Find out what is causing a PMAC Watchdog Timer Failure

The PMAC motion control board has an on-board watchdog timer (sometimes called a dead-man timer or a get-lost timer) circuit whose job it is to detect the conditions that can result in dangerous malfunctions, and shut down the card to prevent a malfunction. The philosophy behind the use of this circuit is that it is safer to have the system not operate at all than to have it operate improperly.

Because the watchdog timer wants to fail and many components of the board, both hardware and software, must be working properly to keep it from failing, it may not be immediately obvious what the cause of a watchdog timer failure is. This application note is a guide to examining the possible causes of watchdog failure.

How the Watchdog Timer Works

The hardware circuit for the watchdog timer requires that two basic conditions be met to keep it from tripping. First, it must see a DC voltage greater than 4.75V. If the supply voltage is below this value, the circuit’s relay will trip. This prevents corruption of registers due to insufficient voltage. The second necessary condition is that the timer must see a square wave input (provided by the PMAC software) of a frequency greater than approximately 25 Hz. If the card, for whatever reason, due either to hardware or software problems, cannot set and clear this bit repeatedly at this frequency or higher, the circuit’s relay will trip.

In the PMAC software, the task that actually writes to the watchdog timer is called the real-time interrupt (RTI). This task is supposed to execute at a fixed number of servo cycles. The number of servo cycles is determined by variable I8 – the RTI executes every I8+1 servo cycles. At the default value of 2, this is every third servo cycle. The servo cycle frequency is divided down from the master clock by an amount determined by the settings of jumpers E98, E29-E33, and E3-E6. At the default settings, the servo frequency is 2.25 kHz, so the default RTI frequency is 750 Hz. If the RTI frequency were to drop below about 50 Hz, it could not provide the 25 Hz square wave and the timer would trip. With the default servo update rate, an I8 value greater than about 30 will cause the timer to trip.

Every RTI, PMAC reads the 12-bit watchdog timer register (Y register $1F) and decrements the value by 8 – this toggles bit 3. If the resulting value is not less than zero, it copies the result into a register that forces the bit 3 value onto the watchdog timer. Repeated, this process provides a square-wave input to the watchdog timer.

If no other task intervened, the RTI would stop toggling the watchdog input after 512 cycles (2^12/8) because the timer register would have counted down to zero. The task that keeps this from happening is the housekeeping function in PMAC’s background tasks. The background tasks execute in the time available between higher priority tasks such as servo cycle updates and RTI tasks. The three main background tasks are responding to host commands, executing PLC programs (1-31), and housekeeping. In the background, PMAC executes one scan through an individual PLC program, then checks to see if there are any complete commands, responding if there are, then executes the housekeeping functions. This cycle is repeatedly endlessly.

Most of the housekeeping functions are safety checks such as following error limits and overtravel limits. When it is done with these checks, PMAC sets the 12-bit watchdog timer register back to its maximum value. As long as this occurs regularly at least every 512 RTI cycles, the watchdog timer will not trip.

The purpose of this two-part control of the timer is to make sure all aspects of the PMAC software are being executed, both in foreground (interrupt-driven) and background. If anything keeps either type of routine from executing, the watchdog will fail quickly.
What to Look for When the Watchdog Trips

The things that can trip the watchdog timer can be divided into four basic classes: constant (or quickly repeatable) hardware problems, intermittent hardware problems, constant (or quickly repeatable) software problems, or intermittent software problems. The following procedure should help you determine which type of problem it is.

If it appears that the watchdog timer has tripped – usually this is suspected when the outputs all turn off and communications to the host is lost – check the LEDs on PMAC. If both the red and green LEDs are on, the watchdog timer has tripped. If both LEDs are off, all 5V power to the card has been lost. If the green LED is on, and the red LED is off, the watchdog timer has not tripped; there is some other problem.

If the watchdog timer has tripped, reset the card either by taking the INIT/ line on the control panel connector low, then high, or by cycling power. If the red LED stays lit after the reset, you have a constant problem; if it goes off, you have an intermittent problem.

**Constant Problem**

If the red LED stays lit on reset, determine whether the constant problem is in hardware or software. The best way to determine this is to turn the power off, change jumper E51 from its default state (take OFF for PMAC-STD, put ON for all others), then turn power back on. This re-initializes PMAC, disabling all software settings and programs that could have tripped the timer. If the red LED stays on, there is a constant hardware problem; if it goes off, there is (or at least was) a constant software problem.

**Note:**

Re-initializing the card works by installing the factory-default I-variable values, instead of the values stored in EAROM. The values that are stored in EAROM are not lost (unless the default values are saved); they are not being used. If the hardware or jumper setting problem is fixed that was causing the watchdog trip, recover the old values simply by powering up the card with E51 in its default state.

**Constant Hardware Problem**

There are several possible causes of a constant hardware problem. The first is low supply voltage. With a voltmeter, probe between +5V and GND on one of the connectors at the card – for example, between pins 1 and 3 on the machine connector. If variation in the supply is suspected, use an oscilloscope to catch changes.

If the supply looks OK, inspect the card itself. Turn off power and remove the card. If there are several circuit boards, make sure that the daughter boards are well seated on the mother board. Look at all the socketed ICs. Make sure that they are firmly in their sockets, with no bent or broken legs. Apply reasonable pressure to each socketed IC. Re-install the card and turn on power.

If the red LED is still on, turn off power and remove the card again. Check the board thoroughly for signs of damage: broken components, damaged circuit board, cracked solder joints, etc. Any of these types of problems will require a factory repair.

**Constant Software Problem**

If using the re-initialization jumper setting on E51 allows the card to power up without tripping the watchdog timer, the problem that caused the tripping was in software, usually caused by a particular software function taking too much of the processor’s time. Now determine which function is causing the problem.

PMAC requires at least 55 microseconds in the servo update time per activated motor. If there are too many active motors and a fast servo update time, the lower-priority tasks may be starved for time including watchdog update and shut down the clock. At the default update rate of 2.25 kHz, it is possible to have all eight motors active. When the card is re-initialized, only motor 1 is left activated (I100=1; I200-I800=0), so the card can run successfully at any servo update rate.
If it is suspected that this may be the problem, check what the servo update rate is. First, calculate it from the master clock frequency and the settings of jumpers E98, E29-E33, and E3-E6. Second, confirm the frequency setting by probing with an oscilloscope the SERVO clock signal on the serial port connector. If the servo clock frequency is too fast for the number of activated motors wanted, slow down the servo clock by changing jumper settings before re-activating all of the motors.

If the servo update frequency does not appear to be the problem, a PLC program probably is. Re-initialization solves this problem by setting I5 to 0, not permitting any PLC programs to run. PLC 0, which runs as an RTI task, is the most likely culprit, because it can starve the background tasks for time, preventing the housekeeping routine from resetting the timer register. If PLC 0 is suspected, delete it now (OPEN PLC 0 CLEAR CLOSE), turn off power, change E51 back to its default setting, and turn power back on. If the watchdog does not trip, it is confirmed that the PLC 0 was the problem.

In general, PLC 0 should be kept very short, and usually the shorter the better. PMAC will execute an enabled PLC 0 every RTI, unless it has not finished the RTI tasks (PLC 0 and motion program) that it started in a previous RTI cycle. If a given PLC 0 takes 95% of the time available for it in one RTI cycle, all of the background tasks put together have to run in the remaining 5%, which may not let them keep the watchdog timer happy. Shortening the PLC 0 slightly, so that it only takes 85% of available RTI time, would allow background tasks to run three times faster, because they would now have 15% of available time. However, lengthening the PLC 0 slightly, so that it takes 105% of the time available in a single RTI cycle, would cause it to execute only every other RTI cycle and leave the majority of the second cycle available for background time.

For a background PLC program (PLC 1 to 31) to cause a watchdog failure, it must be extremely long – probably thousands of lines long, unless it has virtually no time to execute, in which case the real problem is a higher priority task. In general, however, if there is a lot of PLC code, it is better to split it up into several separate PLC programs, because other background tasks, such as communications response and safety checks, will operate more often.

**Intermittent Problem**

If simply resetting the card without re-initializing it causes the card to operate without immediately tripping the watchdog timer, you have an intermittent problem in either hardware or software. As we all know from trying to show a car’s problem to a mechanic, intermittent problems can be much more difficult to track down.

The key in finding an intermittent problem is identifying the pattern of failure. One of the key questions to ask yourself is, "What has changed since the system worked?". Try changing things back to the way they were before you were getting the failure.

Many of the possible causes of intermittent watchdog failure are fundamentally the same as the constant problem causes listed above, but only occurring on occasion. For example, a power supply may provide inadequate voltage only when other components draw a heavy load. An important electrical contact may fail intermittently. A PMAC program may take too much time only when it is in a particular section or logical branch.

If a failure occurs repeatedly in the same section of operation of the system, the cause is probably a software time problem (unless this section cuts the supply voltage by drawing increased load). Find out what is special about the software at this point. For instance, maybe PLC 0 had been disabled and the watchdog tripped when PLC 0 was enabled. Alternately, maybe PLC 0 had been passing over most of its calculations because of the conditional branch it was taking, but the card failed when the condition changed.
Although it is very rare for a motion program to cause a watchdog failure, this does happen on occasion. It is important to understand how the motion program executes. When a **RUN** command is given and every time the actual execution of programmed moves progresses into a new move, a flag is set saying it is time to do more calculations in the motion program for that coordinate system. At the next RTI, if this flag is set, PMAC will start working through the motion program. Program calculations will continue (which means no background tasks will be executed) until one of the following conditions occurs:

1. The next move or dwell is found and calculated.
2. End of or halt to the program (e.g. **STOP**) is encountered.
3. Two jumps backward in the program (from **ENDWHILE** or **GOTO**) are performed.
4. A **WAIT** statement is encountered (usually in a **WHILE** loop).

If calculations stop on condition 1 or 2, the calculation flag is cleared, and will not be set again until actual motion progresses into the next move (1) or a new Run command is given (2). If calculations stop on conditions 3 or 4, the flag remains set, so calculations will resume at the next RTI. In these cases, where there is an empty (no-motion) loop, the motion program acts much like a PLC 0 during this period. These empty loops, usually which are used to wait for a certain condition, provide very fast response to the change in condition, but their fast repetition occupies a lot of CPU time, and can starve the background tasks for time. Particularly if several coordinate systems are executing empty loops at the same time, serious background time limitations are likely, which can be severe enough to trip the watchdog timer.

If there are a huge number of lines of intensive calculations (e.g. 100) before any move or dwell is encountered, there can be such a long time before background calculations are resumed (more than 512 RTI cycles) it is possible to trip the watchdog timer. If this problem occurs, the calculations should be split apart with short **Dwell** commands to give other tasks time to execute.