How 999xx shows up in IRIG-B reading

IRIG-B 9999xx is showed up something like this;

1. Timing slave sends a gate signal along 1PPS signal.
2. Clocks start and are sent to ADC and DAC, but ADC starts 1 cycle (16usec) faster somehow, probably by glitch, noise or something.
3. Realtime loop starts 1 cycle faster as a result by receiving the signals from the ADC that started 1cycle faster.
4. The real time code reads GSP time with 1 cycle faster timing
5. Then it shows 9999xx on IRIG-B.

How to fix?

There are two solutions;

- One is hardwarely to modify to have ADC start slower. However It is not so easy to trace FPGA code on timing slave, it is not so convenient to modify each backplane board or each ADC adapter card.
- Another solution is to add a delay of 1cycle on the realtime code, even ADC runs 1 cycle faster already.

Changes in RCG source code

```
L145;
int irigberror = 0;

L1535~
/// ¥> Cycle 1 and Spectricom IRIGB (standard), get IRIG-B time
information.
      if(cycleNum == HKP_READ_TSYNC_IRIBB)
      {
            if(cdsPciModules.gpsType == TSYNC_RCVR)
            {
                  gps_receiver_locked = getGpsuSecTsync(&usec);
                  pLocalEpics->epicsOutput.irigbTime = usec;
/// - ---- If not in acceptable range, generate error.
                  if((usec > MAX_IRIGB_SKEW) || (usec < MIN_IRIGB_SKEW))
                  {
                        feStatus |= FE_ERROR_TIMING;;
                        diagWord |= TIME_ERR_IRIGB;;
                        irigberror = 1;
                  } else {
                        irigberror = 0;
                  }
            }
      }
```

Changes in RCG source code

L2061~

```
	// Avoid calculating the max hold time for the first few
seconds
	if (cycleNum != 0 && (startGpsTime+3) < cycle_gps_time) {
		if(adcHoldTime > adcHoldTimeMax) adcHoldTimeMax = adcHoldTime;
		if(adcHoldTime < adcHoldTimeMin) adcHoldTimeMin = adcHoldTime;
		adcHoldTimeAvg += adcHoldTime;
		if (adcHoldTimeMax > adcHoldTimeEverMax)  {
			adcHoldTimeEverMax = adcHoldTimeMax;
			adcHoldTimeEverMaxWhen = cycle_gps_time;
			//printf("Maximum adc hold time %d on cycle %d gps %d¥n",
adcHoldTimeMax, cycleNum, cycle_gps_time);
		}
		if (timeHoldMax > cpuTimeEverMax)  {
			cpuTimeEverMax = timeHoldMax;
			cpuTimeEverMaxWhen = cycle_gps_time;
			if(irigberror) cycleNum -= 1;
	}
```

Compensation works only in 3 seconds

- This compensation is performed only within first 3 seconds after the real time code started. In this term, error checking codes are ignored for cycle errors or something.
- After 3seconds, the compensation won't work to distinguish actual failures of timing or other troubles while the real time code is running.
- You can see directly changing the IRIG-B value from 9999xx to 12 over the few seconds.

- IRIG-B issue was gone with this modification. Currently I do not see any problem caused by this mondification.

CPU-max issue

- We are trying to reproduce CPU max issue using a test bench.
- Almost of RT PC in the real KAGRA system has red indicator in CPU MAX.
- Glitch happens every 10 min or something.
- Hyper thread is already off.

Test bench

- 2 RT PCs, with IO chassis.
  - some ADC/DAC/BO/BIO, but no AA/AI
- PXE boot server.
- DAQ: 1 flow of DC, NDS, FW.
- TDS with GPS signal.

The same things

- the number of models
- the number of ADC, DAC, BIO cards
- Length of optical fiber cable: ~100 meter
- Some random output (~3000) on DAC signals.

Differences

- Computer: Slow(2.6GHz) and fast(3.0GHz)
- Version of RGC: 2.9.3 vs. 3.1.1
- ADCs read large amplitude vs. ~0 value

Next step

- We do not see any CPU glitch in the test bench.

- We will computer to slower one for the test bench.
- We are also suspecting terrible traffic on DAQ network, or TCP/IP like EPICS, camera…