High Level SW Algorithm Heart Rate Monitor

Program Registers speaking with Hardware:

1. Heart Rate Monitor Reading Address (HRT\_BEAT\_READING)

Registers Containing Data for Output:

1. Heart Rate BPM Register. (HRT\_RATE\_AVG\_OUT)
2. Peak Heart Rate Register. (HRT\_RATE\_MAX\_OUT)
3. Minimum Heart Rate Register. (HRT\_RATE\_MIN\_OUT)

Registers for program data:

1. Heart Beat Counter. (HRT\_BEAT\_COUNT)
2. Heart Beat Average. (HRT\_BEAT\_AVG)
3. Total 6 second samples gathered. (HRT\_TOTAL\_SMPLS)
4. Current BPM. (HRT\_CURR\_BPM)
5. Peak Heart Rate. (HRT\_RATE\_MAX)
6. Minimum Heart Rate. (HRT\_RATE\_MIN)

INIT:

1. Initialize variables, set registers to zero.
   1. Need to decide what registers we set to zero. For example, minimum and max heart rate values.
   2. Can add a separate routine label that won’t overwrite those registers.
2. Transition to Wait\_for\_pulse().

Wait\_for\_pulse():

1. Read HRT\_BEAT\_READING
2. If reading not found:
   1. Set HRT\_RATE\_AVG to zero.
3. If reading is found:
   1. Transition to Get\_heart\_beats() state.

Get\_heart\_beats():

1. Set HRT\_BEAT\_COUNT to zero.
2. Call Get\_time() routine.
3. Save return value in a register.

Get\_heart\_beats\_loop():

1. Call get\_time() routine, save value in an unused register (temporary works).
2. While the total time elapsed is less than 6 seconds:
   1. Load value from HRT\_BEAT\_READING
   2. If reading indicates a heartbeat:
      1. Increment HRT\_BEAT\_COUNT
   3. Jump back to Get\_heart\_beats\_loop().
3. If timer above 6 seconds and HRT\_BEAT\_COUNT is zero:
   1. Jump to Wait\_for\_pulse().
   2. Initiate Life Alert.
4. If timer above 6 seconds, and HRT\_BEAT\_COUNT is greater than zero:
   1. jump to BPM\_calculation().

BPM\_calculation:

How should this be done? We need a floating-point unit to get an accurate result using RR interval.. The best way may be to get 6 second intervals. Values less than 1 and greater than 0 will be rounded so I can’t use the RR interval for that case. For Now, I will use the 6 second method.

1. Multiply HRT\_BEAT\_COUNT by 10 and store to HRT\_CURR\_BPM.
2. Increment HRT\_TOTAL\_SMPLS
3. Add HRT\_CURR\_BPM to HRT\_BEAT\_AVG
4. Divide HRT\_BEAT\_AVG by HRT\_TOTAL\_SMPLS, store to HRT\_BEAT\_AVG.
5. If HRT\_CURR\_BPM is the highest BPM seen:
   1. Save to HRT\_RATE\_MAX
6. If HRT\_CURR\_BPM is the lowest BPM seen:
   1. Save to HRT\_RATE\_MIN
7. Write HEART\_BEAT\_AVG to HRT\_BEAT\_AVG\_OUT.
8. Write HEART\_BEAT\_MAX to HRT\_BEAT\_MAX\_OUT.
9. Write HEART\_BEAT\_MIN to HRT\_BEAT\_MIN\_OUT.
10. Return to Get\_heart\_beats().

Get\_time():

https://stackoverflow.com/questions/21532972/evaluating-time-elapsed-between-two-actions-in-mips-assembly-language/21533487

https://courses.missouristate.edu/KenVollmar/MARS/Help/SyscallHelp.html

1. Load “$v0” with value 30.
2. Call “syscall”.
3. $a0 and $a1
   1. $a0 has low order 32 bits of system time.
   2. $a1 has high order 32 bits of system time.