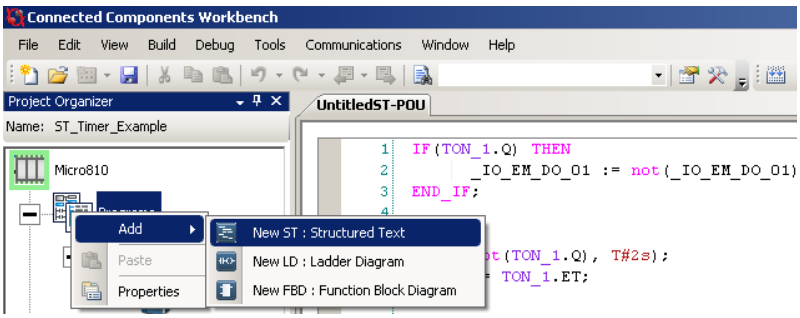
Structured Text Programs

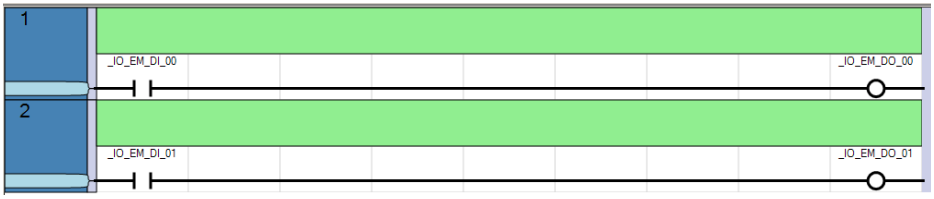
Another style of programming is called *Structure Text*. This is similar to writing a Pascal program (which is very similar to C)

When you start your program, select *New ST: Structured Text*



Basic I/O:

Ladder Logic:

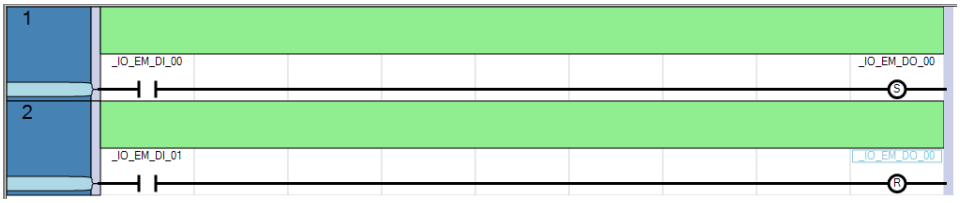


Structured Text:

```
_IO_EM_DO_00 := _IO_EM_DI_00;
_IO_EM_DO_01 := _IO_EM_DI_01;
```

SR Flip Flop:

Ladder Logic:

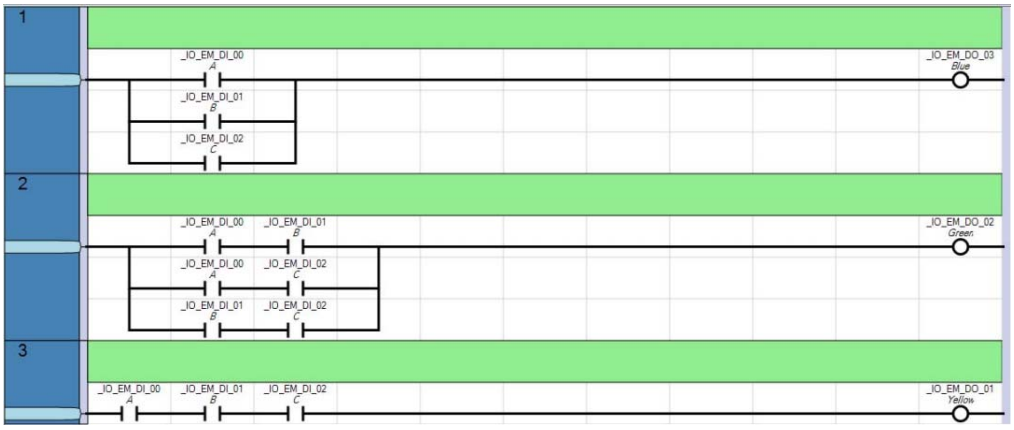


Structured Text:

```
IF (_IO_EM_DI_00) THEN
    _IO_EM_DO_00 := TRUE;
ELSEIF (_IO_EM_DI_01) THEN
    _IO_EM_DO_00 := FALSE;
END_IF;
```

Count How Many Buttons are Pressed:

Ladder Logic:



Combinational Logic:

```
_IO_EM_DO_03 := _IO_EM_DI_00 or _IO_EM_DI_01 or _IO_EM_DI_02;
_IO_EM_DO_02 := (_IO_EM_DI_00 and _IO_EM_DI_01) or
                (_IO_EM_DI_00 and _IO_EM_DI_02) or
                (_IO_EM_DI_01 and _IO_EM_DI_02);
_IO_EM_DO_01 := _IO_EM_DI_00 and _IO_EM_DI_01 and _IO_EM_DI_00;
```

Another Solution (if and case statements)

```
Count := 0;
IF (_IO_EM_DI_00)
    THEN Count := Count + 1;
    END_IF;
IF (_IO_EM_DI_01)
    THEN Count := Count + 1;
    END_IF;
IF (_IO_EM_DI_02)
    THEN Count := Count + 1;
    END_IF;
IF (_IO_EM_DI_03)
    THEN Count := Count + 1;
    END_IF;

CASE ( Count ) OF
   1:
      _IO_EM_DO_03 := FALSE;
      _IO_EM_DO_02 := FALSE;
      _IO_EM_DO_01 := FALSE;
      _IO_EM_DO_00 := TRUE;
   2:
      _IO_EM_DO_03 := FALSE;
      _IO_EM_DO_02 := FALSE;
      _IO_EM_DO_01 := TRUE;
      _IO_EM_DO_00 := FALSE;
   3:
      _IO_EM_DO_03 := FALSE;
      _IO_EM_DO_02 := TRUE;
```

```
         _IO_EM_DO_01 := FALSE;
         _IO_EM_DO_00 := FALSE;
      ELSE
         _IO_EM_DO_03 := TRUE;
         _IO_EM_DO_02 := FALSE;
         _IO_EM_DO_01 := FALSE;
         _IO_EM_DO_00 := FALSE;
      END_CASE;
```

Timers (sort of)

Toggle every 3.6 seconds (?)

```
Count := Count + 1;

if (Count > 1000) THEN
   Count := 0;
   END_IF;

IF (Count = 500) THEN
   _IO_EM_DO_02 := not(_IO_EM_DO_02);
   END_IF;
```

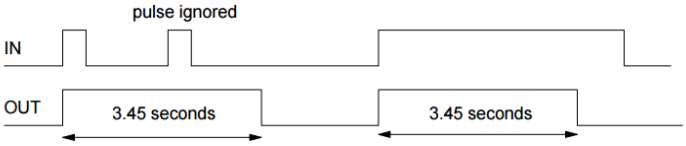
Variation

```
IF (_IO_EM_DI_00) THEN
   Operating_Mode := 1;  (* red light *)
ELSIF (_IO_EM_DI_01) THEN
   Operating_Mode := 2;  (* flashing yellow *)
ELSIF (_IO_EM_DI_02) THEN
   Operating_Mode := 3;  (* normal G - Y - R *)
END_IF;

if (Operating_Mode = 1) THEN
   _IO_EM_DO_00 := FALSE;
   _IO_EM_DO_01 := FALSE;
   _IO_EM_DO_02 := FALSE;
   _IO_EM_DO_03 := TRUE;
   END_IF;


if (Operating_Mode = 2) THEN
   TimeReal := TimeReal + 0.0036;
   if(TimeReal > 1.00) THEN
  _IO_EM_DO_01 := NOT(_IO_EM_DO_01);
  TimeReal := 0.0;
  END_IF;

   _IO_EM_DO_00 := FALSE;
   _IO_EM_DO_02 := FALSE;
   _IO_EM_DO_03 := FALSE;


   END_IF;
```

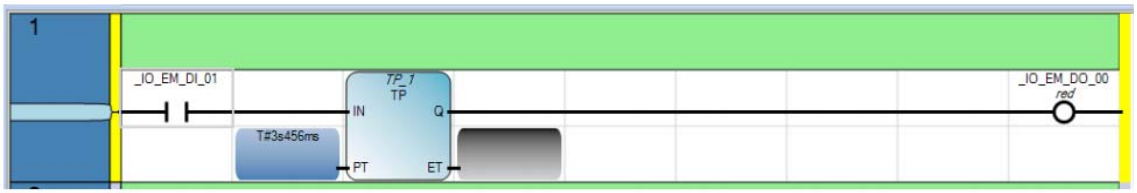
Timer Blocks

TP: Tiper Pulse:

Functionality:



Ladder Diagram:

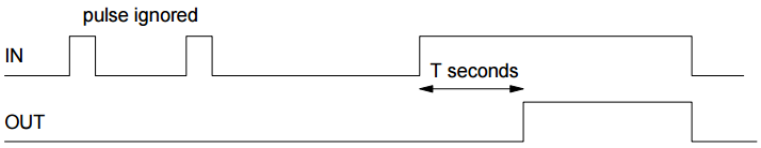


Structured Text:

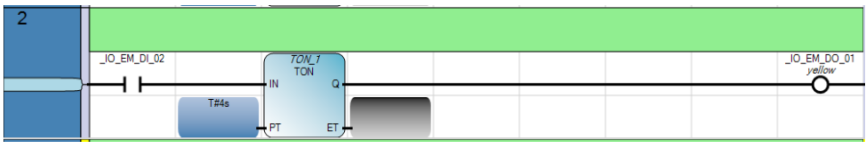
```
TP_1(_IO_EM_DI_01, T#3s456ms);
   Timer1_Time := TP_1.ET;
   _IO_EM_DO_00 := TP_1.Q;
```

TON: On Timer

Functionality: Output only if the input is held on for 4 seconds



Ladder Diagram:

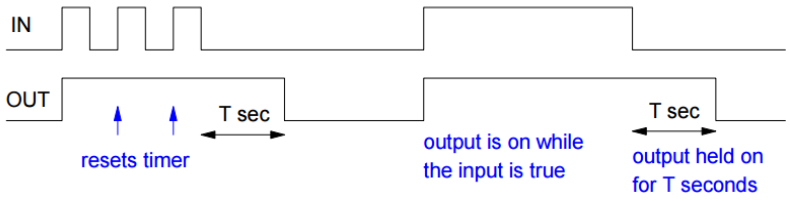


Structured Text:

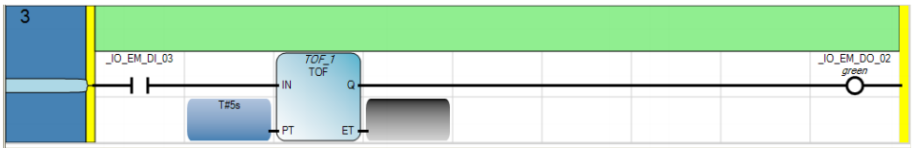
```
TON_1(_IO_EM_DI_02, T#4s);
  Timer1_Time := TON_1.ET;
  _IO_EM_DO_01 := TON_1.Q;
```

TOF: Off Timer

Functionality: Hold the output true for 5 seconds after the button is released



Ladder Diagram

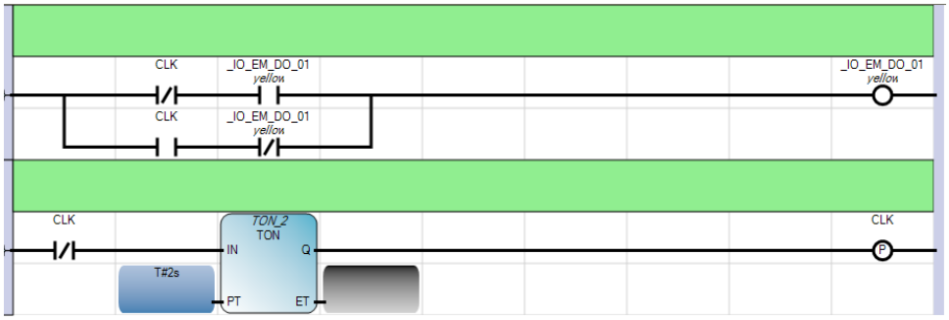


Structured Text:

```
TOF_1(_IO_EM_DI_03, T#5s);
   Timer1_Time := TOF_1.ET;
   _IO_EM_DO_02 := TOF_1.Q;
```

Toggle the Yellow Light Every 2 Seconds

Ladder Diagram



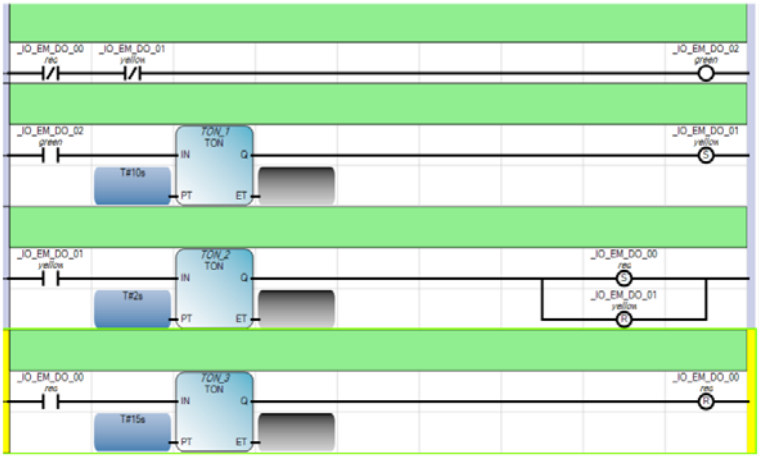
Structured Text:

```
IF(TON_1.Q) THEN
   _IO_EM_DO_01 := not(_IO_EM_DO_01);
   END_IF;
TON_1(not(TON_1.Q), T#2s);
Time1 := TON_1.ET;
```

note: Time1 is a variable type TIME

Green - Yellow - Red Stoplight

Ladder Diagram:



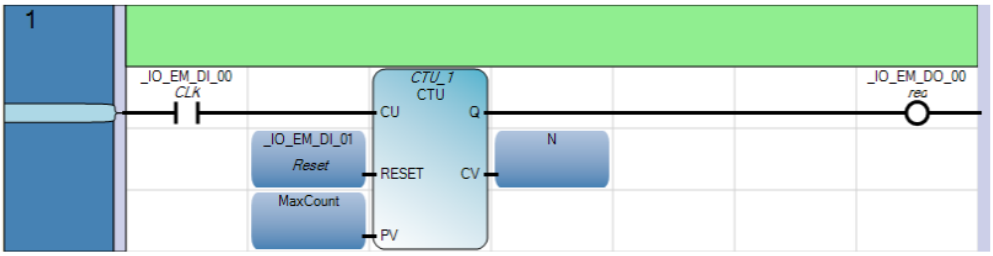
Structured Text:

```
_IO_EM_DO_02 := ( not(_IO_EM_DO_00) and not(_IO_EM_DO_01) );


TON_1(_IO_EM_DO_02, T#5s);
   Time1 := TON_1.ET;
   IF(TON_1.Q) THEN
      _IO_EM_DO_01 := TRUE;
      END_IF;

TON_2(_IO_EM_DO_01, T#1s);
   Time2 := TON_2.ET;
   IF(TON_2.Q) THEN
      _IO_EM_DO_00 := TRUE;
      _IO_EM_DO_01 := FALSE;
      END_IF;

TON_3(_IO_EM_DO_00, T#3s);
   Time3 := TON_3.ET;
   IF(TON_3.Q) THEN
      _IO_EM_DO_00 := FALSE;
      END_IF;
```

Counter Blocks

CTU: Up Counter

Ladder Diagram

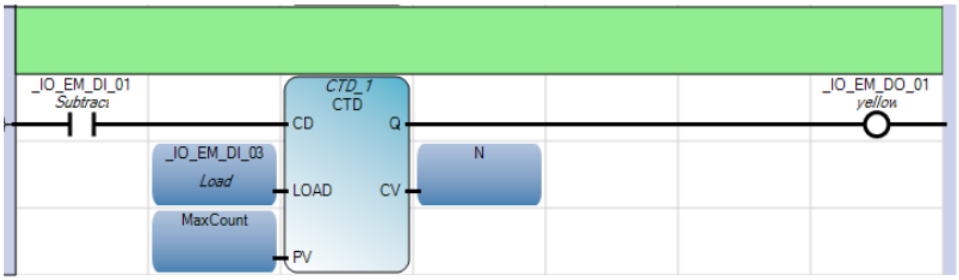


Structured Text Code:

```
(*         CU         RESET          PV     *)
CTU_1(_IO_EM_DI_00,_IO_EM_DI_01,  10);
   _IO_EM_DO_00 := CTU_1.Q;
   Count := CTU_1.CV;
```

CTD: Down Counter

Ladder Diagram

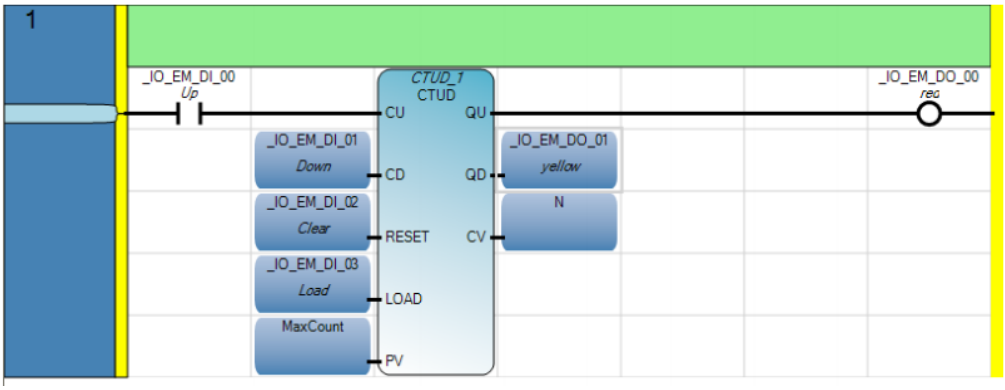


Structured Text Code:

```
(*         CD            LOAD       PV     *)
CTD_1(_IO_EM_DI_01, _IO_EM_DI_03, 10);
   _IO_EM_DO_01 := CTD_1.Q;
   Count := CTD_1.CV;
```

CTUD: Up Down Counter

Ladder Diagram



Structured Text:

```
(*          CU           CD           RESET          LOAD          PV      *)
 CTUD_1(_IO_EM_DI_00, _IO_EM_DI_01, _IO_EM_DI_02, _IO_EM_DI_03,  10);
    _IO_EM_DO_00 := CTUD_1.QU;
    _IO_EM_DO_01 := CTUD_1.QD;
    Count := CTUD_1.CV;
```

Ring Counter

Count 0 .. 9 and repeat

```
(*      Up            Clear     #    *)
CTU_1(_IO_EM_DI_00,  CTU_1.Q,  10);
   Count1 := CTU_1.CV;
```

Case Statement:

Display different light patters based upon the count value (0 .. 9 and repeat)

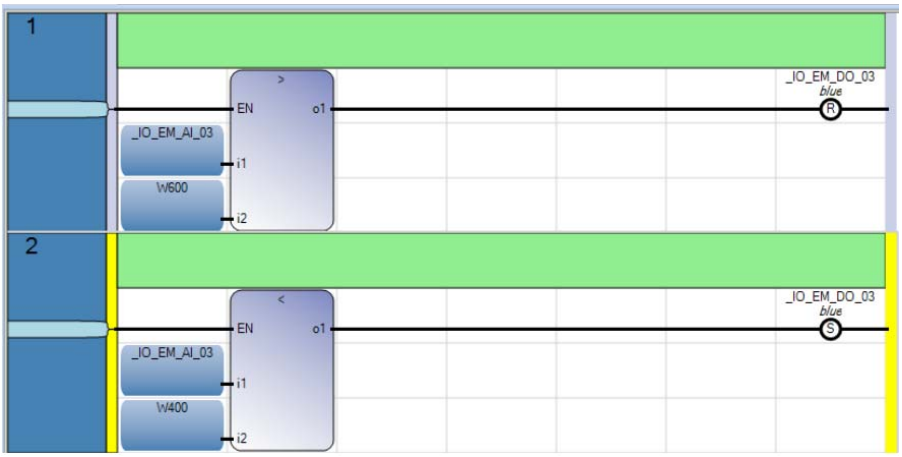
```
(*      Up            Clear    #   *)
CTU_1(_IO_EM_DI_00,  CTU_1.Q,  10);
   Count1 := CTU_1.CV;

CASE (Count1) OF
0:  (* red light *)
   _IO_EM_DO_00 := TRUE;
   _IO_EM_DO_01 := FALSE;
   _IO_EM_DO_02 := FALSE;
   _IO_EM_DO_03 := FALSE;
1:  (* yellow light *)
   _IO_EM_DO_00 := FALSE;
   _IO_EM_DO_01 := TRUE;
   _IO_EM_DO_02 := FALSE;
   _IO_EM_DO_03 := FALSE;
2:  (* green light *)
   _IO_EM_DO_00 := FALSE;
   _IO_EM_DO_01 := FALSE;
   _IO_EM_DO_02 := TRUE;
   _IO_EM_DO_03 := FALSE;
3:  (* blue light *)
   _IO_EM_DO_00 := FALSE;
   _IO_EM_DO_01 := FALSE;
   _IO_EM_DO_02 := FALSE;
   _IO_EM_DO_03 := TRUE;
ELSE (* lights off *)
   _IO_EM_DO_00 := FALSE;
   _IO_EM_DO_01 := FALSE;
   _IO_EM_DO_02 := FALSE;
   _IO_EM_DO_03 := FALSE;
END_CASE;
```

Analog Inputs

Function:

- Turn off relay 3 when the voltage (AI3) is more than 6.00V
- Turn on relay 3 when the voltage (AI3) is less than 4.00V

Ladder Diagram



Structured Text:

```
Analog3 := _IO_EM_AI_03;

IF(Analog3 > 600) THEN
    _IO_EM_DO_03 := FALSE;
    END_IF;

IF(Analog3 < 400) THEN
   _IO_EM_DO_03 := TRUE;
   END_IF;
```

note: Analog3 is type *Word*