MACQUARIE University
SYDNEY·AUSTRALIA

Embedded Systems Design

ELEC3042 EMBEDDED SYSTEMS

Lecture 4

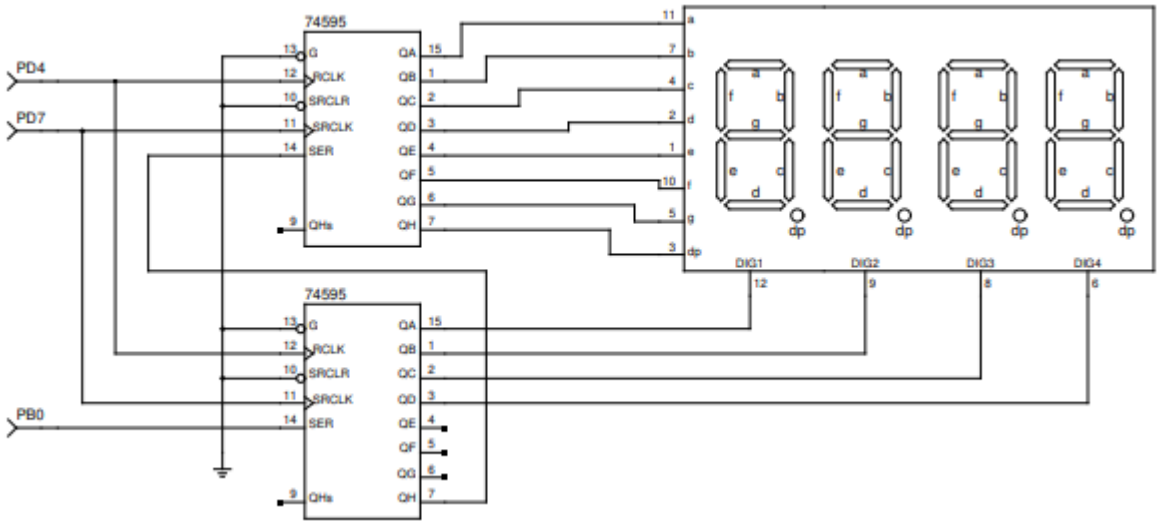
Computer Programs

MACQUARIE University
SYDNEY·AUSTRALIA

- Are step-by-step instructions for a computer to solve a problem
- Most problems are difficult to solve all at once. Begin by decomposing the problem into smaller parts
  - For each part, identify the input(s) and output(s)
  - Identify the steps that transform the input(s) into output(s)
  - Write the code to instruct the computer how to transform the input to output (usually as a function)
- Once you have solved all the smaller parts, combine them together to solve the overall problem

7-Segment Display

- Allow alphanumeric characters to be displayed with simple hardware
- HW-262 has four 7-segment displays connected to two 8-bit shift registers in series to form a 16-bit shift register
  - 8 bits are used to turn on/off the segments
  - 4 bits are used to turn on/off the digits
  - 4 bits are unused
- Shift register is controlled via 3 lines
  - Data (PB0) – the 16 control bits
  - Clock (PD7) – data shifts into registers on rising edge
  - Latch (PD4) – latches data to shift register output to be displayed



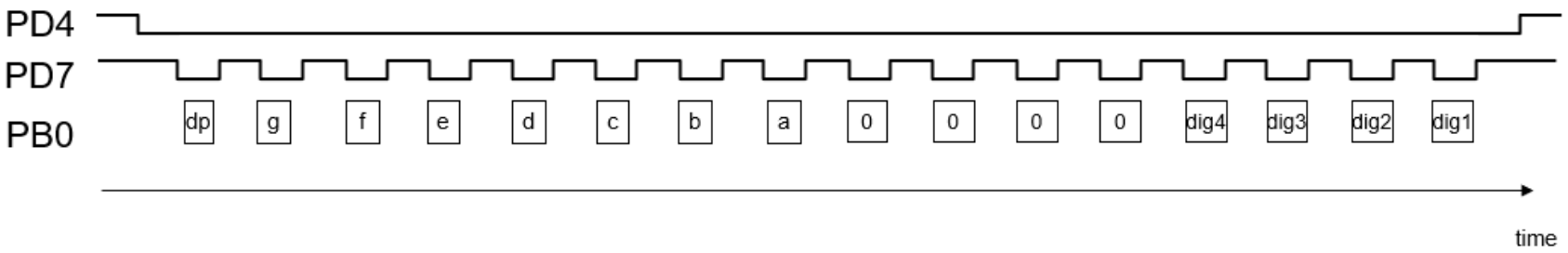
- Turn on/off a segment by sending a low/high voltage (active low)
- Select a digit by setting the corresponding bit high
- To show a four-digit number, we have to time multiplex the display

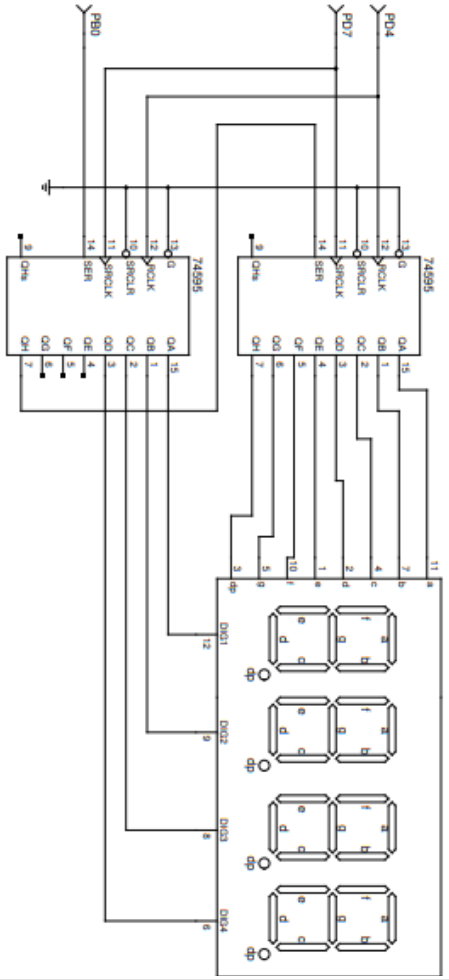
Exercise

WRITE A PROGRAM TO SHOW A FOUR-DIGIT NUMBER

- Break the problem into parts
- For each part, identify its inputs and outputs

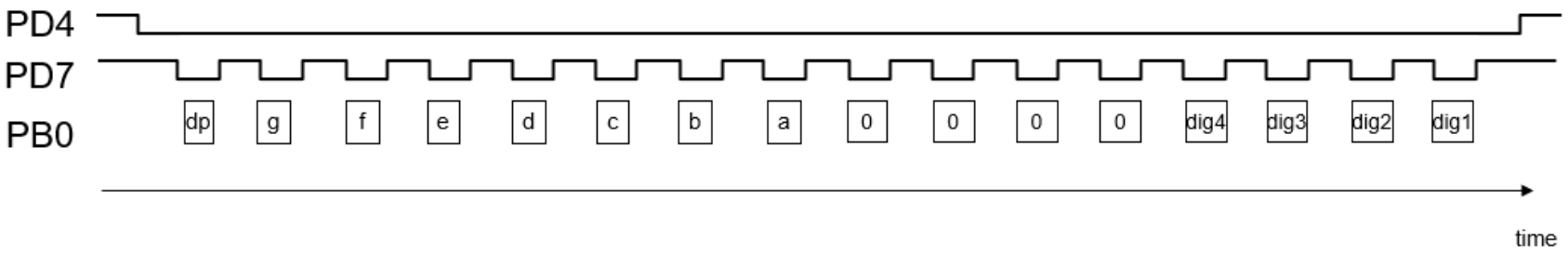
Showing a single character

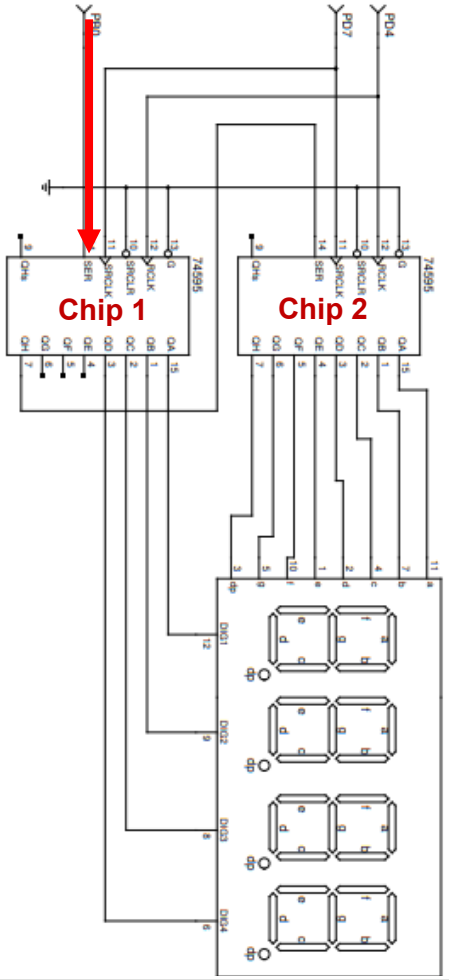




Showing a single character

AT EACH 0 -> 1 TRANSITION OF PD7, A NEW BIT IS CLOCKED IN

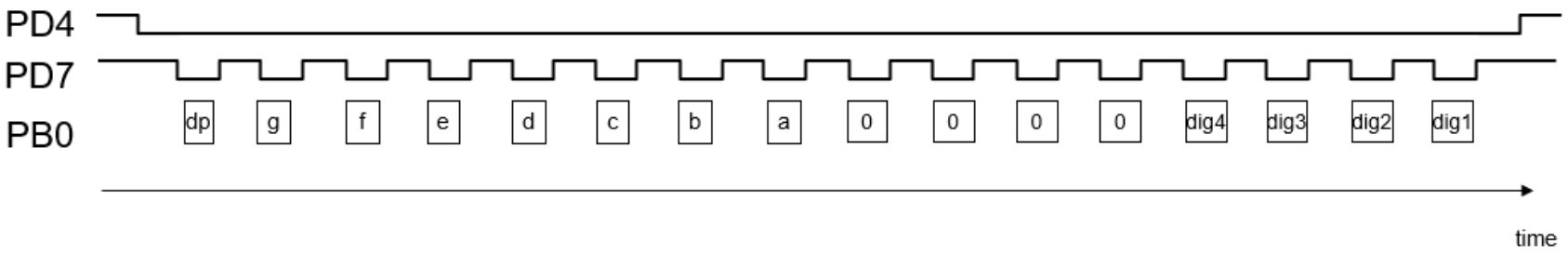


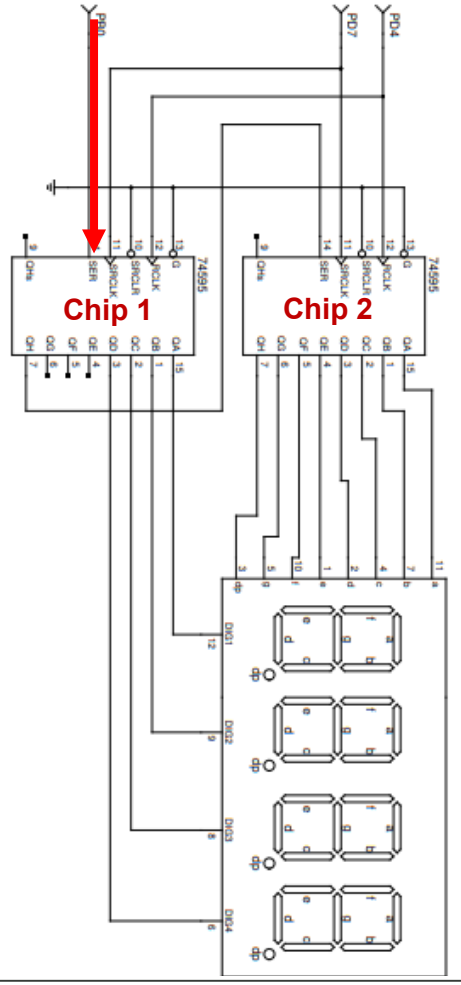


Chip 1								Chip 2							
QA	QB	QC	QD	QE	QF	QG	QH	QA	QB	QC	QD	QE	QF	QG	QH
dp															

Showing a single character

AT EACH 0 -> 1 TRANSITION OF PD7, A NEW BIT IS CLOCKED IN

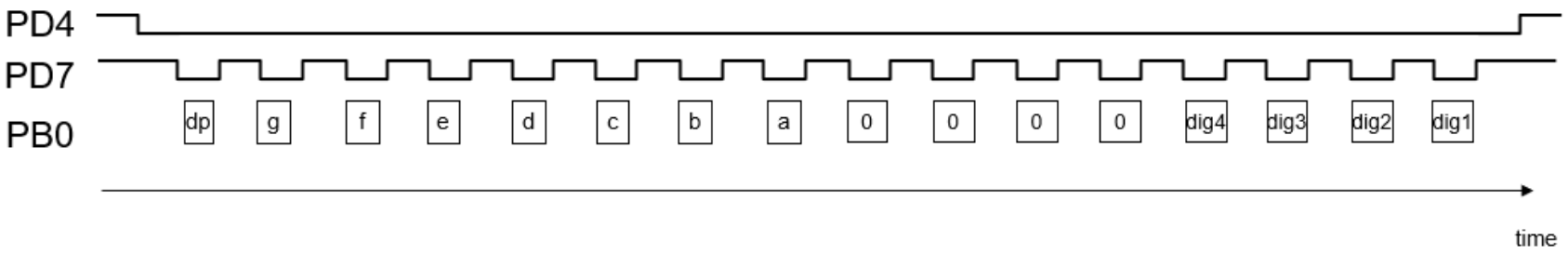


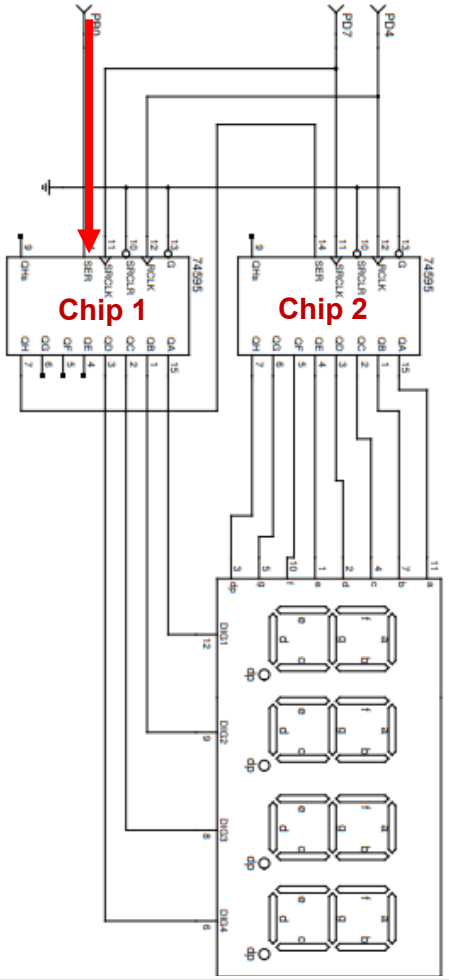


Chip 1								Chip 2							
QA	QB	QC	QD	QE	QF	QG	QH	QA	QB	QC	QD	QE	QF	QG	QH
g	dp														

Showing a single character

AT EACH 0 -> 1 TRANSITION OF PD7, A NEW BIT IS CLOCKED IN

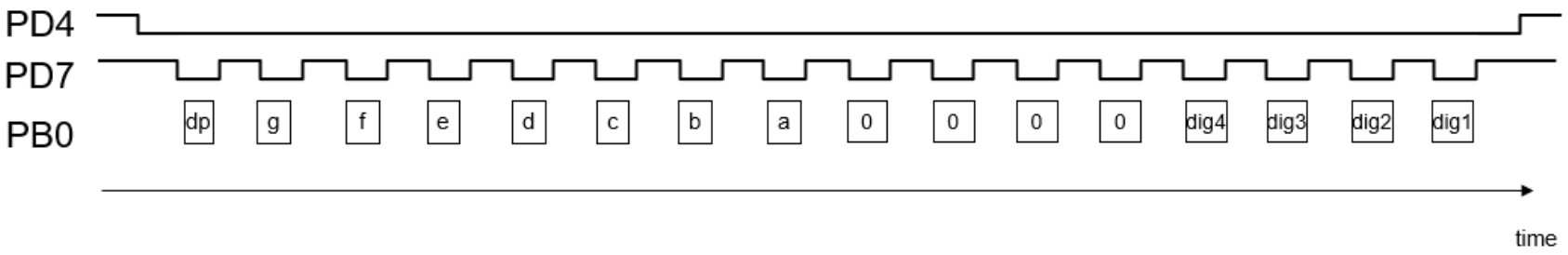


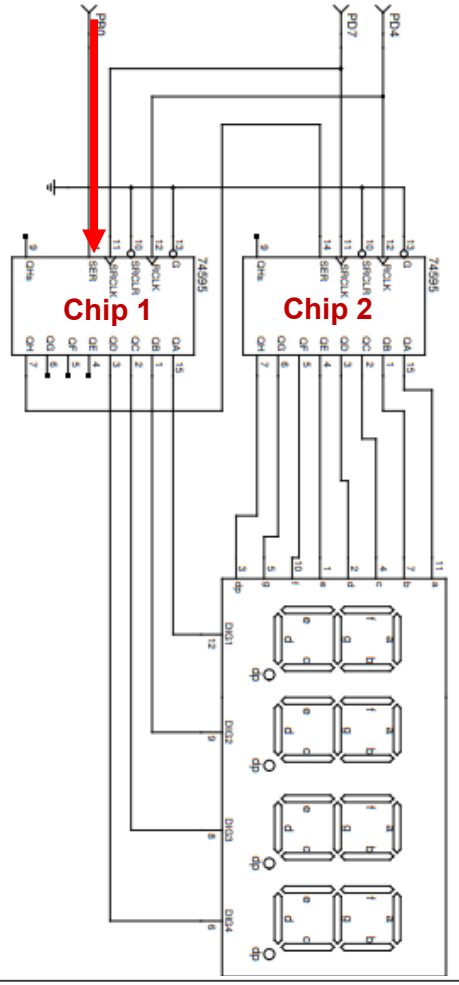


Chip 1								Chip 2							
QA	QB	QC	QD	QE	QF	QG	QH	QA	QB	QC	QD	QE	QF	QG	QH
f	g	dp													

Showing a single character

AT EACH 0 -> 1 TRANSITION OF PD7, A NEW BIT IS CLOCKED IN

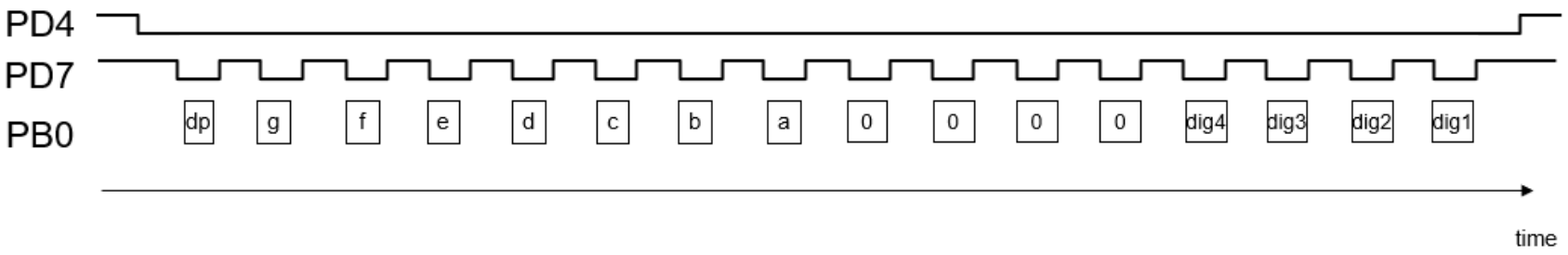


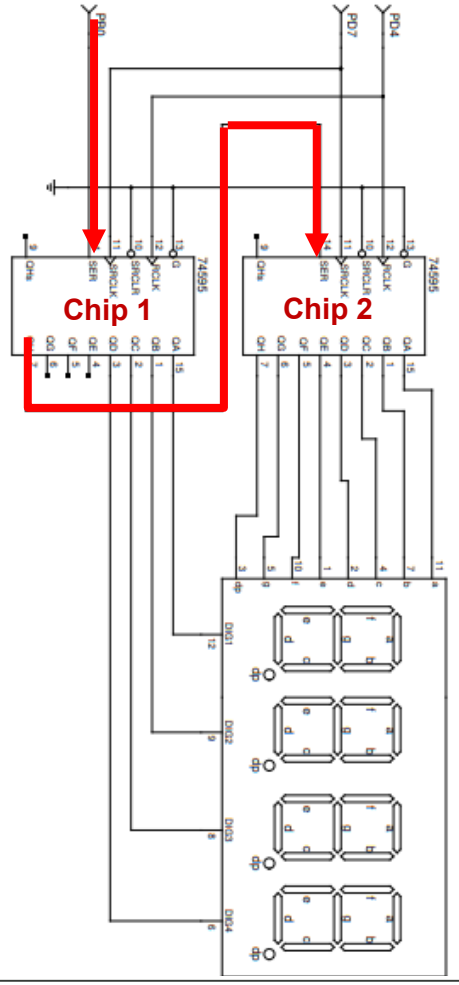


Chip 1								Chip 2							
QA	QB	QC	QD	QE	QF	QG	QH	QA	QB	QC	QD	QE	QF	QG	QH
e	f	g	dp												

Showing a single character

AT EACH 0 -> 1 TRANSITION OF PD7, A NEW BIT IS CLOCKED IN

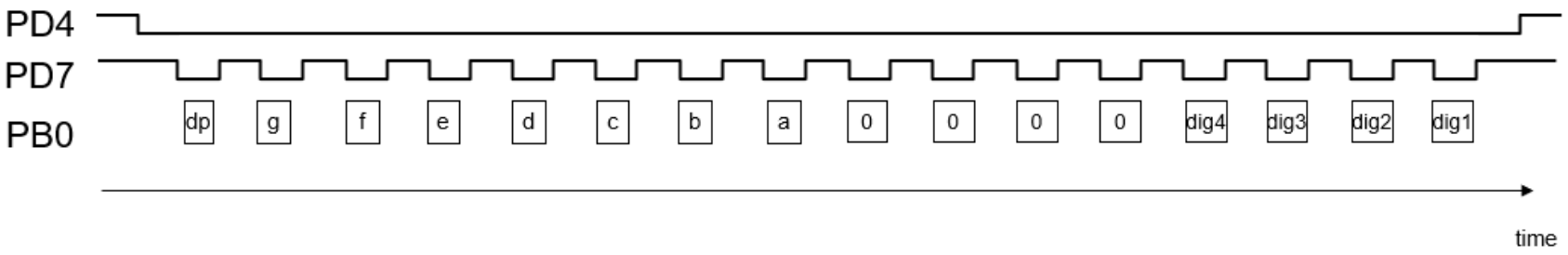


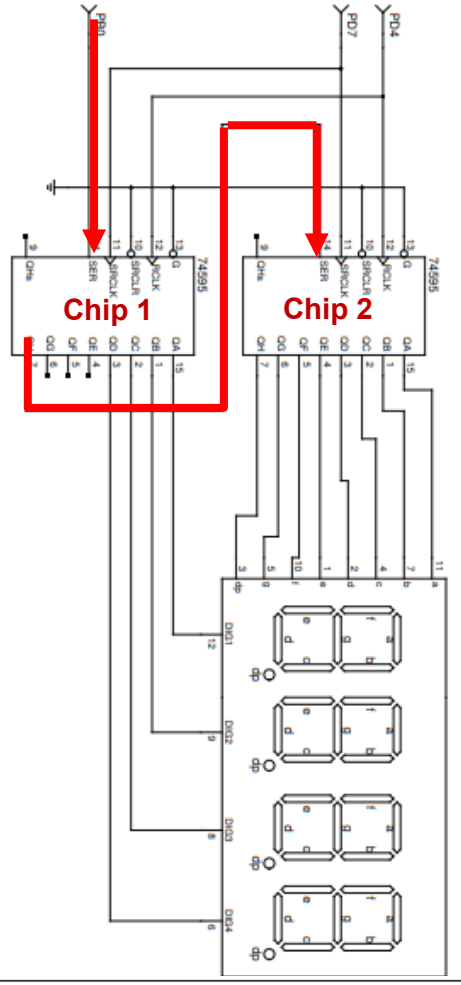


Chip 1								Chip 2							
QA	QB	QC	QD	QE	QF	QG	QH	QA	QB	QC	QD	QE	QF	QG	QH
0	a	b	c	d	e	f	g	dp							

Showing a single character

AT EACH 0 -> 1 TRANSITION OF PD7, A NEW BIT IS CLOCKED IN

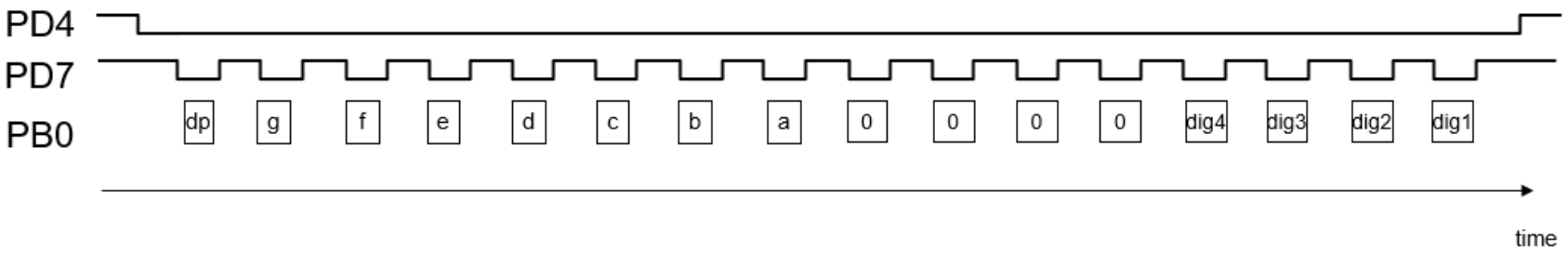


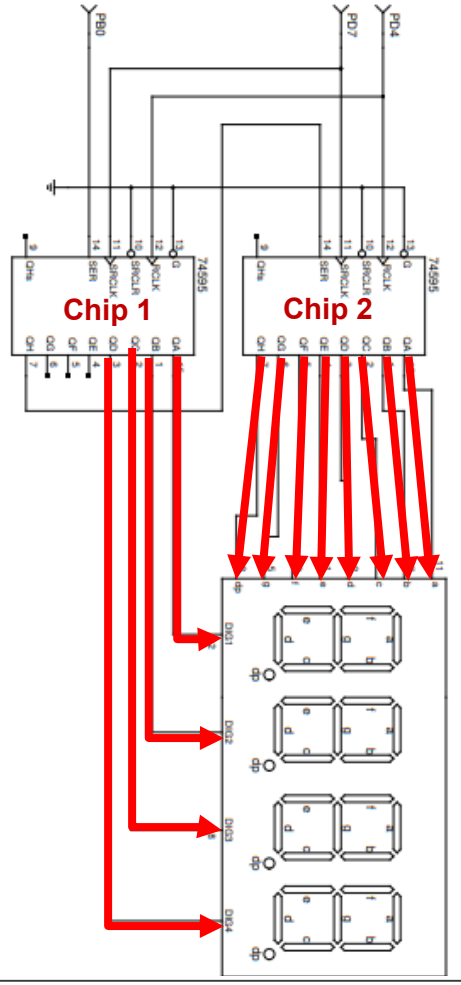


Chip 1								Chip 2							
QA	QB	QC	QD	QE	QF	QG	QH	QA	QB	QC	QD	QE	QF	QG	QH
dig1	dig2	dig3	dig4	0	0	0	0	a	b	c	d	e	f	g	dp

Showing a single character

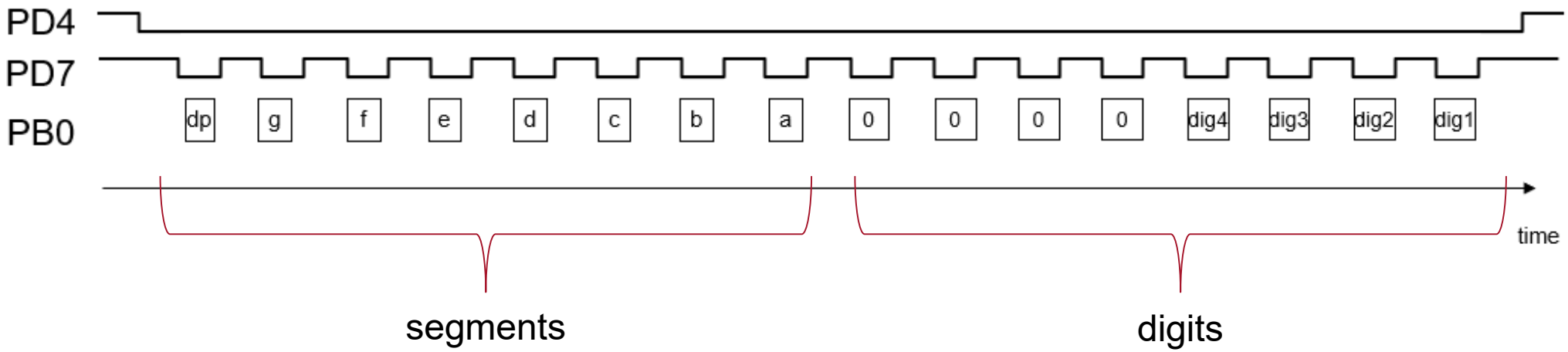
PD4 GOES HIGH TO LATCH TO DISPLAY





Chip 1								Chip 2							
QA	QB	QC	QD	QE	QF	QG	QH	QA	QB	QC	QD	QE	QF	QG	QH
dig1	dig2	dig3	dig4	0	0	0	0	a	b	c	d	e	f	g	dp
dig1	dig2	dig3	dig4	0	0	0	0	a	b	c	d	e	f	g	dp

Showing a single character




```
// send the 8 bits of data in segments (top bit first)
// then the 8 bits of data in digits (top bit first)
void sendData(uint8_t segments, uint8_t digits) {
    // TODO
}
```

Showing four digits

MACQUARIE University
SYDNEY·AUSTRALIA

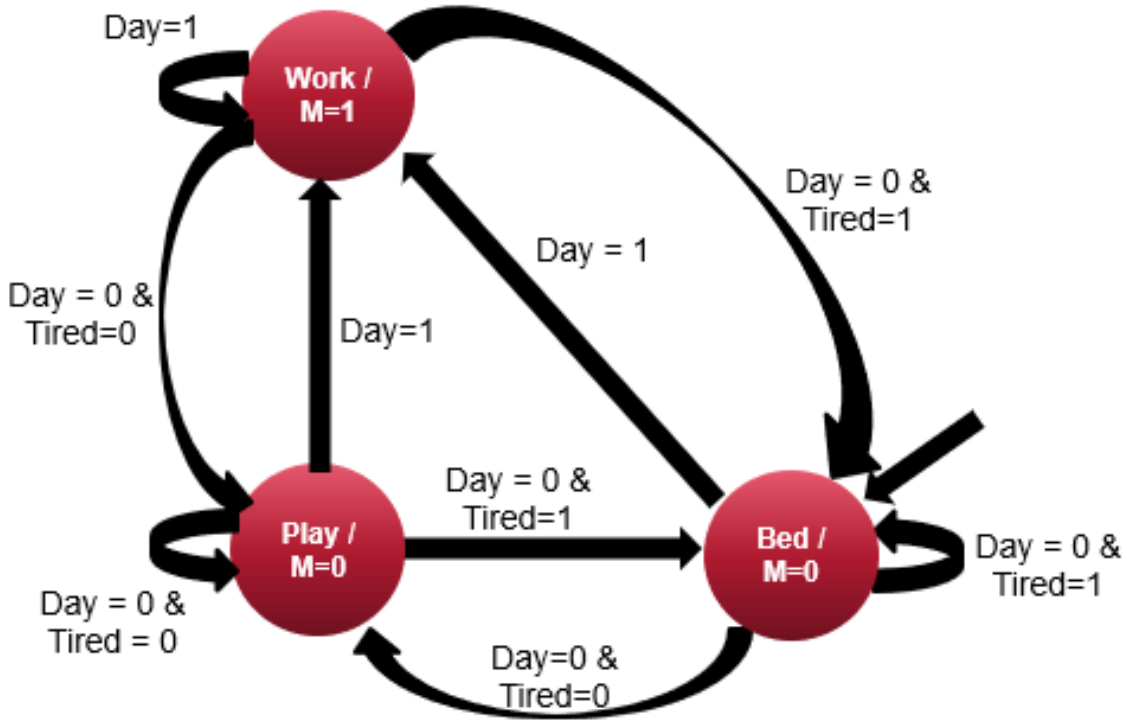
```
char digits[4];

void showDigits() {
    for (int i = 0; i < 4; i++) {
        sendData(segmentMap(digits[i]), (1 << i));
    }

}
```

State Machines

MACQUARIE University
SYDNEY·AUSTRALIA

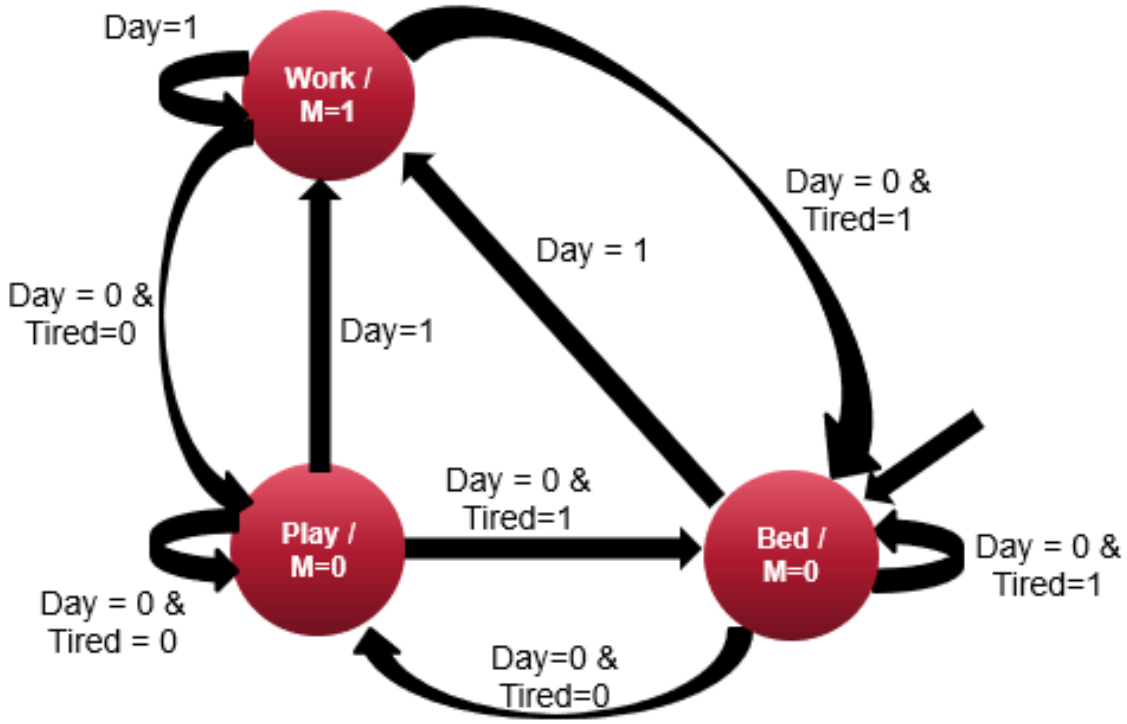
- Is a model describing the behaviour of a system as a set of states
  - A system can only be in one state at any time
  - Transition between states is dependent on the current state and the inputs to the system
- We can describe the relationship between states and their transitions via diagrams, equations, or tables
  - For this unit, you will be required to document the design of your system using:
    1. State transition diagrams
    2. State transition tables

State Transition Diagram



- Pictorial representation that is easy to follow
- Shows relationships between the states of a system

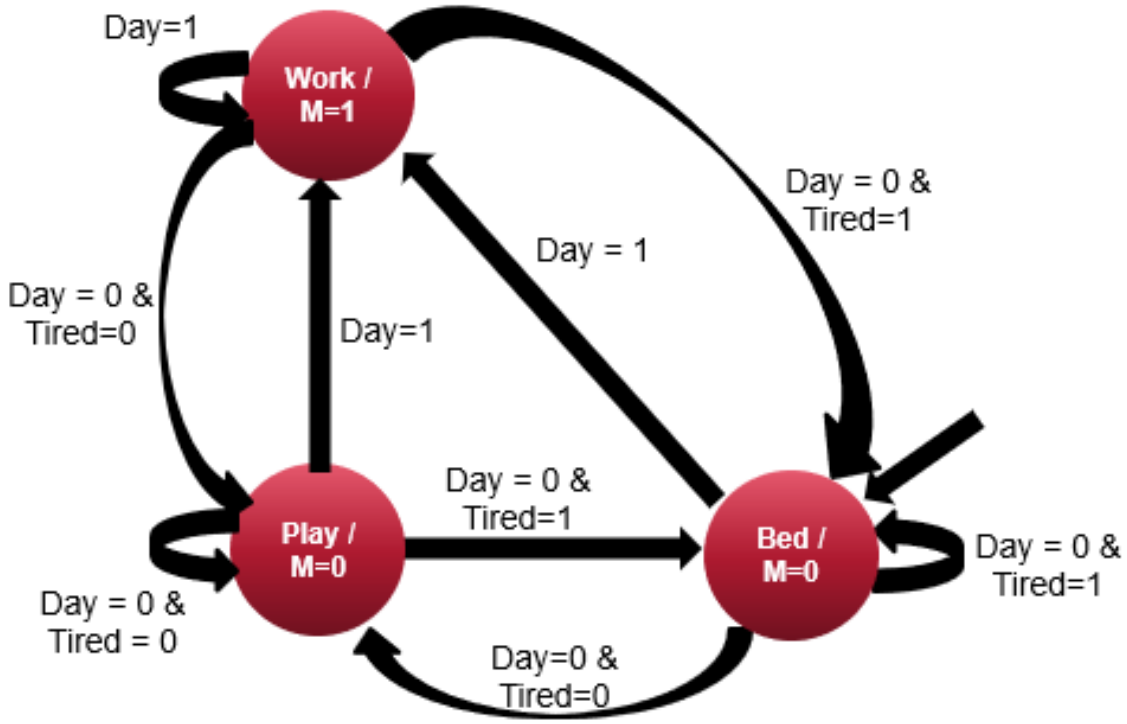
State Transition Table



- Easy way to ensure all possible inputs to a state are accounted for

Inputs		Current State	Next State	Output
Day	Tired			M
0	0	Work	Play	1
0	1	Work	Bed	1
1	0	Work	Work	1
1	1	Work	Work	1
0	0	Play	Play	0
0	1	Play	Bed	0
1	X	Play	Work	0
0	0	Bed	Play	0
0	1	Bed	Bed	0
1	X	Bed	Work	0

State Diagram to Code




```
switch (current_state) {
    case Work:
        M = 1;
        if (Day == 0) {
            if (Tired == 0) {
                current_state = Play;
            } else {
                current_state = Bed;
            }
        } else {
            current_state = Work;
        }
        break;
    case Play:
        M = 0;
        if (Day == 0) {
            if (Tired == 0) {
                current_state = Play;
            } else {
                current_state = Bed;
            }
        } else {
            current_state = Work;
        }
        break;
    case Bed:
        M = 0;
        if (Day == 0) {
            if (Tired == 0) {
                current_state = Play;
            } else {
                current_state = Bed;
            }
        } else {
            current_state = Work;
        }
        break;
}
```

Design Process

MACQUARIE University
SYDNEY · AUSTRALIA

1. Identify the main states of the system
2. What events (inputs) causes the system to change states?
3. What outputs are produced at each state?
   - If output is complex (more than just on/off), leave as a subsystem and pass a control flag
4. Construct state diagram and table showing the relationship between inputs, states and outputs
5. Repeat steps 1-4 for subsystems identified in step 3
   - An input to this subsystem would be the control flag
6. Draw a system block diagram showing the relationship between subsystems
   - Connect subsystem blocks by showing signal flows
7. Identify resources needed to implement system
8. Write your code
   - For each subsystem, write as a function
   - Write the code for the system state diagram in the main loop

Minor Project

MACQUARIE University
SYDNEY·AUSTRALIA

DRAW THE STATE DIAGRAM AND CONSTRUCT THE STATE TRANSITION TABLE

Pass Specifications

Operation Requirements

The general operations of the stopwatch are as follows. The initial state of the stopwatch is 0 minutes, 0 seconds and 0 fractions (1/10th) of a second. The stopwatch should start counting upwards from its initial state when the start/stop button is pressed. While counting up, pressing the start/stop button will pause the count and the time the button is pressed should be displayed. Pressing the start/stop button again should resume the count. Pressing the reset button should reset the stopwatch to its initial state.

The 7-segment display will indicate the current time of the stopwatch. The time should be shown with the format M.SS.f, where the left-most digit shows minutes (M) passed, the middle two digits show the seconds (SS), and the right-most digit for fraction of a second (f). The dot (.) is the decimal point on the 7-segment display. In its initial state, the display should show 0.00.0.

Button S1-A1 is used to reset the stopwatch to its initial state.

- Pressing S1-A1 at any time should stop the count and reset the display.

Button S2-A2 is used to start or pause/stop the stopwatch.

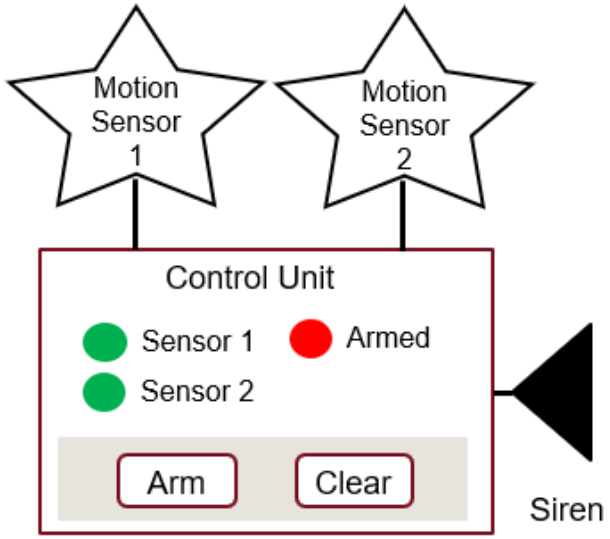
- If the stopwatch is not currently counting, pressing S2-A2 once will start the stopwatch counting up.
- If the stopwatch is currently counting up, pressing S2-A2 will pause the count and display the time when the S2-A2 was pressed.
- If S2-A2 is pressed and the display is showing a time other than 0.00.0, it should resume counting upwards.
- Pressing S2-A2 after a reset should start the count from 0.00.0.

LED D4 is used to indicate that the stopwatch has been paused.

- When the stopwatch is in its initial state, D4 should be off.
- When counting up, D4 should be off.
- When paused, D4 should be on.

Home alarm system

- System consists of:
  - Two motion sensors
  - Two buttons
  - Three status LEDs
  - Alarm siren



- The system should start with the alarm disarmed, and all LEDs and siren are off.
- When one of the motion sensors is triggered, the corresponding LED should light up and remain active for as long as motion is detected.
- To arm the system, the user should press the Arm key. This will put the system into the triggered state. During the triggered state, the Armed LED should flash at a rate of 1 Hz. The triggered state lasts for 5 s, after which the system will move into the armed state. In the armed state, the Armed LED should be continuously lit.
- During the armed state, if either of the motion sensors are triggered, it should set off the siren after a 5 second delay. The siren should sweep through frequencies 400 to 2000 Hz every two seconds.
- To stop the siren or to disarm the system, a user should press the Clear button.