Shared Data Problem

- Code monitors two temperatures
- Temperatures must be equal
- If not -> ALARM
- vReadTemps() is the ISR, called periodically

Behavior:

The code occasionally sets off the ALARM, even if everything seems normal. ???

```
static int iTemps[2];
void interrupt vReadTemps() {
  iTemps[0] = // read in value from HW;
  iTemps[1] = // read in value from HW:
}
void main(void) {
  int iTemp0, iTemp1;
  while (TRUE) {
     iTemp0 = iTemps[0];
     iTemp1 = iTemps[1];
     if (iTemp0 != iTemp1) {
        // Set off ALARM
     }
  }
}
```

Shared Data Problem

Sequence of events:

1. main: iTemp0 = iTemps[0];
2. ISR: updates iTemps[0] and iTemps[1]
3. main: iTemp1 = iTemps[1];
4. main: iTemp0 != iTemp1 → ALARM!

Alternative main(): $\longrightarrow$

```
....
void main(void) {
while (TRUE) {
    if (iTemps[0] != iTemps[1]) {
      // Set off ALARM
    }
  }
```

Does it fix the problem? **NO!**

IT always comes at the wrong time.

Shared Data Problem

Source of the problem:

iTemps[] array is shared between the main() and the ISR. If IT happens while main() is using the array -> the data may be in an inconsistent state.

Solving the problem:

Enable/disable ITs

Atomic/critical section →

```
…
void main(void) {
  int iTemp0, iTemp1;
  while (TRUE) {
     disableIT();
     iTemp0 = iTemps[0];
     iTemp1 = iTemps[1];
     enableIT();
     if (iTemp0 != iTemp1) {
        // Set off ALARM
     }
  …
```

Interrupt latency

How fast will a system react to interrupts? Depends on:

1. Max. time while IT-s are disabled.
2. Max. time taken to execute higher priority IT-s.
3. Time taken by ISR invocation (context save, etc.) and return (context restore)
4. “Work” time in ISR to generate a response.

Values:

For 3: see processor docs.

Others: count instructions – does not work well for processors with cache!

General rule: WRITE SHORT IT SERVICE ROUTINES!

Alternative to disabling IT-s

```
int iTempAs[2];
int iTempBs[2];
bool fUsingB = FALSE;
void interrupt vReadTemps() {
    if(fUsingB) {
            iTempAs[0] = // read from HW
            iTempAs[1] = // read from HW
    } else {
            iTempBs[0] = // read from HW
            iTempBs[1] = // read from HW
    }
}
```

Two sets of variables

One flag to control which set is used

Alternative to disabling IT-s

```
void main () {
    while (TRUE) {
          \boxed{if(fUsingB) {}
             if (iTempBs[0] != iTempBs[1]) {
                // set off ALARM
             }
          } else {
             if (iTempAs[0] =! iTempAs[1]) {
                // set off ALARM
             }
          }
          fUsingB = !fUsingB;
    }
}
```

<u>Assumption</u>:

Changing the flag is an atomic operation!