Program Modeling Concepts:

Lesson-6: FSM STATE TABLE AND ITS APPLICATIONS

FSM State Table

- *A state table* can then be designed for representation of every state in its rows. The following six columns are made for each row.

1. *Present State* name or identification
2. *Action (s)* at the state until some event(s)
3. *The events (tokens)* that cause the execution of state transition function
4. *Output (s)* from the state output function (s)
5. *Next State*
6. *Expected Time Interval* for finishing the transitions to a new state after the event

Coding using while ( ) for each row

- while (*presentState*) {***action*** ( ); if (event = ...; token = ...) {output = ...; ***stateTransitionFunction*** ( ); };}

Coding using switch and case for each row

- Switch (State)
- Case *presentState:* ***action*** ( ); if (event = ...; token = ...) {output = ...; ***stateTransitionFunction*** ( ); };}

Meaning of terms

- *presentState* a Boolean variable, which is true as long as the present state continues and turns false on transition to the next.
- The ***action*** ( ) is a function that executes at the state.
- If certain events occur and tokens are received (for example, clock input in a timer), a state transition function, ***stateTransitionFunction***, is executed which also makes *presentState* = false and transition occurs to the next state by setting *nextState* (a Boolean variable) = true.

Example– Mobile phone key '5' of T9 keypad

- A mobile phone T9 keypad key marked 5 has five states. It undergoes transitions from initial state (0, 5) as follows: (0, 5) $\rightarrow$ (1, 5) $\rightarrow$ (1, *j*) $\rightarrow$ (1, *k*) $\rightarrow$ (1, *l*) $\rightarrow$ (l, 5) $\rightarrow$ (1, *j*).

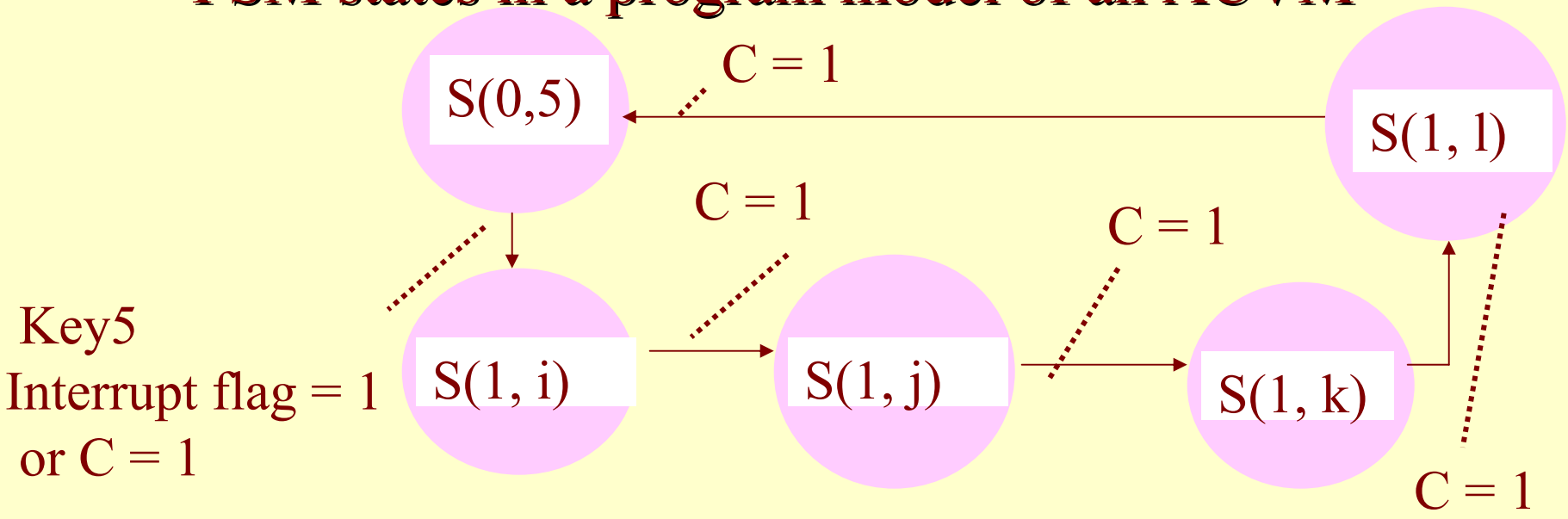
Example State transition at key ‘5’ of T9 keypad

- Occurs when key-input flag KF = 1 (set).
- State transition occurs when TF = 0 and KF = 1.
- The transition resets the key.
- The timer starts on key-interrupt and when is in ON state and the timer run flag
- TR = true.

Example State transition at key ‘5’ of T9 keypad

- When timer is off or till timer timeouts TR = 0.
- When count = x, timer flag TF = 0 (false) before timeout. Time occurs when count x = compare register loaded for 1 s. After 1 s, the TF = true.
- There are twelve finite number of states of the key and timer together.

FSM states in a program model of an ACVM



if (key5interruptflag == 1 & TR == 1 & TF == 0) then C = 1
else if (TR == 0 & TF == 1) then C = 0; /* TR = 1 means timer is running after key interrupt, TF = 1 means timer timeout. C = 1 means that key interrupt has occurred, timer then started running but 1s timeout flag is no yet set.

States and rows in State table for FSM for key 5

Present State				Event		Next State on event			
Key	TR	TF	KF	KF	Count	Key	TR	TF	KF
(0, 5)	0	0	0	1	x'	(1, j)	1	0	0
(1,5)	1	0	0	1	x	(1, j)	1	0	0
(1, j)	1	0	0	1	x	(1, *k)*	1	0	0
(1, *k)*	1	0	0	1	x	(1, *l)*	1	0	0
(1, *l)*	1	0	0	1	x	(1, *5)*	1	0	0
(1,5)	0	1	0	0	0	(1,5)	0	0	0
(1, j)	0	1	0	0	0	(1,j)	0	0	0
(1, *k)*	0	1	0	0	0	(1,k)	0	0	0
(1, *l)*	0	1	0	0	0	(1,l)	0	0	0

States and rows in State table for FSM for key 5

- Count = x' initial count register value. = x means counts greater than x' and less than a compare register value set for 1 s timeout.
- TR = 0 means timer stopped.
- TR =1 means timer running after loading a value in compare register for capture time out after 1 s.
- TF = 1 means timer compare timeout.
- KF = 1 key press event.
- KF = 0 means key read or initial inactive

C Code from the Table

- # define true 1
- # define false 0
- # define initialState “05000”
- # define state1 “15100”
- # define state2 “1j100”
- # define state3 “1k100”
- # define state4 “11100”
- # define state5 “15010”
- # define state6 “1j010”

C Code from the Table

- # define state7 “1k010”
- # define state8 “1l010”
- # define state9 “15000”
- # define state10 “1j000”
- # define state11 “1k000”
- # define state12 “1l000”

C Code from the Table

- void Key5FSM ( ) {
- char [ ] *state*;
- initialState = “05000”
- while (true) { /* An infinite loop */
- /*--*/
- /* function display (“*x*”) shows character x on the screen and function cursor_next ( ) moves the cursor position to next when keyiing in an SMS text message. SWI is software interrupt instruction */

C Code from the Table

- Switch (State) {
- /**********************************/
- initialState: if ((KF == 1) && Count == 0)) {
- SWI timerstart; /* Execute Interrupt routine to start the timer */
- display (“5”); State = State1;}
- break;
- /**********************************/

C Code from the Table

```
State1:  if ((KF == 1) && Count == x)) {
SWI timerRestart; /* Execute Interrupt routine
   to restart the timer */
display ("j"); State = State2;}
      break;
```

C Code from the Table

```
State2:  if ((KF == 1) && Count == x)) {
SWI timerRestart; /* Execute Interrupt routine
   to restart the timer */
display ("k"); State = State3;}
      break;
/**************************/
```

C Code from the Table

```
State3:  if ((KF == 1) && Count == x)) {
SWI timerRestart; /* Execute Interrupt routine
   to restart the timer */
display ("k"); State = State4;}
      break;
```

C Code from the Table

```
State4:  if ((KF == 0) && Count == x)) {
SWI timerRestart; /* Execute Interrupt routine
   to restart the timer */
display ("1"); State = State5;}
      break;
/**************************************/
```

C Code from the Table

- State5: if ((KF == 0) && Count == 0)) {
- SWI timerReset; /* Execute Interrupt routine to reset and stop the timer */
- display ("5"); cursor_next ( ); State = State9;}
- exit ( );
- /**********
  **********/

C Code from the Table

- State6: if ((KF == 0) && Count == 0)) {
- SWI timerReset; /* Execute Interrupt routine to reset and stop the timer */
- display ("*j*"); cursor_next ( ); State = State10;}
- exit ( );

C Code from the Table

- State7: if ((KF == 0) && Count == 0)) {
- SWI timerReset; /* Execute Interrupt routine to reset and stop the timer */
- display ("*k*"); cursor_next ( ); State = State11;}
- exit ( );
- /**********
  **********/

C Code from the Table

- State8: if ((KF == 0) && Count == 0)) {
- SWI timerReset; /* Execute Interrupt routine to reset and stop the timer */
- display ("*l*"); cursor_next ( ); State = State12;}
- exit ( );
- /**********
  **********/

C Code from the Table

```
}
/*------End of Switch -case ---------------*/
} /* End of While infinite loop */
} /* End of Key5FSM */
```

Summary

We learnt

- When using a FSM model, a state table representation becomes very handy while en coding
- Use of state table for programming for the key 5 in mobile T9 keypad

End of Lesson 6 of Chapter 6