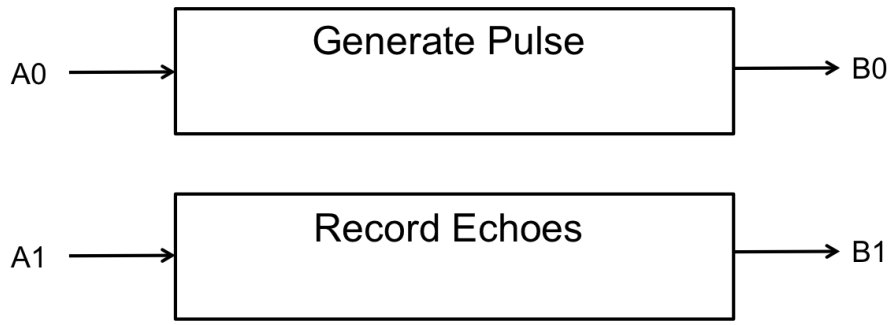
Name: _____

PES, Section 5.1 – 5.7

Concurrent SynchSMs, Keeping Distinct Behaviors Distinct, Converting Multiple SynchSMs to C, Converting SynchSM Local Variables to C, Converting Different-period Tasks to C, Creating a Task Structure in C, Simple Cooperative Task Scheduler

An active SONAR system for underwater object detection is implemented as a concurrent synchSM with two tasks, as shown below.



These exercises will examine conversion to C under varying assumptions about how the two tasks are implemented.

Name: _____

1. Fill in the four blank lines to complete the C implementation of the SONAR system, under the following assumptions:
   - Both tasks have a 200 ms period
   - The "Generate Pulse" task has two local variables: `unsigned char i, pulse;`
   - The "Record Echoes" task has one local variable: `unsigned char echoes;`

```
#include "RIMS.h"
volatile unsigned char TimerFlag=0;
void TimerISR() { TimerFlag = 1; }

enum GP_States { ... } GP_State;
void GP_Tick() {

  __static__ unsigned char i, pulse;

    ... // Standard switch statements for SM
}

enum RE_States { ... } RE_State;
void RE_Tick() {

  __static__ unsigned char echoes;

    ... // Standard switch statements for SM
}

void main() {
  B = 0;
  TimerSet(200);
  TimerOn();
  GP_State = GP_Start;
  RE_State = RE_Start;

  while (1) {

        ____ GP_Tick();___

        ____ RE_Tick();___

        while (!TimerFlag){}   // Wait for timer period
        TimerFlag = 0;         // Lower flag raised by timer
  }
}
```

Name: _____

2. Complete the C implementation of the SONAR system, under the following assumptions:
   - The “Generate Pulse” task has a 200 ms period.
   - The “Record Echoes” task has a 50 ms period.

```
#include "RIMS.h"
volatile unsigned char TimerFlag=0;
void TimerISR() { TimerFlag = 1; }

enum GP_States { ... } GP_State;
void GP_Tick() { ... }

enum RE_States { ... } RE_State;
void RE_Tick() { ... }

void main() {

 unsigned long GP_elapsedTime = 0;
 unsigned long RE_elapsedTime = 0;
 const unsigned long timerPeriod = 50;


 B = 0;
 TimerSet(50);
 TimerOn();
 GP_State = GP_Start;
 RE_State = RE_Start;

 while (1) {

      if (GP_elapsedTime >= 200) {
           GP_Tick();
           GP_elapsedTime = 0;
      }
      if (RE_elapsedTime >= 50) {
           RE_Tick();
           RE_elapsedTime = 0;
      }
      while (!TimerFlag){}   // Wait for timer period
      TimerFlag = 0;         // Lower flag raised by timer
      GP_elapsedTime += timerPeriod;
      RE_elapsedTime += timerPeriod;



 }
}
```

Name: _____

3. Complete the C implementation of the SONAR system, under the following assumptions:
   - The “Generate Pulse” task has a 200 ms period.
   - The “Record Echoes” task has a 50 ms period.

   This time, use the task struct and cooperative task scheduler.

```
#include "RIMS.h"

typedef struct task {

  ____ int state ____________________;

  ____ unsigned long period_________;

  ____ unsigned long elapsedTime ___;

  ____ int (*TickFct)(int) _________;
} task;

task tasks[tasksNum];
const unsigned char tasksNum = 2;
const unsigned long tasksPeriodGCD =          _50__;

const unsigned long periodGeneratePulse =     _200_;

const unsigned long periodRecordEchoes =      _50__;


enum GP_States { ... } GP_State;
_int__ GP_Tick(__int state __) { ... }

enum RE_States { ... } RE_State;
_int_ RE_Tick(___int state ____) { ... }


void TimerISR() {
     unsigned char i;
     for (i = 0; i < tasksNum; ++i) {

          if ( __tasks[i].elapsedTime >= tasks[i].period__ ) {

               tasks[i].state =
                         __tasks[i].TickFct(tasks[i].state)__;

               tasks[i].elapsedTime = __________0____________;
          }

          __tasks[i].elapsedTime__ += tasksPeriodGCD;
     }
}
```

Name: _____

```
int main() {

  unsigned char i=0;

  tasks[i]._state___________ = __GP_Start_____________________;

  tasks[i]._period__________ = __periodGeneratePulse__________;

  tasks[i]._elapsedTime_____ = __tasks[i].period______________;

  tasks[i]._TickFct_________ = __&GP_Tick_____________________;

  ++i;

  tasks[i]._state___________ = __RE_Start_____________________;

  tasks[i]._period__________ = __periodRecordEchoes___________;

  tasks[i]._elapsedTime_____ = __tasks[i].period______________;

  tasks[i]._TickFct_________ = __&RE_Tick_____________________;


  TimerSet(_______50________);
  TimerOn();

  while(1) { Sleep(); }
  return 0;

}
```