(options)](angle):(text)` (no default)

This key is meant to place a comment at the *left* side of the transition<sup>7</sup>. It has nearly the same effect as the `label` key, only the placement position is change and following style is used additionally and automatically:

`/tikz/every info'` (style, no value)

Set this style to configure the styling of info' labels. Since this key is *not* used with normal labels, it provides an easy way of changing the way info labels look without changing other labels.

The *(options)* and *(angle)* are passed directly to the `label` command.



```
\begin{tikzpicture}[circuit plc sfc,thick,y=1.3|tikzcircuitssizeunit,
every info'/.style={red,font=\ttfamily}]
\draw (0,0)
to [sfcstep={info=S1, info'={(* S1 *)}}] ++(0,-4)
to [sfctransition={info=b1 AND b2, info'={(* t1 *)}}] ++(0,-1)
to [sfcstep={info=S2, info'={(* S2 *)}}] ++(0,-4);
\end{tikzpicture}
```

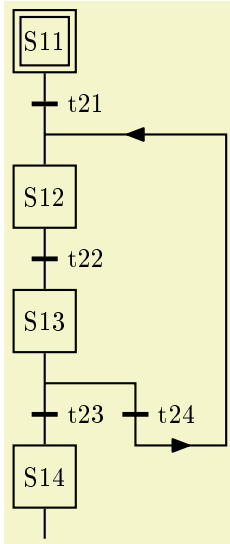
Note that this key affects all `info'` keys, including those used in steps. For a detailed discussion of the `label` option refer to the TikZ manual.

## 7 Flow direction

When necessary for clarity, an arrow can be used to indicate control flow. When this feature is used, the arrow shall be located in the middle of the line, or at least not at the line ends<sup>8</sup>.

<sup>7</sup>Note that, IEC 61 131 does allow comments, but it is not clear where they are allowed.

<sup>8</sup>IEC 61 131-3/2013 demands that “When this feature is used, the corresponding character shall be located between two “ characters, that is, in the character sequence “-<-” or “->-”...” indicating that the arrow shall not be in the end of a line.



```
\begin{tikzpicture}[circuit plc sfc,thick,circuit symbol unit=9pt,
x=1.2cm, y=1.3|tikzcircuitssizeunit]
\node[sfcstepi={info=S11},name=n0]{};
\draw (n0)
to [sfctransition={info=t21}] ++(0,-3) coordinate(n1)
to [sfcstep={info=S12}] ++(0,-4)
node [sfctransition={info=t22}]{}
to [sfcstep={info=S13}] ++(0,-4) coordinate(n2) -- ++(0,-1)
node [sfctransition={info=t23}]{}
to [sfcstep={info=S14}] ++(0,-4)
(n2) -| ++(1,-1)
node [sfctransition={info=t24}]{}
-- ++(0,-1) to [flow direction] +(1,0) |- ($n1)+(2,0)$) to [flow
direction] (n1);
\end{tikzpicture}
```

There are two symbols for indicating flow directions:

/tikz/flow direction:  $\longrightarrow$  and

/tikz/flow direction':  $\longleftarrow$

In other words, the apostrophe (') reverts the arrow direction.

The to command is used to place the arrow between two coordinates and the position of the arrow can be fine-tuned with pos key. Do not use pos=1, it makes impossible for the internal algorithm to establish the correct orientation.

If there is nothing beyond the next coordinates, pos behaves quit predictably:

```
\draw (0,0) to [flow direction={pos=0}] (0,-1);
```

```
\draw (0,0) to [flow direction={pos=0.05}] (0,-1);
```

```
\draw (0,0) to [flow direction={pos=0.5}] (0,-1);
```

```
\draw (0,0) to [flow direction={pos=0.95}] (0,-1);
```

```
\draw (0,0) to [flow direction={pos=1}] (0,-1);
```

where  $\updownarrow$  is one unit of length long and serves to indicate where the \draw command starts and ends. It points out that the pos establishes the arrow centre (not its tip as intuition might suggest).

Care should be taken when the next command is also a to because the next length will not be taken into account for the placement of the arrow. This can be misleading:

```
\draw (0,0) to [flow direction={pos=0}] ++(0,-1) to [sfcstepi={info=S1}] ++(0,-4);
```

```
\draw (0,0) to [flow direction={pos=0.05}] ++(0,-1) to [sfcstepi={info=S1}] ++(0,-4);
```

```
\draw (0,0) to [flow direction={pos=0.5}] ++(0,-1) to [sfcstepi={info=S1}] ++(0,-4);
```



```
\draw (0,0) to [flow direction={pos=0.95}] ++(0,-1) to [sfcstepi={info=S1}] ++(0,-4);
```

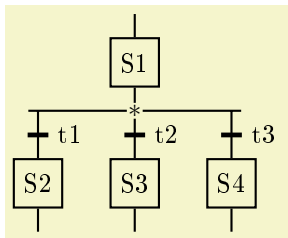


```
\draw (0,0) to [flow direction={pos=1}] ++(0,-1) to [sfcstepi={info=S1}] ++(0,-4);
```

Note that the distance considered by `pos` is not from  $(0,0)$  to top of the initial step block, but from  $(0,0)$  to  $(0,-1)$  which, in this case, is a point half the way from  $(0,0)$  to top of the step block. The arrow need to be placed near  $(0,-1)$ , but since `pos=1` turns the arrow around, the solution is to use `pos=0.95`.

## 8 Divergence of sequence with left to right priority

A selection between several sequences is represented by as many transition symbols, under the horizontal line, as there are different possible evolutions. The asterisk denotes left-to-right priority of transition evaluations. To place an asterisk, we use the symbol `sfcstar`:

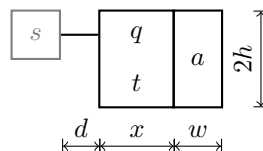


```
\begin{tikzpicture}[circuit plc sfc,thick,x=2.6|tikzcircuitssizeunit,
y=1.3|tikzcircuitssizeunit]
\draw (0,0)
to [sfcstep={info=S1,pos=0.4},sfcstar={name=divORC,pos=0.8}] ++(0,-5)
node [sfctransition={info=t2}]{}
to [sfcstep={info=S3}] ++(0,-4)
(divORC) -- ++(-2.2,0) ++(0.2,0) -- ++(0,-1)
node [sfctransition={info=t1}]{}
to [sfcstep={info=S2}] ++(0,-4)
(divORC) -- ++(2.2,0) ++(-0.2,0) -- ++(0,-1)
node [sfctransition={info=t3}]{}
to [sfcstep={info=S4}] ++(0,-4);
\end{tikzpicture}
```

The asterisk has to be placed with the special syntax `to` in order to interrupt the lines near it. Thus, the first `to` command places both the step `S1` and the asterisk. To leave enough room, the usual distance of four units of length were increased to five units of length. The step is placed two units of length below the line start ( $5 \cdot 0.4 = 2$ ) and the asterisk is placed one unit of length above the transition ( $5 - 5 \cdot 0.8 = 1$ ). See Section 10 for hints on how to set the unit of length to a sensible value.

## 9 Actions

One or more actions (actually, action blocks) can be associated with a step. Each association is done by the annotation<sup>9</sup> `sfcaction`. The first action associated with the step will be placed at the right side of the step and a line will be drawn connecting the action and the step. The distance between the step and the action is controlled by `label distance`. The default distance is  $2\text{tikzcircuitssizeunit}$ . Any further actions will be placed below the last action associated with the step. The action template is:



where:

`s` is the associated step;

<sup>9</sup>The `sfcaction` style is not an annotation any longer, but it was initially written as an annotation and behaves like one. The main difference from a real annotation is that there is not `sfcaction'` to place a mirrored action.



*q* is the action qualifier set by key `qualifier`. The style for all qualifiers can be set by `every sfcqualifier/.style`. Accordingly to the standard IEC 61131-3, the valid qualifier are: (none), N, R, S, L, D, P, SD, DS, SL, P1 and P0. No test is performed, however, to reinforce this rule and it is the user responsibility to cope with the standard in this respect;

*t* is the duration time (to be used with qualifiers: L, D, SD, DS, and SL) and set by key `time`. Again, no test is performed and any qualifier can be associated with a time, even if it is not in accordance to the standard implying in a syntax error. The style for all duration times can be set by `every sfctime/.style`. Remember that the symbol `#` shall be preceded by a `\`, e.g., to obtain `T#3s` type `T\#3s`;

*a* the action name set by key `info`. The style for all action names can be set by `every sfcactionname/.style`;

*d* the horizontal distance between the step and the action(s) associated with the step. This can be set globally using something like `every sfcaction/.style={label distance=1cm}` or individually placing a label distance specification on the desired `sfcaction` – this potentially breaks multiple actions alignment on the same step;

*x* the qualifier width set, usually globally, by key `sfcaqw`;

*w* the action name width set, usually globally, by key `sfcanw`; and

*h* the single line action height set, usually globally, by key `sfcah`. Actions with time association have a height of  $2h$  and this is done automatically.

Initially, the qualifier width, *x* in the figure, `sfcaqw` is set to zero because either qualifiers are not used (defaulted to “null qualifier”) or it needs to be set manually to match the biggest qualifier or qualifier duration. It is possible to set `sfcaqw` individually, but usually it is not desirable.

In the first example, for instance, the qualifier width `sfcaqw` is set to `8\tikzcircuitssizeunit` to leave room for the duration time of `T#3s400ms`, then three actions were associated with step I.

`/tikz/every sfcaction` (style, no value)

Set this style to configure the styling of action blocks. The horizontal distance between the step and the actions is controlled by `label distance`, here is place to adjust it globally.

`/tikz/every sfcqualifier` (style, no value)

Set this style to configure the styling of action qualifiers.

`/tikz/every sfctime` (style, no value)

Set this style to configure the styling of action qualifier time durations.

`/tikz/every sfcactionname` (style, no value)

Set this style to configure the styling of action names.

`/tikz/info=[<options>]<text>` (no default)

This key is meant to place the action name inside the action block. The following style is used additionally and automatically:

`/tikz/every info` (style, no value)

Set this style to configure the styling of info labels. Since this key is *not* used with normal labels, it provides an easy way of changing the way info labels look without changing other labels.

Note that this key affects all `info` keys, including those used in steps and transitions.

`/tikz/info'=[<options>]<text>` (no default)

This key is meant to place a comment at the *right* side of the action<sup>10</sup>. The following style is used additionally and automatically:

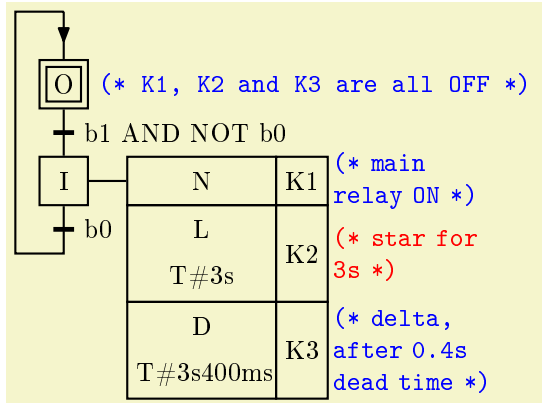
---

<sup>10</sup>Note that, IEC 61131 does allow comments, but it is not clear where they are allowed.

`/tikz/every info'`

(style, no value)

Set this style to configure the styling of info' labels. Since this key is *not* used with normal labels, it provides an easy way of changing the way info labels look without changing other labels.



```
\begin{tikzpicture}[circuit plc sfc,thick,
x=2.6\tikzcircuitssizeunit,
y=1.3\tikzcircuitssizeunit,
sfcqw=8\tikzcircuitssizeunit,
every info'/.style={blue,font=\ttfamily,
text width=2.2cm}]
\draw (0,0)
to [flow direction={pos=0.95}] ++(0,-1)
to [sfcstepi={info=0,info'=
[[text width=]right:(* K1{,} K2 and K3 are all OFF *)}]
++(0,-4)
node [sfctransition={info=b1 AND NOT b0}] {}
to [sfcstep={info=I,
sfcaction={info=K1,qualifier=N,
info'=(* main relay ON *)},
sfcaction={info=K2,qualifier=L,time=T|#3s,
info'={red}(* star for 3s *)},
sfcaction={info=K3,qualifier=D,time=T|#3s400ms,
info'=(* delta{,} after 0.4s dead time *)}]
] ++(0,-4)
node [sfctransition={info=b0}]{}
|- ++(-1,-1) |- (0,0);
\end{tikzpicture}
```

Note that this key affects all info' keys, including those used in steps and transitions. Also, observe that the commas in “K1, K2 and...” and “delta, after...” are surrounded by curly brackets. This is also necessary for colon (:) and equal sign (=). However, the colon in `[text width=]right:(* K1{,}...` it is not meant to be printed, it is a *angle* specification marker and such as it is not surrounded by curly brackets.

## 10 Design Guidance

This section brings some recommendations that reflect the way I produce SFC diagrams. It may or may not work for you. Feel free to e-mail me if you have better ideas.

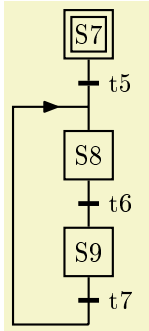
The first thing to consider is that, as the manual says, “TikZ ist kein Zeichenprogramm” which translates to “TikZ is not a drawing program”. You shall start with a draft of your chart and then codify it using TikZ. Generally, a handmade pencil sketch will do. In this draft, you shall use node names for future reference.

The second thing to consider is that `tikz-sfc` uses the `\tikzcircuitssizeunit` to keep all figures proportional. Therefore, when you consider any dimension related to symbol size it is good idea to set that dimension in respect to `\tikzcircuitssizeunit`, i.e., using `\tikzcircuitssizeunit` as the unit of length. The default value of `\tikzcircuitssizeunit` is 7 pt or approximately 2.46 mm and it can be set by the `circuit symbol unit` key among several other keys.

Steps and transitions are placed along a vertical line. An even space between them are desired. The `to` command is used to interrupt the line and place the symbol between two coordinates. Since a transition is only an horizontal line crossing the vertical line, it does not matter if the vertical line is interrupted or not. Thus, it is possible to employ the `node` to place a transition. In this way, the transition centre coordinates will be precisely known, but the centre coordinates of the step will be in the middle point of two known points. For a single sequence, it makes sense to place all transitions and then connect them with evenly spaced steps. This is rarely the case, however. Commonly, it is necessary to place an initial step in the begin of a line and dummy nodes (coordinate point) for loops.

Since steps are basically squares measuring  $2.6 \times 2.6$  `\tikzcircuitssizeunit`, one possible solution is to set the vertical unit of length to half of this distance. Thus, if two transitions are placed four units of length apart, it leaves one unit of length between each transition and the step between them. This is why you've been seen `y=1.3\tikzcircuitssizeunit` in the examples.

For instance:



```

\begin{tikzpicture}[circuit plc sfc,thick,x=1cm,
y=1.3|tikzccircuitssizeunit]
\node[sfcstepi={info=S7},name=n0]{};
\draw (n0)
to [sfctransition={info=t5}] ++(0,-3) coordinate(n1)
to [sfcstep={info=S8}] ++(0,-4)
node [sfctransition={info=t6}]{}
to [sfcstep={info=S9}] ++(0,-4)
node [sfctransition={info=t7}]{}
-- +(0,-1) coordinate(n2) (n1) to [flow direction'] ++(-1,0) |- (n2);
\end{tikzpicture}

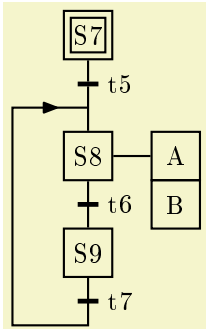
```

the initial step S7 is drawn with command `\node` and named `n0` for future reference. Note that `\node` requires an argument, in this case an empty text `{}`. Then, starting from `n0` (the initial step name for drawing purposes) a descending vertical line is drawn. Three units of length below the centre of `n0`, `n1` is placed. Between `n0` and `n1` the transition with receptivity `t5` is placed. Here is the beauty of the placement algorithm: the distance between `n0` centre and `n1` is 3 units, but the distance between `n0` border and `n1` is only 2 unit (the height of `n0` is 2 units, so 1 unit from the centre to the border); `t5` is placed between the border (not the centre) of `n0` and `n1`. This will certainly cause you troubles soon or later. Now, S8 is placed between `n1` and an unnamed point 4 units below `n1`. At this point, `t6` is place. All the same for S9 and `t7`. After placing `t7`, we move down one vertical unit and place `n2` (for future reference). Then restarting at `n1` we move left one horizontal unit of length and place an arrow pointing right (with `flow direction'`) between this location and `n1`. Finally, we move to `n2`, but not in a straight line, with the commando `|- (n2)`.

There are, of course, other ways of drawing this chart. For instance, the last line could be replaced by `|- +(-1,-1) -- ($n1)+(-1,0)$) to [flow direction] (n1)`; Thus, after placing `t7`, we move down one vertical unit and left one horizontal unit (`|- +(-1,-1)`); then move up to a point aligned with `n1`, one horizontal unit at its left computed by `($n1)+(-1,0)$`; then finally go to `n1` placing an arrow on the path (`flow direction`). In both cases, we do not have to know the distance between `t7` and `n1`, so additional steps and transitions can be inserted in the middle of the chart without the need to change the loop.

The distance between every feature, disregarding the line width, is one unit of length.

Suppose now that we need to associate actions with S8. The first one will be placed at S8 right side, but the second will be aligned with `t6`. Since `t6` is a simple and, more important, small expression, it causes no harm.

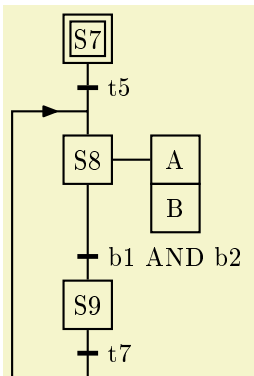


```

\begin{tikzpicture}[circuit plc sfc,thick,x=1cm,
y=1.3|tikzccircuitssizeunit]
\node[sfcstepi={info=S7},name=n0]{};
\draw (n0)
to [sfctransition={info=t5}] ++(0,-3) coordinate(n1)
to [sfcstep={info=S8,sfaction={info=A},sfaction={info=B}}] ++(0,-4)
node [sfctransition={info=t6}]{}
to [sfcstep={info=S9}] ++(0,-4)
node [sfctransition={info=t7}]{}
|- +(-1,-1) -- ($n1)+(-1,0)$) to [flow direction] (n1);
\end{tikzpicture}

```

but what if `t6` were a long expression like, e.g., `b1 AND b2`? In this case, a filler is needed to create a dead space between S8 and the following transition.



```

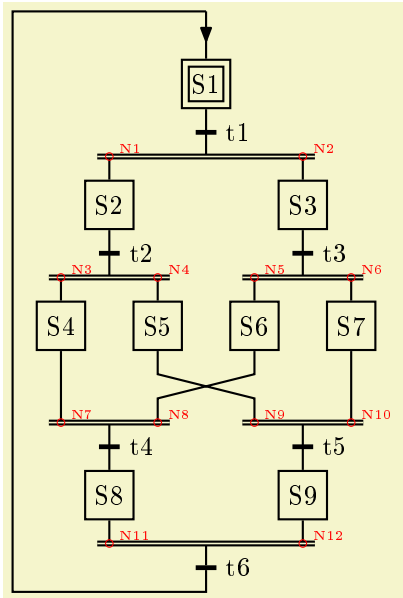
\begin{tikzpicture}[circuit plc sfc,thick,x=1cm,
y=1.3|tikzccircuitssizeunit]
\node[sfcstepi={info=S7},name=n0]{};
\draw (n0)
to [sfctransition={info=t5}] ++(0,-3) coordinate(n1)
to [sfcstep={info=S8,sfaction={info=A},sfaction={info=B}}] ++(0,-4) -- ++(0,-2)
node [sfctransition={info=b1 AND b2}]{}
to [sfcstep={info=S9}] ++(0,-4)
node [sfctransition={info=t7}]{}
|- +(-1,-1) -- ($n1)+(-1,0)$) to [flow direction] (n1);
\end{tikzpicture}

```

The filler `--+(0,-2)` is placed just after the second coordinate used to draw S8. The rule of thumb for fillers is to allow *two* vertical units for every action exciding the first one, *four* if the action involves duration (key `time` is used). If the first action also involves duration, add *two* vertical units.

## 10.1 Divergence and convergence

Divergence and convergence can be exclusive (single line) or simultaneous (double line). They both can be done using the `\coordinate` command to place points during the construction of the path, then using those points to draw the divergence and convergence lines. In the following example, the second `\draw` command extends every horizontal line by half unit of length in both directions.



```
\begin{tikzpicture}[circuit plc sfc,thick,
x=1.3|tikzcircuitssizeunit,
y=1.3|tikzcircuitssizeunit]
\draw (0,0)
to [flow direction={pos=0.95}] ++(0,-1)
to [sfcstepi={info=S1}] ++(0,-4)
node[sfctransition={info=t1}]{} -- ++(0,-1)
++(4,0) coordinate(N2) ++(-8,0) coordinate(N1)
to [sfcstep={info=S2}] ++(0,-4)
node[sfctransition={info=t2}]{} -- ++(0,-1)
++(2,0) coordinate(N4) ++(-4,0) coordinate(N3)
to [sfcstep={info=S4}] ++(0,-4) -- ++(0,-2) coordinate(N7)
(N4) to [sfcstep={info=S5}] ++(0,-4) -- ++(4,-1)
-- ++(0,-1) coordinate(N9)
++(-6,0) -- ++(0,-1) node[sfctransition={info=t4}]{}
to [sfcstep={info=S8}] ++(0,-4) coordinate(N11)
(N2) to [sfcstep={info=S3}] ++(0,-4)
node[sfctransition={info=t3}]{} -- ++(0,-1)
++(2,0) coordinate(N6) ++(-4,0) coordinate(N5)
to [sfcstep={info=S6}] ++(0,-4)
--++(-4,-1) --++(0,-1) coordinate(N8)
(N6) to [sfcstep={info=S7}] ++(0,-4) -- ++(0,-2) coordinate(N10)
++(-2,0) -- ++(0,-1) node[sfctransition={info=t5}]{}
to [sfcstep={info=S9}] ++(0,-4) coordinate(N12)
++(-4,0) to[sfctransition={info=t6}] ++(0,-2)
-- ++(-8,0) |- (0,0);
\draw[double=graphicbackground]
(N1) ++(-0.5,0) -- (N2) -- ++(0.5,0)
(N3) ++(-0.5,0) -- (N4) -- ++(0.5,0)
(N5) ++(-0.5,0) -- (N6) -- ++(0.5,0)
(N7) ++(-0.5,0) -- (N8) -- ++(0.5,0)
(N9) ++(-0.5,0) -- (N10) -- ++(0.5,0)
(N11) ++(-0.5,0) -- (N12) -- ++(0.5,0);
\foreach \x in {1,...,12}
\node[label={{label distance=-4pt,above
right, red,font=\tiny}N\x},
circle,inner sep=1pt,draw=red,thin] at (N\x) {};
\end{tikzpicture}
```

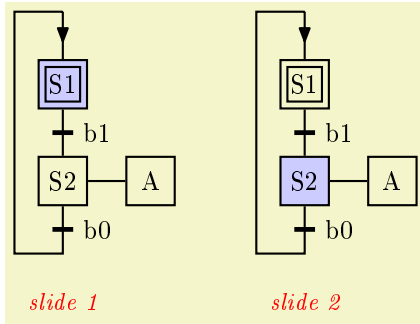
The `\foreach` command is used to print small circles marking the nodes with node names in tiny red letters and those are not part of the chart, yet they do nicely for debug purposes. By the way, `graphicbackground` is the background colour of the examples in the document.

## 10.2 Beamer presentation with overlay

Perhaps the most important feature of BEAMER is its capacity of creating a series of slides based in one slide for, e.g., showing one concept step by step. It is called overlay. Unfortunately, BEAMER overlay and TikZ present some compatibilities issues when overlay macros are typed in the options list of a TikZ command. This can be solved setting two TikZ keys that takes advantage of the fact that `\pgfkeysalso` doesn't change the path.

```
\tikzset{ % alt and visible (overlay)
alt/.code args={<#1>#2#3}{%
|alt<#1>{\pgfkeysalso{#2}}{\pgfkeysalso{#3}}
},
visible/.code args={<#1>#2}{%
|alt<#1>{\pgfkeysalso{#2}}{}
}
}
```

Once these keys are in place, they can be used instead of `\visible` and `\alt` with a syntax slightly different. The following example creates two slides, one with S1 active and the other with S2 active (activations are shown by filling the steps in light blue). Below, the result is shown side by side:



```
\begin{tikzpicture}[circuit plc sfc,thick,
x=2.6\tikzcircuitssizeunit,
y=1.3\tikzcircuitssizeunit]
\draw
(0,0) to [flow direction={pos=0.95}] ++(0,-1)
to [sfcstepi={info=S1,
visible={<1>{fill=blue!20}}}] ++(0,-4)
node [sfctransition={info=b1}]{}
to [sfcstep={info=S2,sfcaction={info=A},
visible={<2>{fill=blue!20}}}] ++(0,-4)
node [sfctransition={info=b0}]{}
|- ++(-1,-1) |- (0,0);
\end{tikzpicture}
```

Thus, you can explain that, initially, S1 is active, pressing the push button b1 the transition fires deactivating S1 and activating S2 that turns on motor A. The S1 light blue background is visible in slide 1, but it is not in slide 2; conversely, S2 background is visible only on slide 2. One cool trick is that you can press push button b0 returning to slide 1.

### 10.3 Adjusting the diagram size

The diagram size can be adjusted in two ways: fully or just the distance between the elements.

Typing the key `circuit symbol unit= $\langle dimension \rangle$`  in TikZ options before setting anything using `\tikzcircuitssizeunit` will adjust everything but the font size and line width.

To adjust the horizontal distance between the symbols, set the  $x$  unit of length with `x= $\langle dimension \rangle$` . For vertical distances use `y= $\langle dimension \rangle$` . The default procedure adopted in this document is to set `y=1.3\tikzcircuitssizeunit` in order to get a nice vertical distance between the elements. Usually, only the fillers will need to be rethought after a change in  $y$  and it can be done multiply the old filler distance by the ratio between the old and new values of  $y$ .

## 11 Known Issues

The user interface is a bit inconsistent: texts are treated differently depending on where they should appear. Normally, `info` and `info'` keys are used to place text on the chart, but sometimes actions need one or two extra text (qualifier and time duration) and it is done by other keys, namely, `qualifier` and `time`. The problem is that these keys do not accept optional arguments like `info` and `info'`. It is possible to modify those keys to deal with optional arguments, but there is price to pay.

The keys `info` and `info'`, largely employ to place text, actually uses the `label` interface. This is great because you can pass an optional list of configurations. The trade-off is that you have to be careful when writing the text, particularly you need to surround the commas, colons<sup>11</sup> and equal signs in curly brackets. This is due to `\pgfkeys`.

If you want the report a bug or have any suggestion, please feel free to send me an e-mail. Contact details are in the first page. Every feedback is important.

## 12 Final Remarks

This package has been tested and used for more than three years, so I do believe it is mature by now and I decided to share it. On the other hand, I was the only person who used it, therefore idiosyncrasies were not detected.

Any comments, suggestions, and feedbacks are welcomed. I will do my best to answer as soon as possible. My contact e-mail is in the first page.

It should be great if someone with experience in writing TikZ libraries could have a look in the code and point out error or improvements to be made.

<sup>11</sup>If you literally mean “:” and not a  $\langle angle \rangle$  specification.

## Appendix A – Beamer example of delta-star motor starter with dead time in SFC

In this example, the animation is done in two layers: the chart is drawn on foreground with step, transitions, actions and animation (filling and colouring); on the background, circles (actually, ellipses since  $x$  and  $y$  scales are different) are drawn in light red to highlight the activated transition. In order to place the ellipses in the precise position, every transition is named.

At first, only the SFC is drawn and the example of Section 2 is used as starting point. Since the fonts used in the beamer presentation are different in size, it is necessary to adjust the chart size. In this particular, `sfcq` was increased to `9\tikzcircuitssizeunit`. When the chart is done, the animation part is made adding the `visible` key in the foreground and the `\visible` macro in the background.

Macro `\overlaynumber`, placed in the frame title, is used in the development phase to show the slide number in a sequence (animation). It helps the development of animated slides because you can see the slide number to be used with `visible` and `alt` keys or `\visible` and `\alt` macros.

After the L<sup>A</sup>T<sub>E</sub>X code, the slides were included for reference. They shall be found as a separated PDF file in the same folder you have found this document.

```
%% This is file 'BeamerAnimation.tex'
%% Version: 1.0.1
%% Version date: 2018-15-12
%%
%% Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br
%%
%% This package can be redistributed and/or modified under the terms
%% of the LaTeX Project Public License distributed from CTAN
%% archives in directory macros/latex/base/lppl.txt; either
%% version 1 of the License, or (at your option) any later version,
%% with 'The Package' referring to the software 'tikzlibrarycircuits.plc.sfc.code.tex'
%% and its accompanying documentation and 'The Copyright Holder' referring to
%% the person Luis Paulo Laus.
%%
%% IMPORTANT NOTICE:
%%
%% For error reports, comments or suggestions in case of UNCHANGED
%% versions send mail to:
%% laus@utfpr.edu.br
%%
%%

\documentclass{beamer}
\usepackage{tikz,units}
\usetikzlibrary{backgrounds, circuits.plc.sfc}

\makeatletter
\newcommand*{\overlaynumber}{\number\beamer@slideinframe}
\makeatother

\tikzset{ % alt and visible (overlay)
  alt/.code args={<#1>#2#3}{%
    \alt<#1>{\pgfkeysalso{#2}}{\pgfkeysalso{#3}}
  },
  visible/.code args={<#1>#2}{%
    \alt<#1>{\pgfkeysalso{#2}}{}
  }
}

\colorlet{LBlue}{blue!20}
\colorlet{LRed}{red!30}

\begin{document}

\begin{frame}{Delta-star motor starter with dead time in SFC: \overlaynumber{}}
\begin{columns}[c]

\column{0.4\textwidth}

\noindent \begin{center}
\begin{tikzpicture}[circuit plc sfc,thick,x=2.6\tikzcircuitssizeunit,
y=1.3\tikzcircuitssizeunit,sfcq=9\tikzcircuitssizeunit]

```

```

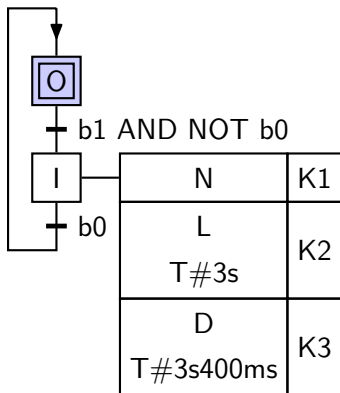
\draw (0,0)
  to [flow direction={pos=0.95}] ++(0,-1)
  to [sfcstepi={info=O, visible={<1,2,7>{fill=LBlue}}}] ++(0,-4)
  node [sfctransition={info=b1 AND NOT b0},name=t1,visible={<2>{red}}] {}
  to [sfcstep={info=I, visible={<3-6>{fill=LBlue}},
    sfcaction={info=K1, qualifier=N, visible={<3-6>{fill=LBlue}}},
    sfcaction={info=K2, qualifier=L, time=T\#3s, visible={<3>{fill=LBlue}}},
    sfcaction={info=K3, qualifier=D, time=T\#3s400ms, visible={<5-6>{fill=LBlue}}}] ++(0,-4)
  node [sfctransition={info=b0},name=t2,visible={<6>{red}}] {}
  |- ++(-1,-1) |- (0,0);
\begin{pgfonlayer}{background}
\visible<2>{
  \draw[fill=LRed,LRed](t1) circle (0.4);
}
\visible<6>{
  \draw[fill=LRed,LRed](t2) circle (0.4);
}
\end{pgfonlayer}
\end{tikzpicture}
\par\end{center}
\column{0.6\textwidth}
Delta-star with dead time:
\begin{itemize}[<+>]
\item initially, the motor is off
\item b1 is pressed
\item K1 and K2 are activated — star
\item after $\unit[3]{s}$, K2 is deactivated — dead time
\item after $\unit[0.4]{s}$, K3 is activated — delta
\item b0 is pressed
\item motor is off
\end{itemize}
\end{columns}
\noindent \begin{center}
\par\bigskip
Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br
\end{center}
\end{frame}
\end{document}

```

# Delta-star motor starter with dead time in SFC: 1

Delta-star with dead time:

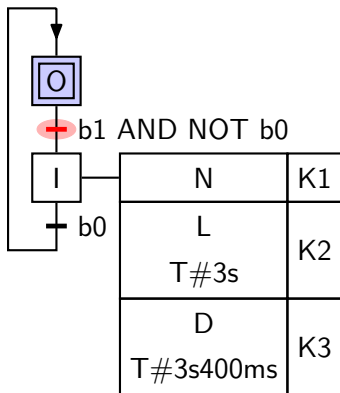
- ▶ initially, the motor is off



Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br



## Delta-star motor starter with dead time in SFC: 2

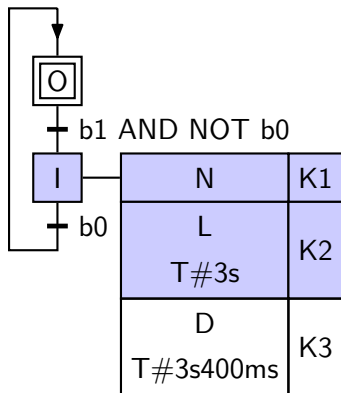


Delta-star with dead time:

- ▶ initially, the motor is off
- ▶ b1 is pressed

Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br

## Delta-star motor starter with dead time in SFC: 3

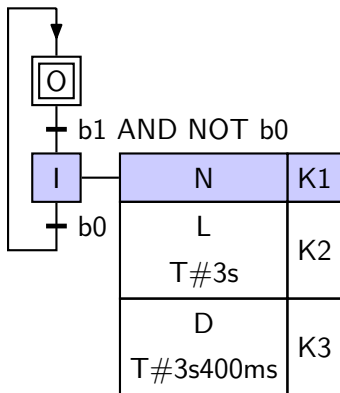


Delta-star with dead time:

- ▶ initially, the motor is off
- ▶ b1 is pressed
- ▶ K1 and K2 are activated – star

Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br

## Delta-star motor starter with dead time in SFC: 4

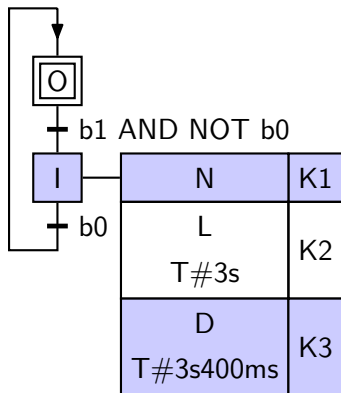


Delta-star with dead time:

- ▶ initially, the motor is off
- ▶ b1 is pressed
- ▶ K1 and K2 are activated – star
- ▶ after 3 s, K2 is deactivated – dead time

Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br

## Delta-star motor starter with dead time in SFC: 5

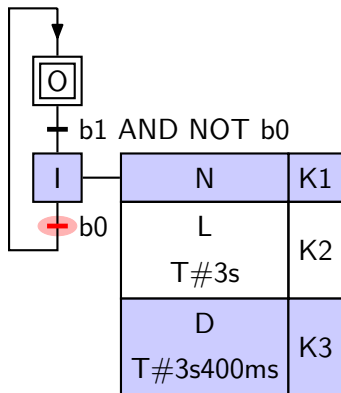


Delta-star with dead time:

- ▶ initially, the motor is off
- ▶ b1 is pressed
- ▶ K1 and K2 are activated – star
- ▶ after 3 s, K2 is deactivated – dead time
- ▶ after 0.4 s, K3 is activated – delta

Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br

## Delta-star motor starter with dead time in SFC: 6

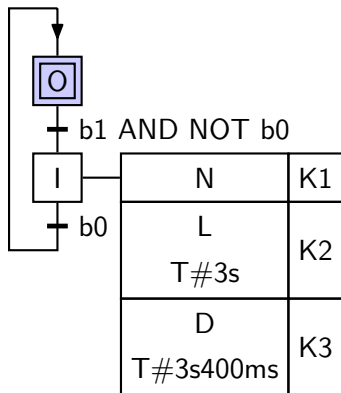


Delta-star with dead time:

- ▶ initially, the motor is off
- ▶ b1 is pressed
- ▶ K1 and K2 are activated – star
- ▶ after 3 s, K2 is deactivated – dead time
- ▶ after 0.4 s, K3 is activated – delta
- ▶ b0 is pressed

Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br

## Delta-star motor starter with dead time in SFC: 7



Delta-star with dead time:

- ▶ initially, the motor is off
- ▶ b1 is pressed
- ▶ K1 and K2 are activated – star
- ▶ after 3 s, K2 is deactivated – dead time
- ▶ after 0.4 s, K3 is activated – delta
- ▶ b0 is pressed
- ▶ motor is off

Copyright (C) 2018 by Luis Paulo Laus, laus@utfpr.edu.br