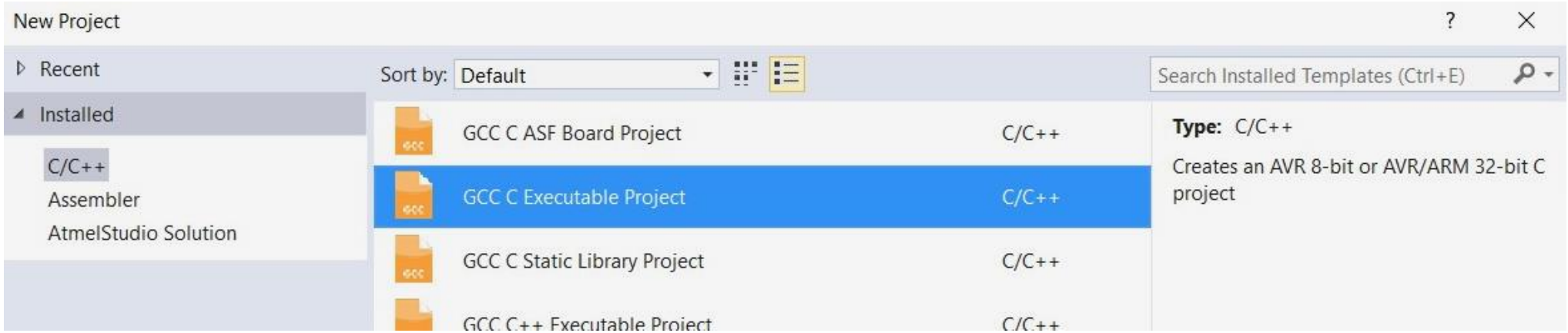
C Project

Ensure you select the ‘GCC C Executable Project’ type

This will matter in later weeks as we start to interface with more complex subsystems of the lab board



Multiplexer

- Parallel Port A is Multiplexed
  - Allows us to connect multiple sources to a single port
- PE1/PE0 control the Multiplexer
  - Setup the Multiplexer in your Setup function, setting PE1/0 to the appropriate levels based on the problem

```
// Setup Toggle Switches

void setup(void) {
    DDRE = 0b00000011;
    PORTE = 0b00000000;
}
```

See Appendix C (Lab Book Pg. 49) for more information about the Multiplexer

Delays

- Define the CPU Frequency
  - #define F_CPU 8000000UL – Define before calling the `include`
- Include the Delay header file
  - #include <util/delay.h>
- Call the function in your code
  - _delay_ms({time});

```
// Ex. Toggle Light @ 5Hz

#define F_CPU 8000000UL
#include <util/delay.h>

int main(void)
{
    setup();
    while(1) {
        toggleLight;
        _delay_ms(100);
    }
}
```

Delays are briefly covered on Pg. 12 of the Lab Book

State Machines

- State Machines use distinct States
  - Each state should reflect a different output state on the microprocessor
- State Machines will navigate to the state ‘expression’ each time it is run
  - Ensure ‘`expression`’ is defined outside of the switch and the `while(1)` loop
  - Otherwise, the value will reset every time!
- The ‘`switch`’ will keep running the current case until ‘`expression`’ has changed to a different value
  - Ensure each state has a way of exiting to another state.

```
// Example State Machine with two States
#define firstCase   1
#define secondCase  2

int expression = firstCase;

int main(void)
{
    setup();
    while(1)
    {
        switch (expression)
        {
            case firstCase:
                /* code to run in this
                   state */
                /* check conditions to see if
                   state should be switched */
                break;

            case secondCase:
                /* code */
                break;
        }
    }
}
```

State Machines

```
// Example State Machine with two States
#define firstCase   1
#define secondCase  2

int expression = firstCase;

int main(void)
{
    setup();
    while(1)
    {
        switch (expression)
        {
            case firstCase:
                /* code */
            break;

            case secondCase:
                /* code */
            break;
        }
    }
}
```

```
// State Machine adapted to use If Statements
#define firstCase   1
#define secondCase  2

int expression = firstCase;

int main(void)
{
    setup();
    while(1)
    {
        if (expression == firstCase)
        {
            /* code */
        }
        else if (expression == secondCase)
        {
            /* code */
        }

    }
}
```

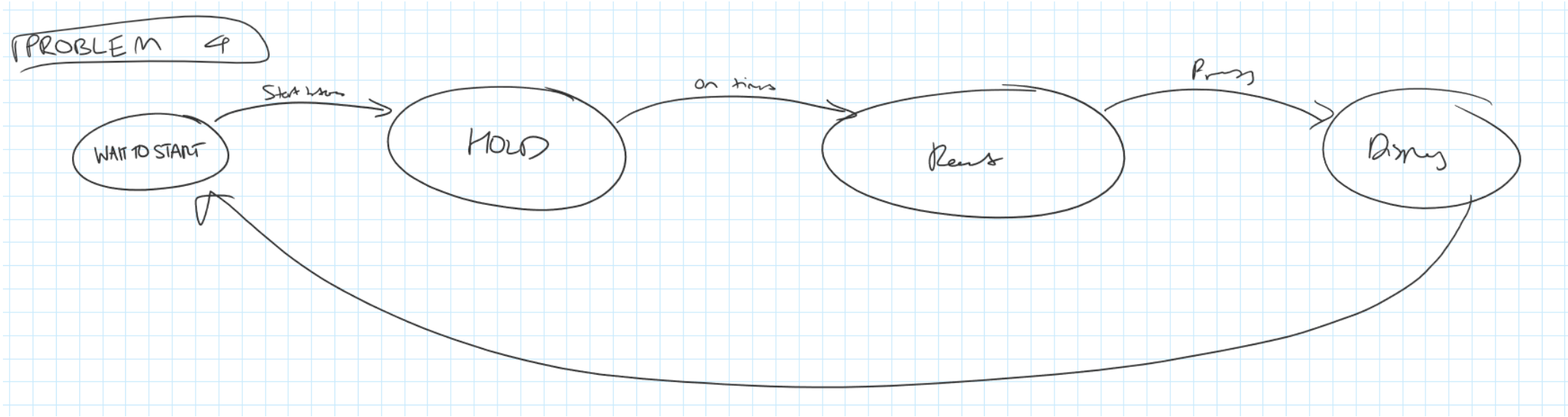
Example State Machine

Problem 4 – Reaction Timer:

The reaction timer tests the reaction time of two opponents. The game begins when the start button is pressed, then a buzzer will sound in a random time frame between one and five seconds. The two users will respond to the sound of the buzzer by pressing their respective buttons. The person who presses their button first after the buzzer sounds will win, indicated by their associated light coming on for three seconds and then their reaction time displayed on PORTC. The game will begin again when the start button is pressed.

Example State Machine

Problem 4 – Reaction Timer



Example State Machine

Problem 4 – Reaction Timer