SCT
STATE CONFIGURABLE TIMER

EXTERNAL USE

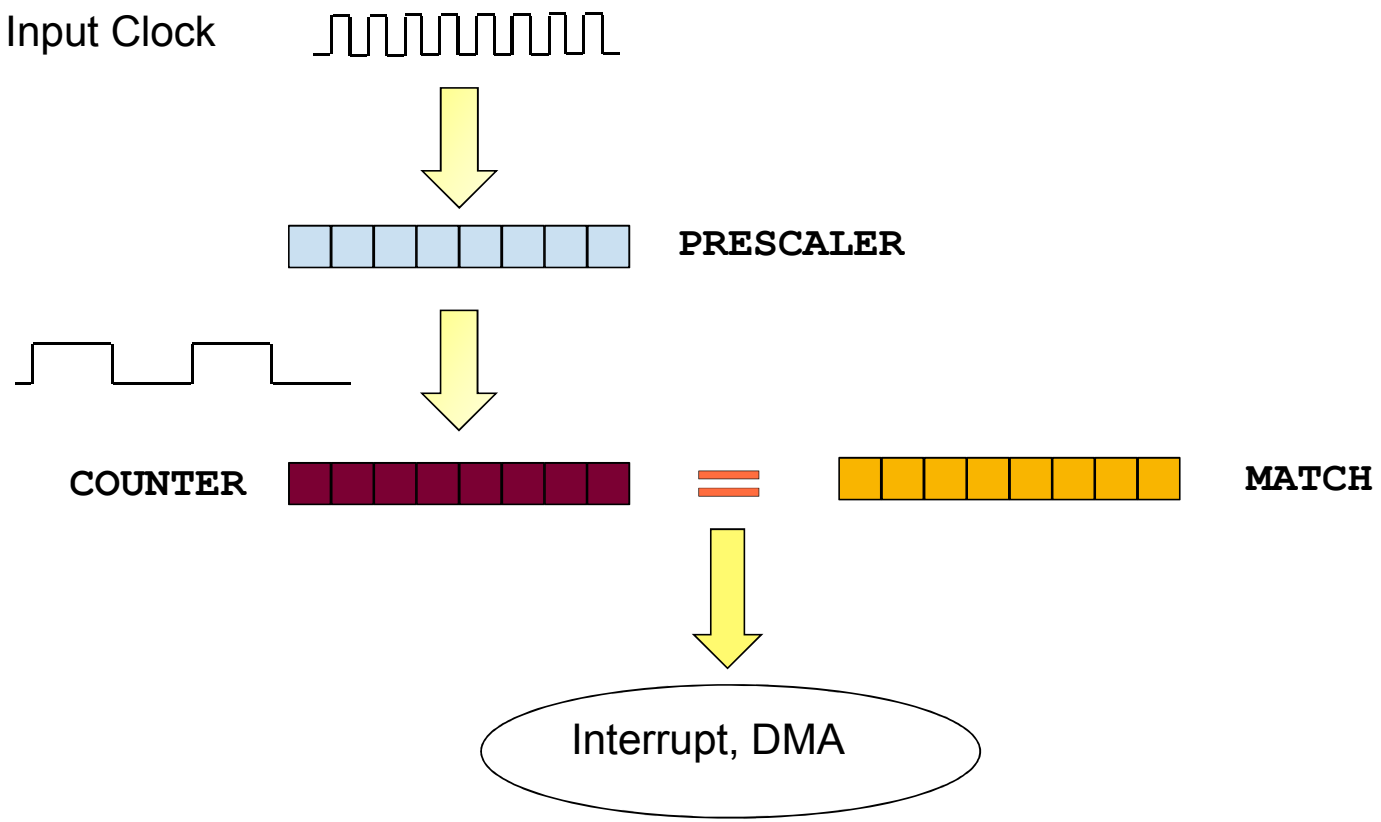
SECURE CONNECTIONS
FOR A SMARTER WORLD

Agenda

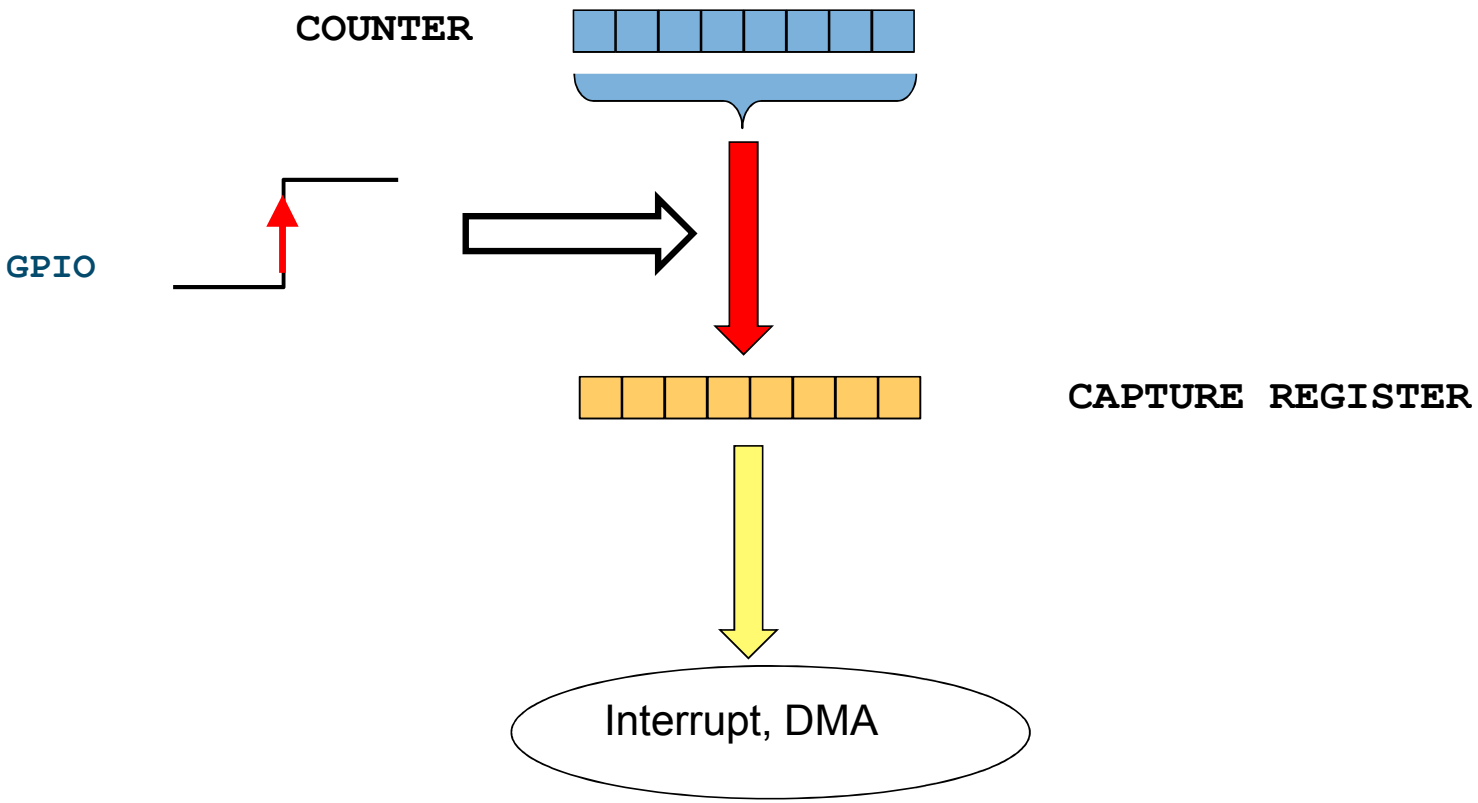
- Timer/State machine basics
- SCT introduction
- SCT availability
- SCT tools & resources
- SCT application analysis

TIMER/STATE MACHINE BASICS

Basics of Timers – match (compare) function



Basics of Timers – capture function

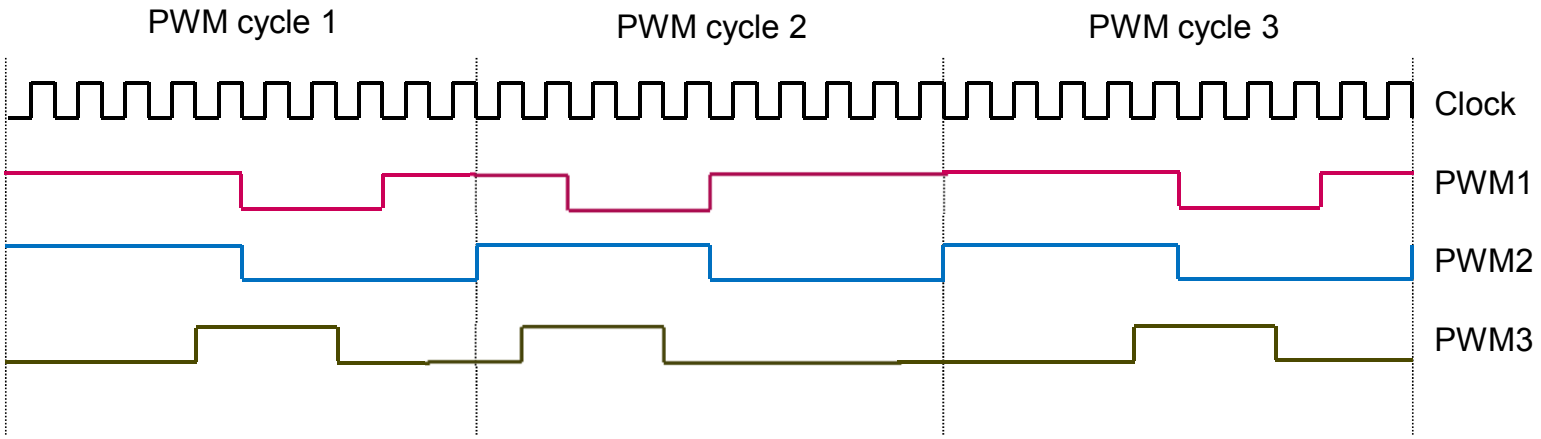


Timer basics – typical features

- Up, Down, Up-Down counting
  - Generate an interrupt or DMA request on events
- Operations on match event:
  - Continue counting or stop the counter
  - Reset (limit) counter
  - Set or clear an assigned GPIO signal
- Operations on capture event:
  - Take a counter „snapshot“
  - Reset (limit) the timer

Timer basics – PWM function

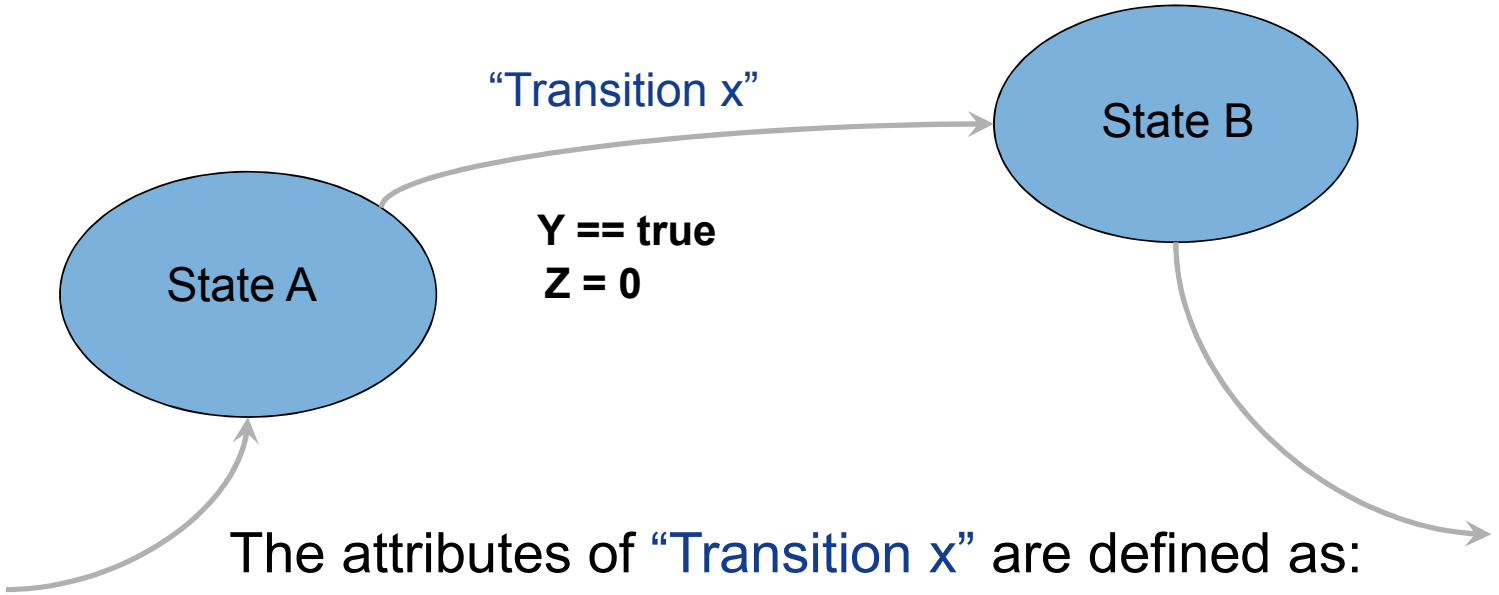
- Pulse-Width-Modulated signals can be generated by the standard timer block, enhanced by a few additional gates
- Transitions of the output signals are caused by periodic matching events



What is state machine

- A state machine is made of:
  - States
  - Inputs
  - Outputs
  - Transitions
- Can be represented in a flow graph
- Defines the behavior model for a system

Example of a State Machine Diagram



The attributes of "Transition x" are defined as:

- while the system is in State A
- when condition Y is true

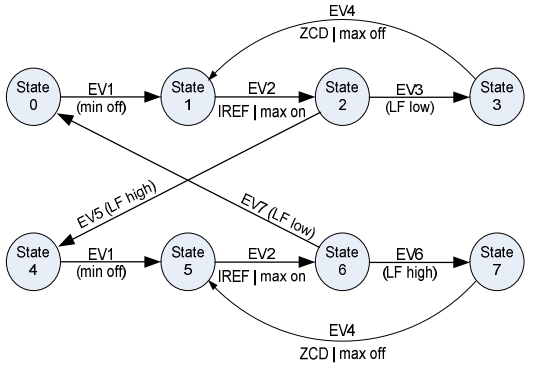
} REQUIRED CONDITIONS

- drive output Z low
- jump to State B

} TRIGGERED ACTIONS

SCT
INTRODUCTION

What is the SCT ?

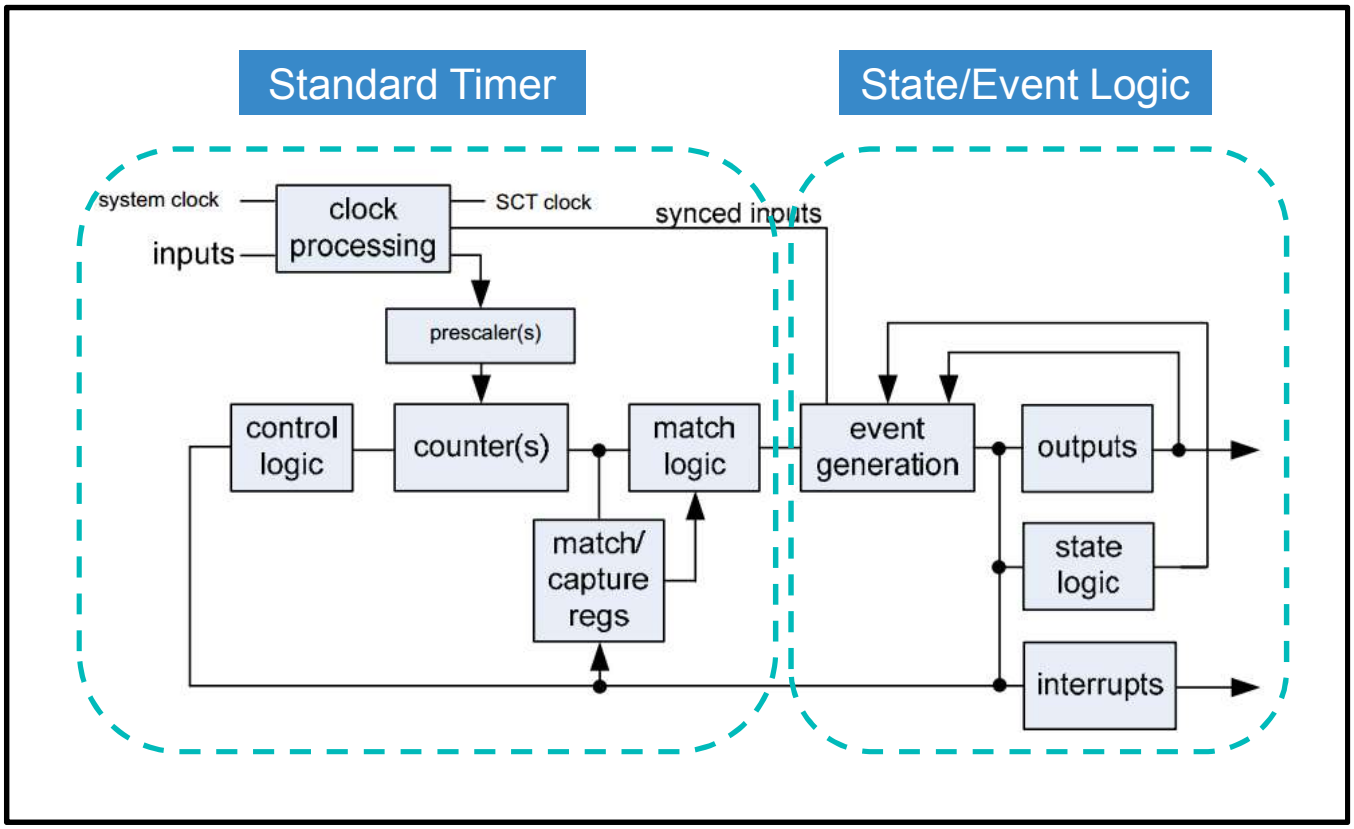


- Timer only or H/W FSM engine or Both
- As a timer:
  - UP, UPDN, Reload, MAT, CMP, IRQ, DMA
- As a Hardware Finite State Machine (FSM) engine
  - defines the behavior of counter, outputs, interrupts, dma in a flexible way
- As both
  - (Cowork) A lot of interconnections between all these
  - Timer&I/O generate events, events control timer

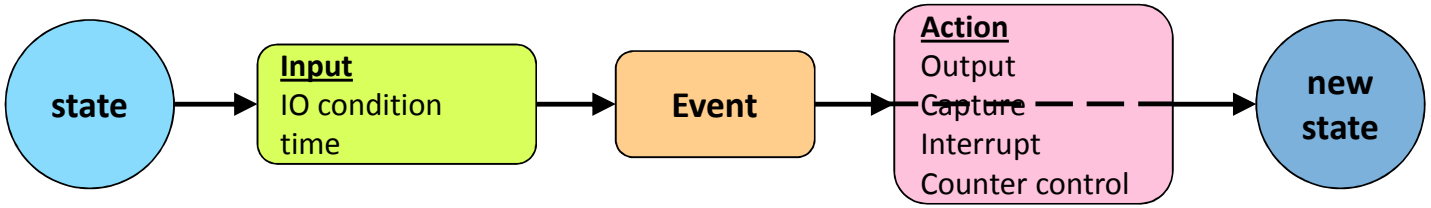
SCT Building blocks

- **Timer**
  - Can be partitioned as two 16-bit or one 32-bit timer
- **Events**
  - Can trigger a transition on outputs, change the state, change counter status
- **States**
  - Define the context in which the defined events are evaluated
- **Inputs**
  - Signals which get evaluated by SCT and might contribute to the generation of events
- **Outputs**
  - Signals generated by the SCT, which can also contribute to generation of events

SCT Block Diagram



Always keep this order in mind:



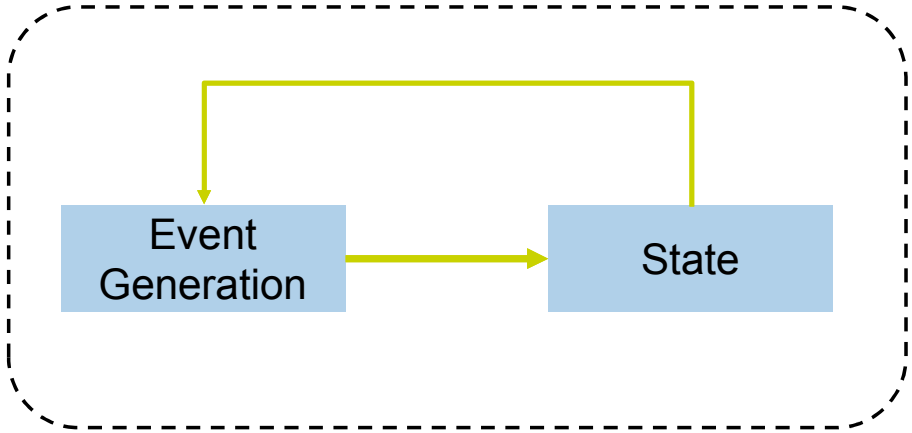
SCT - Events

- Source of an event can be:
  - Time based value (timer match)
  - Signal level (high / low) or rising / falling edge (for both inputs and outputs)
  - Time based value [AND | OR] [signal level | signal edge]

- Any event can:
  - Drive an output signal
  - Make the timer state machine jump to another state
  - Start / Stop / Halt / Limit the timer (also the other timer half!)
  - Capture the current counter value
  - Generate an interrupt or DMA request

SCT - States

- Usage of states is optional, but of course this is exactly for what the SCT is designed for ☺
- Each 16-bit timer half has its own dedicated state machine (32 states each)
- You can specify (mask) in which states a specific event is considered
- States allow for easy visual association between the behavior of the application and the SCT configuration



SCT - Inputs and outputs

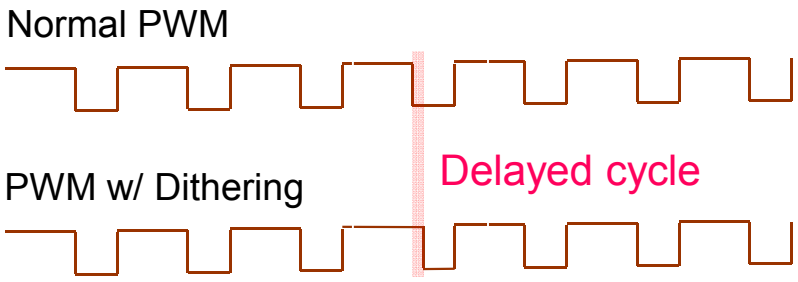
- **Inputs:**
  - Up to 8
  - Source can be outside or inside of the chip (physical IO pins or output signals coming from other on-chip peripherals, i.e. comparators, GPIOs, serial interfaces etc.)
  - Synchronized to the input clock
- **Outputs:**
  - Up to 16
  - Can also be evaluated like “inputs” and generate events (after the next counter clock cycle)
  - Can be routed to other IP blocks, like ADCs as trigger signals

State Machine vs. SCT - in a nutshell

Element	SCT implementation
States	- tracked in STATE register - updated according to EVENT CONTROL register
Inputs	- specified in EVENT CONTROL register
Outputs	- driven by events specified in SET and CLEAR registers - can also be associated with transitions in the EVENT CONTROL register
Transitions	- called “events” - defined in EVENT CONTROL register - enabled in EVENT STATE MASK register

Dithering functionality

- Improve average PWM output resolution(16 times)
- The dither engine delays the assertion of a match by one counter clock every n (0 to 15) out of 16 counter cycles
- n is specified in the 4-bit FRACMAT register
  - Eg, 15 counter period: duty cycle = 1/16,
    1 counter period: duty cycle = 2/16,
    16 counter period: average duty cycle 17/256, resolution from 16 to 256



Dithering table

	COUNTER CYCLE															
FRACMAT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000																
0001									D							
0010					D								D			
0011					D				D				D			
0100			D				D				D				D	
0101			D				D		D		D				D	
0110			D		D		D				D		D		D	
0111			D		D		D		D		D		D		D	
1000		D		D		D		D		D		D		D		D
1001		D		D		D		D	D	D		D		D		D
1010		D		D	D	D		D		D		D	D	D		D
1011		D		D	D	D		D	D	D		D	D	D		D
1100		D	D	D		D	D	D		D	D	D		D	D	D
1101		D	D	D		D	D	D	D	D	D	D		D	D	D
1110		D	D	D	D	D	D	D		D	D	D	D	D	D	D
1111		D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

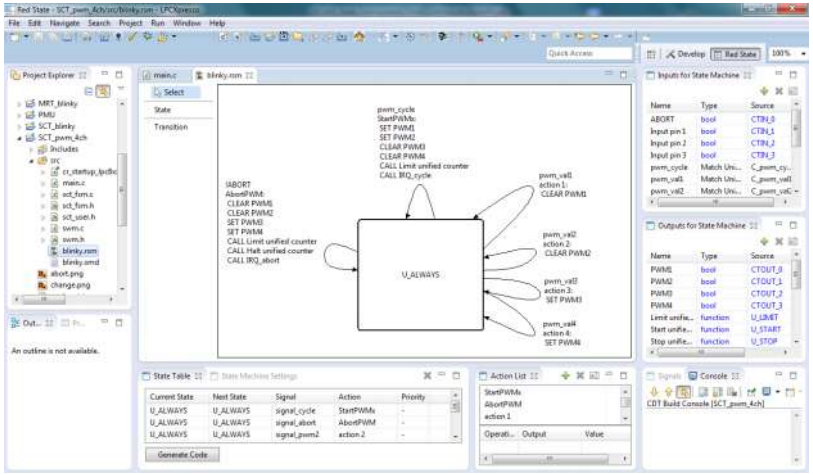
SCT AVAILABILITY

SCT implementation summary table

NXP Part	INPUT	OUTPUT	States	Event	MAT/ CAP	SCTIPU	Dither	PLL
LPC81x	4	4	2	6	5	No	No	No
LPC82x	4	6	8	8	8	No	No	No
LPC11U6x/E6x – SCT0/1	4	4	8	6	5	No	No	No
LPC15xx – SCT0/1 (Largest)	8	10	16	16	16	Yes	Yes	Yes
LPC15xx – SCT2/3	3	6	10	10	8	No	No	No
LPC18/43xx/LPC18S/43Sxx(flashless)	8	16	32	16	16	No	No	No
LPC18/43xx/LPC18S/43Sxx (flash)	8	16	32	16	16	No	Yes	No
LPC5410x	8	8	13	13	13	No	No	No
LPC5411x	8	8	10	10	10	No	No	No
LPC5460x/54S60x	8	10	10	10	10	No	No	No

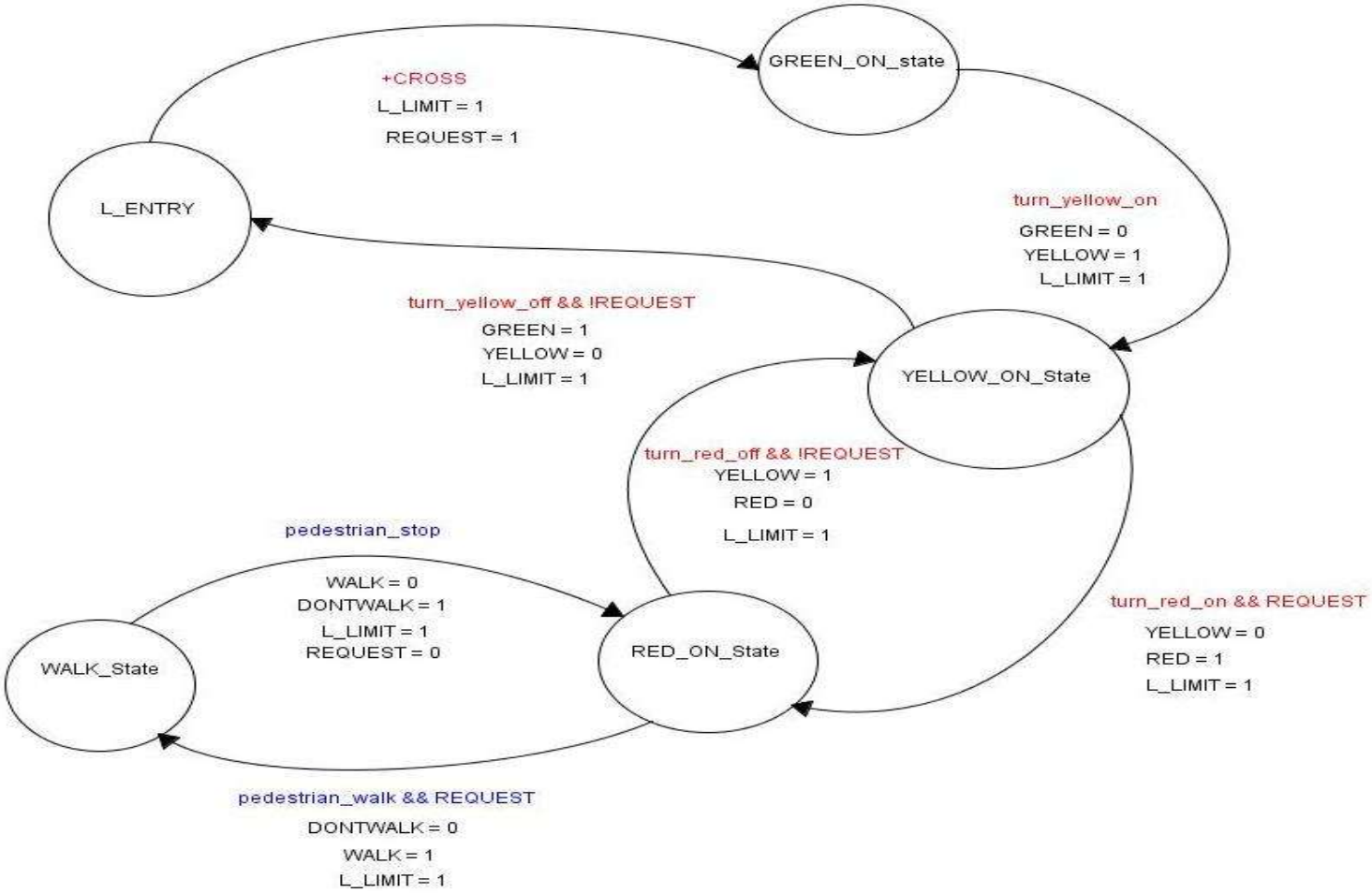
SCT TOOLS & RESOURCES

RedState in LPCXPresso IDE



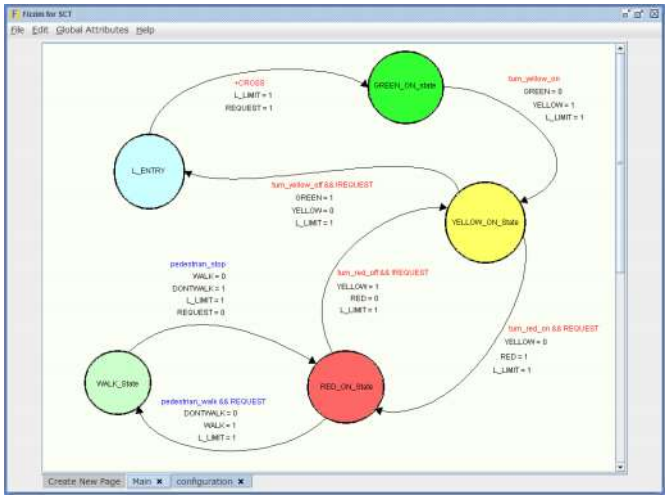
- Integrated into Eclipse based LPCXpresso IDE
- Supports both SCT and generic state machines programming
  - More information about the products on:
    - http://www.nxp.com/products/software-and-tools/software-development-tools/software-tools/lpc-microcontroller-utilities/lpcxpresso-ide-v8.2.2:LPCXPRESSO
    - https://community.nxp.com/community/lpcxpresso-ide
- See also:
  - AN11161 - Using the SCT in LPCXpresso, Keil, and IAR (with software)

Example of state machine configuration



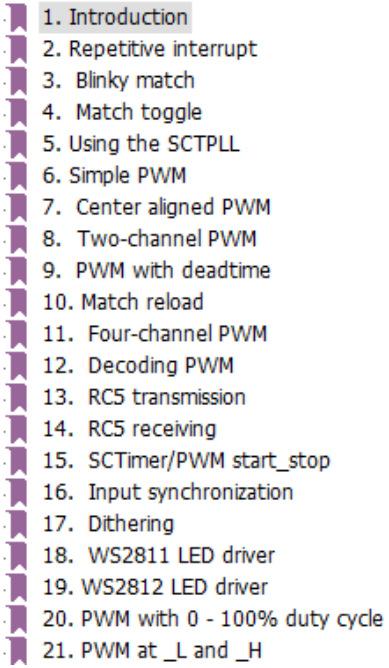
SCT-Tools

- Free of charge
- Standalone, graphic editor is a Java based tool
- Generates C code register initializations and header file
- Package includes
  - Program installer for Windows
  - Installation guide and user manual
  - Programming examples, tutorial
- Example projects are based on Keil μVision but can be easily adapted for any other IDE. They include the SM definition file and call the C code generator from the IDE as a “custom build” step
- Mainly used for LPC18/43xx, no further update



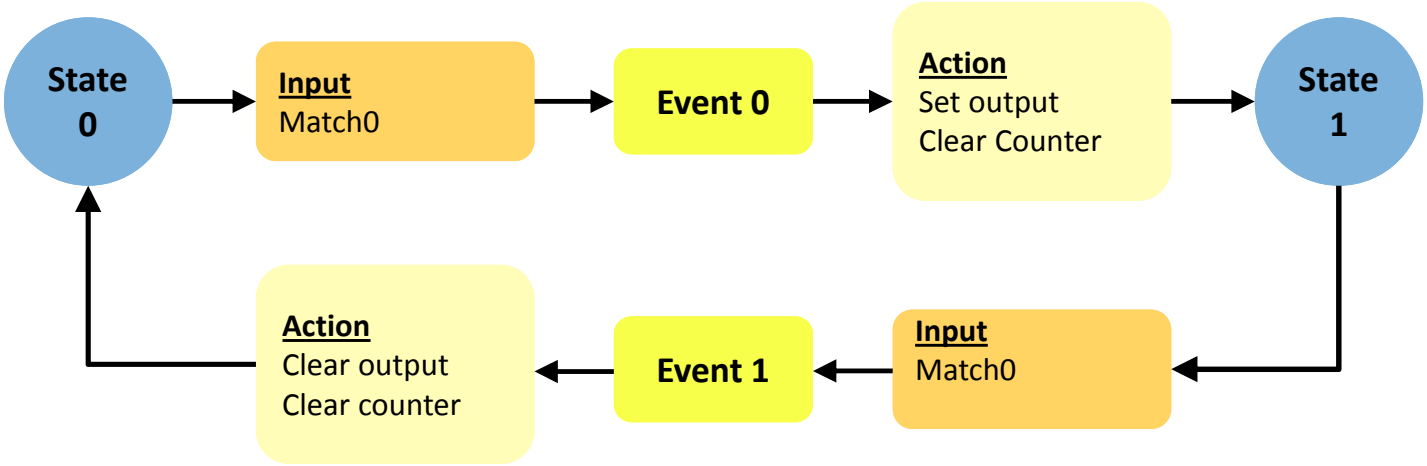
SCT Cookbook

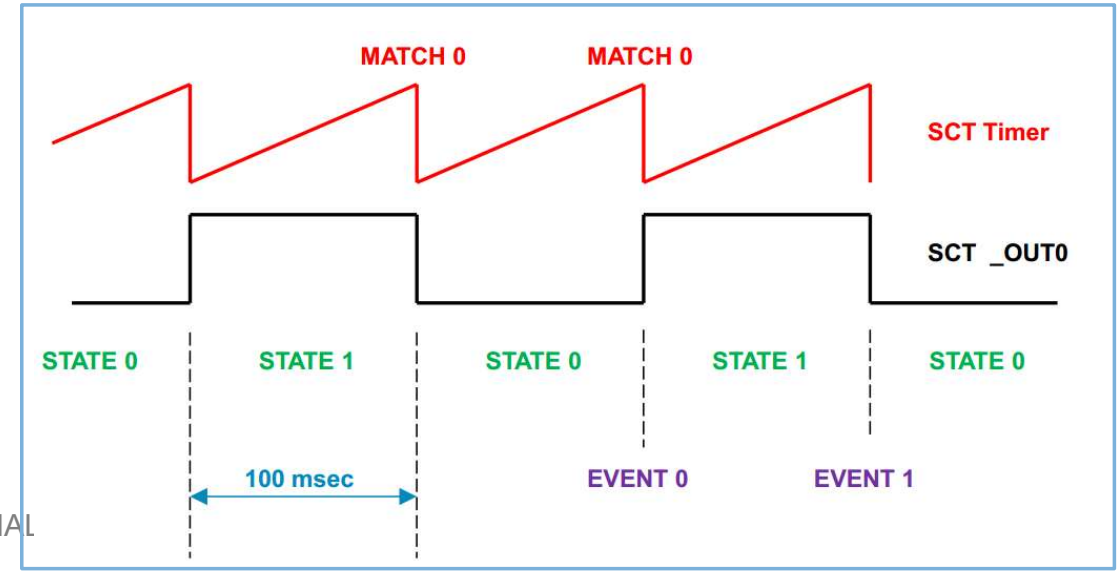
- **AN11538**: **SCTimer/PWM Cookbook**
- Collection of code examples (Keil, IAR and LPCXpresso)
- Each code example summarized in Cookbook document
- Available so far (and more to follow):



SCT APPLICATION ANALYSIS

Blinky Match



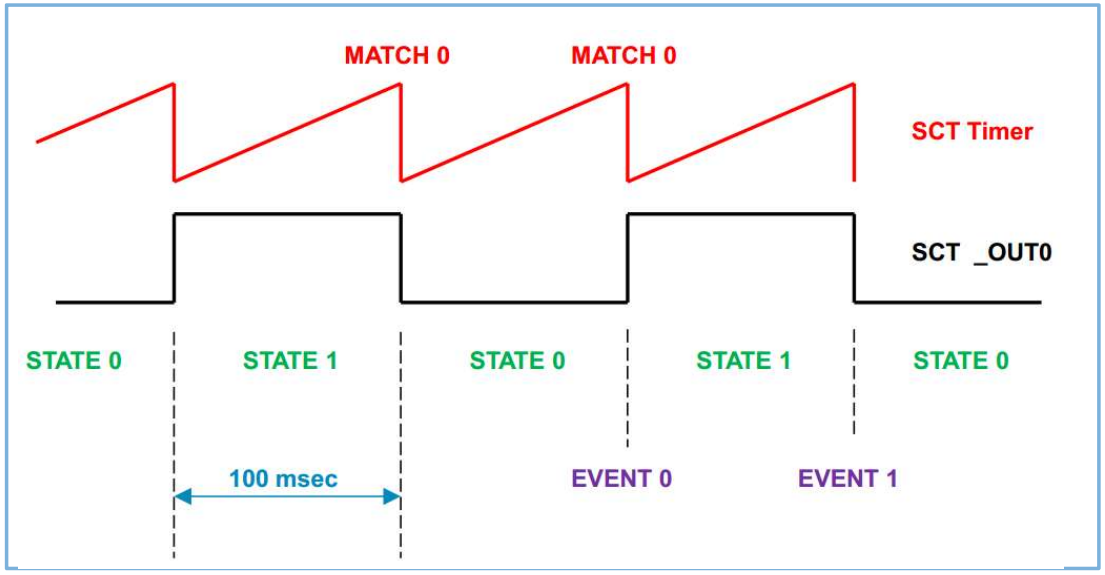


Source code implementation

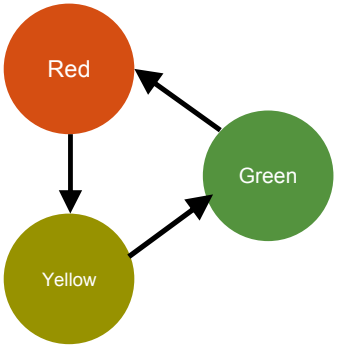
```
void SCT_Init(void)
{
    LPC_SCT->CONFIG |= 0x1;                                  // unified timer

    LPC_SCT->MATCH[0].U    = SystemCoreClock/10;             // match 0 @ 100 msec
    LPC_SCT->MATCHREL[0].U = SystemCoreClock/10;

    LPC_SCT->EV[0].STATE = 0x00000001;                        // ev 0 happens in state 0
    LPC_SCT->EV[0].CTRL  = (0 << 0)  |                        // related to match 0
                           (1 << 12) |                        // match condition only
                           (1 << 14) |                        // STATEV is new state
                           (1 << 15);                         // STATEV[15] = 1
    LPC_SCT->EV[1].STATE = 0x00000002;                        // ev 1 happens in state 1
    LPC_SCT->EV[1].CTRL  = (0 << 0)  |                        // related to match 0
                           (1 << 12) |                        // match condition only
                           (1 << 14) |                        // STATEV is new state
                           (0 << 15);                         // STATEV[15] = 0

    LPC_SCT->OUT[0].SET = (1 << 0);                          // event 0 sets SCT_OUT_0
    LPC_SCT->OUT[0].CLR = (1 << 1);                          // event 1 clears SCT_OUT_0
    LPC_SCT->LIMIT_L = 0x0003;                               // event 0 and 1 are limits

    LPC_SCT->CTRL_L &= ~(1 << 2);                            // unhalt the timer}
}
```

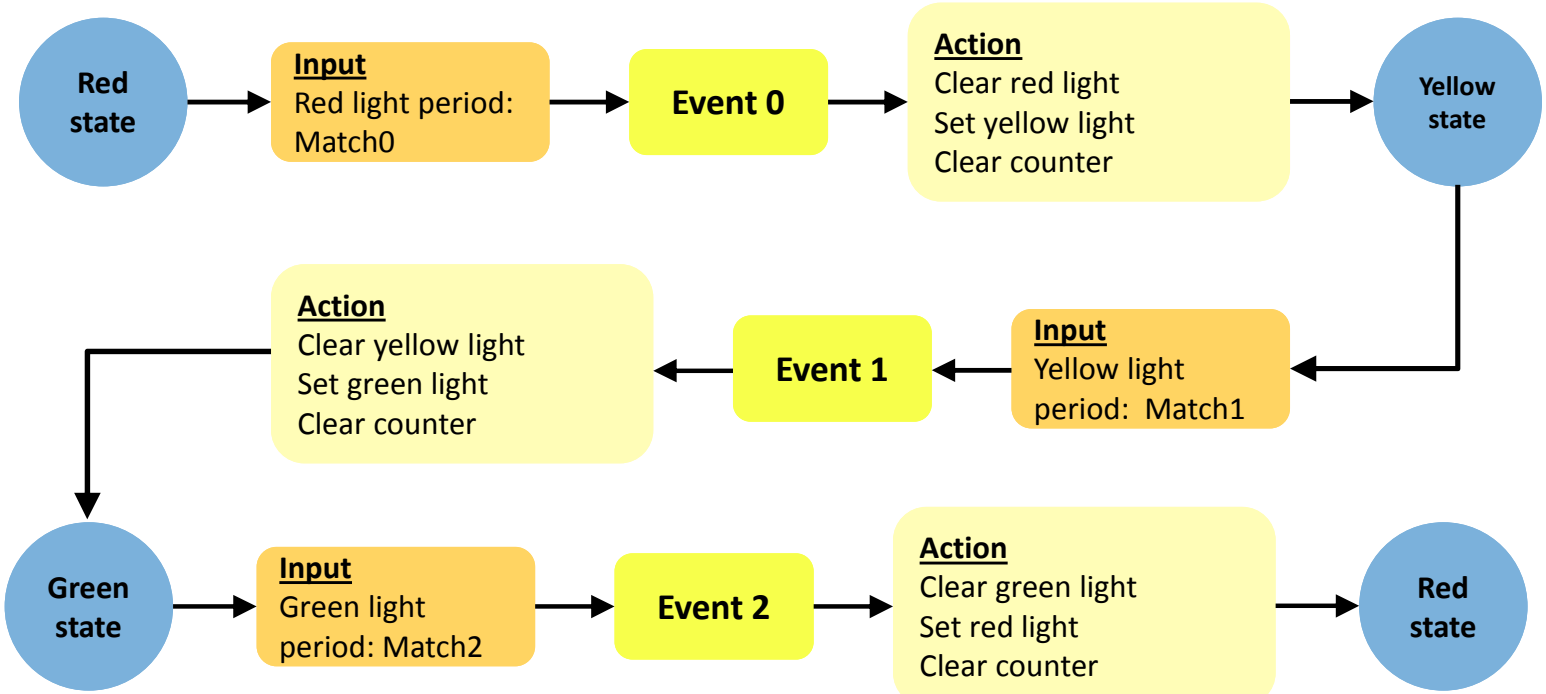
Configuration Analysis

- Blinky Match Configuration
  - Match register：match0@100ms
  - Output：SCT_OUT0，SCTx_OUT0 connected to an LED that is illuminated when the output is low (during state 0)
  - Event: Event 0 and Event 1, Event0 sets SCT_OUT0, Event1 clear SCT_OUT0
  - State: State0 and State 1, Event0 enabled in State0, Event1 enabled in State1



Traffic Light





Application use cases

- Motor Control
  - Generation of PWM outputs, triggering of ADC sample points
- Lighting
  - Modulated PWM outputs, reaction to lamp sensor feedback
- Generation of custom control signals in hardware, like:
  - Clock or signal gating
  - Complex modulation of outputs
  - Pulse sequences
- Custom sampling of input signals for
  - Frequency detection
  - Pulse width detection
  - Phase detection
- And others

Motor control

SCT is used to handle signals like:

- Hall sensor feedback, i.e. for a Brushless DC motor
  - Problem is to determine the motor position during rotation
  - The 3 Hall sensors provide positional information
  - a total of 6 different combination of the signals provide a 60° resolution
- ADC triggering
  - Used for sampling the currents on the motor windings, i.e on Brushless AC motor
  - Sampled values flow back in the control algorithms like field oriented control (FOC)
- PWM signals
  - Depending on the type of motor, and the power stage configuration, up to 6 PWM signals (phases) need to be generated

SECURE CONNECTIONS
FOR A SMARTER WORLD