The 8052.com guide
to reset, supervisor, watchdog
and other pets.

Thanks to
Erik Malund, Kai Klaas, Peter Dannegger, Oleg Sergeev, Oliver Sedlacek,
Ijaz Ahmed, Richard Erlacher, Neil Kurzman, Lynn Reed;
the whole 8052.com community;
and, of course, Craig Steiner, The Webmaster.

1. Reset

1.1 So, what animal is that reset?

Microcontrollers - such as the 8051 - and processors are full of sequential logic - registers, flip-flops - and after powerup they are in random state. However, it is vital that a processor or controller starts up in a known state - at least, it should start from a defined point of the program (which means that the program counter should have a defined content at startup - e.g. for '51 it starts from zero). In microcontrollers, most of the integrated peripherals (e.g. I/O pins) should have a defined initial state, too.

To accomplish this function, there is a signal distributed within the chip to these registers - the reset signal - which should be activated during powerup. Traditionally, this signal is input from outside the chip, via a dedicated reset pin; but in the last years reset (and associated) circuits integrated into the microcontroller gain increased popularity (less external components, spared pin).

1.2 What else does it do?

During powerup and powerdown, when the supply voltage is lower than needed for proper operation, “holding” the controller in reset prevents it from performing “random” uncontrolled actions, which would cause data corruption or damage in attached peripherals. Also during operation, reset should react to abnormalities in power supply (glitches, undervoltage (brownout)).

Reset can be also activated deliberately, when the processor/microcontroller has to be brought into a defined (initial) state - either by supervising circuits (watchdog) when program runaway occurs and is detected; or manually by the user. This is sometimes called warm reset (contrary to the powerup reset, called cold reset).

1.3 How does it look like?

Reset is a normal logic signal, and we can describe it by its polarity and duration.

1.3.1 Polarity

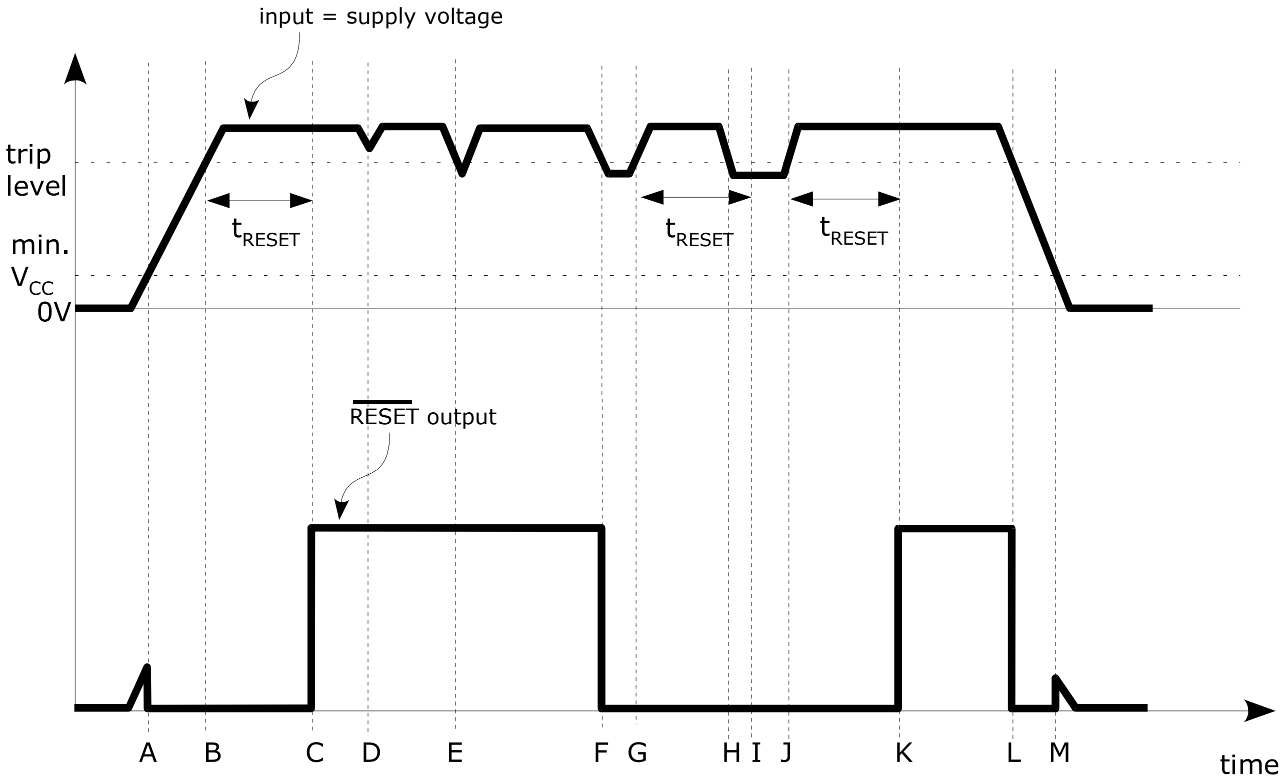
As the reset circuit should engage mainly while supply voltage is low, it sounds logical that it should be active low, as in that way it's level is well-defined even during “abnormal” supply voltage. Indeed, most of the reset signals are active low, the most notable exception, where reset is active high is... the 8051.

1.3.2 Duration

The simplest sequential circuits (e.g. the 74xx74 flip-flop) have asynchronous reset - it means, that the shortest reset pulse would bring it into a defined state, regardless of the clock input. However, the much more complex microcontrollers usually employ synchronous reset, so the reset pulse should last while one, or several clock transition occurs (depending on the particular design). In 8051, the reset pulse should be active for at least 24 oscillator cycles, for a valid reset to occur.

However, note, that internal oscillators of microcontrollers start upon powerup or upon active level of reset pulse after powerup. It takes some time for a crystal oscillator to start up and stabilise its oscillations. This time depends on the particular type of oscillator and crystal, and can be in the order of several tens of milliseconds. The reset pulse must be long enough to cover this time plus the required 24 oscillator cycles.

Some of the *integrated* reset circuits account for this and don't release until a certain number of valid clock cycles occurs (this number can be sometimes even adjusted). However, it is rare for external reset circuits to have such feature and it is the responsibility of the designer to ensure that the reset pulse will last long enough even for the worst case of oscillator startup. On the other hand, some applications are required to run as soon as possible after powerup - here, the choice of reset pulse duration is a matter of compromise.

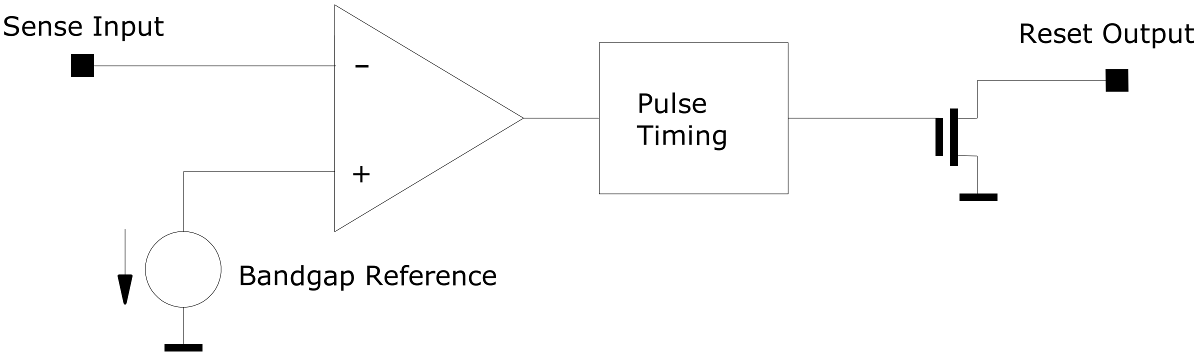


***Input and output of a reset IC -** This is a simple active low "three-legged" or similar reset IC. Until input voltage is below the minimum V_{CC} for proper function of the reset IC, the output transistor is closed and the output voltage follows the rising V_{CC}, until point A is reached (assuming that there is a pullup resistor on the output, or an integrated pullup in the microcontroller's reset input). There, the circuit starts to output a valid reset level. At point B, the V_{CC} reaches the trip level and the timing circuit (one-shot) starts to time the reset pulse. At point C, the timing circuit times out and the output is released. The glitch at point D is ignored, as it does not drop below the trip level. Although the glitch at point E drops below the trip level, it is so short (a few ns) that it gets suppressed. At point F, V_{CC} drops below the trip level and remains there long enough for a valid reset to be generated (brownout). At point G full power is recovered and the timing circuit starts, but at point H a repeated brownout prevents the reset to be released. The timing circuit restarts at point J where full V_{CC} is restored again and the reset remains active until point K. When V_{CC} starts to fall at shutdown, reset is tripped at point L, and remains active until at point M drops below the minimum $V_{CC,}$ where it is released. Note: in this simplified diagram only one trip level is shown, although usually there is a hysteresis introduced, so the trip level for rising V_{CC} is somewhat higher than that for falling $V_{CC.}$*

1.4 How is a reset circuit constructed?

To ensure proper reset pulse generation under various conditions (power-up, power-down, brownout), the reset circuit consists of:

- comparator - this compares the current supply voltage (or voltage on a dedicated “sense” input) with a stabilised voltage reference, and if it is below this reference, it flips the reset output;
- voltage reference - provides a precisely stabilised voltage for the comparator;
- one-shot or other circuit providing a timed pulse - this determines the duration of reset after the supply/input voltage reached the preset value;
- output circuitry - this conditions the reset output according to required polarity, output current etc.



The reset output gets active immediately after the comparator senses drop in the input voltage and remains active until the supply/input voltage regains the proper value plus a predetermined time period. Some reset circuits (usually the integrated) feature short glitches suppression, so they won't trip if the voltage drops below the trigger level for a very short time (to provide extra noise immunity).

1.4.1 Can the supply/input trip level be adjusted?

Usually, the reset ICs have a fixed trip level. However, they are manufactured in a wide selection of trip voltages, stepped in few tens of millivolts. Usually the trip voltage is either directly contained in the marking (as a suffix), or in encoded form (usually a letter in the suffix).

Note, that the actual trip voltage might depart from the nominal value by a few %, as a combination of manufacturing tolerance and temperature-dependent variation.

1.4.2 Can the reset pulse duration be adjusted?

The simplest reset circuits provide a fixed reset pulse duration, but some of them are manufactured in multiple “duration” grades. In more sophisticated circuits, either multiple choices can be made via connecting a dedicated pin to ground/supply/leave open, or they provide for pulse adjustment by having attached a capacitor or resistor, value of which determines the reset pulse length.

1.4.3 What are the options in outputs?

The output can be of two polarities: active low or active high; some circuits feature both.

The output circuit itself can be either of push-pull type or open collector (open drain). In the latter case, the output transistor can be on during active reset, or during the inactive (normal run) state.

One of the main concerns and “gray areas” is the output level when the supply voltage of reset IC itself drops below its specified minimum. Unfortunately, most reset ICs have no defined behaviour specified under these conditions. Some of the ICs simply release their output, so that an appropriate pullup or pulldown can provide a valid reset level.

A bullet-proof solution is to use a battery-powered reset IC, but this is impractical unless a battery is already provided for timekeeper or RAM backup; and also the IC choice has to be made based on its idle power consumption to preserve the battery (and the suitable reset ICs tend to be those of the higher-end both in performance and price).

1.5 What's wrong with RC reset?

Some microcontrollers' datasheets suggest, that it is enough to attach a RC circuit (or a single capacitor) to the reset input.

Generally, this is a wrong practice.

The main problem with RC reset (or a single capacitor, if the appropriate resistor is integrated into the reset input circuit in the microcontroller) is, that it depends on the circumstances. It provides valid powerup reset only if the power supply ramps up fast enough. Heavy power supply filtering might cause problems with this. Switched-mode power supplies sometimes tend to bounce upon powerup, which might cause short reset pulses when RC reset is used. Too short reset pulses (less than 24 oscillator cycles) may cause unexpected (undefined) behaviour of the microcontroller.

Another problem is, that the RC circuit does not act upon powerdown, nor upon power glitches, brownouts and rapid powerdown/powerup conditions. Although the microcontroller might not be harmed under these conditions, its proper running is not fully guaranteed and also it might bring unwanted problems with the attached peripherals. For example, if a CMOS microcontroller is not held in reset upon powerdown, it might continue to run at V_{CC} as low as 2V, while the possibly TTL-compatible peripherals stop to provide valid input levels at these supply conditions; the microcontroller will act upon invalid inputs and might either falsely start some actuators or store invalid values into attached nonvolatile memory (e.g. battery-backuped SRAM).

These problems get even more aggravated when the microcontroller's code memory is FLASH-based, as the FLASH might get easily corrupted (unintendedly rewritten or erased) when run under rapidly varying or low supply conditions.

Some of the modern microcontrollers do have a fully functional built-in reset circuit (see chapter 1.8); however, these then typically don't need the RC (or C) circuit at reset input at all.

1.6 How “sensitive” should the reset be?

Generally, it should “catch” any disturbance in supply voltage which would cause incorrect operation of the microcontroller and/or the surrounding circuitry. This does not mean that a reset has to be triggered upon each small glitch - most of the microcontrollers are immune to *certain* amount of noise in the supply line. Therefore, the internal resets sometimes have some amount of built-in noise immunity so they can suppress very short glitches (few tens of ns typically).

External resets of course don't know about the “noise immunity” of the microcontrollers, therefore tend to act pretty fast, reacting to a few ns glitches, although it is rare that a manufacturer specifies this feature in the datasheet. So to avoid unwanted spurious resets with external reset IC, it is vital that it has appropriately filtered VCC/sense input. On the other hand, if the reset IC has a dedicated sense input, heavy filtering of that input especially with RC filters results in degradation of the reset IC to the “unwanted” RC-reset.

1.7 What else is related to reset? What is typically integrated within a reset IC?

Resets are analog ICs of relatively low density of integration, so it makes sense to add simple features, usually of a similar analog character; some of them result in triggering a reset, too. However, each added feature tends to consume additional pins, which relates directly to increase of cost.

Such complex circuits then are called different names such as “supervisor IC”, “power management IC” and similar, but here we will stick to the simple “reset IC”.

For example of such IC see the ..69x family in chapter 3.

1.7.1 Reset input, pushbutton input

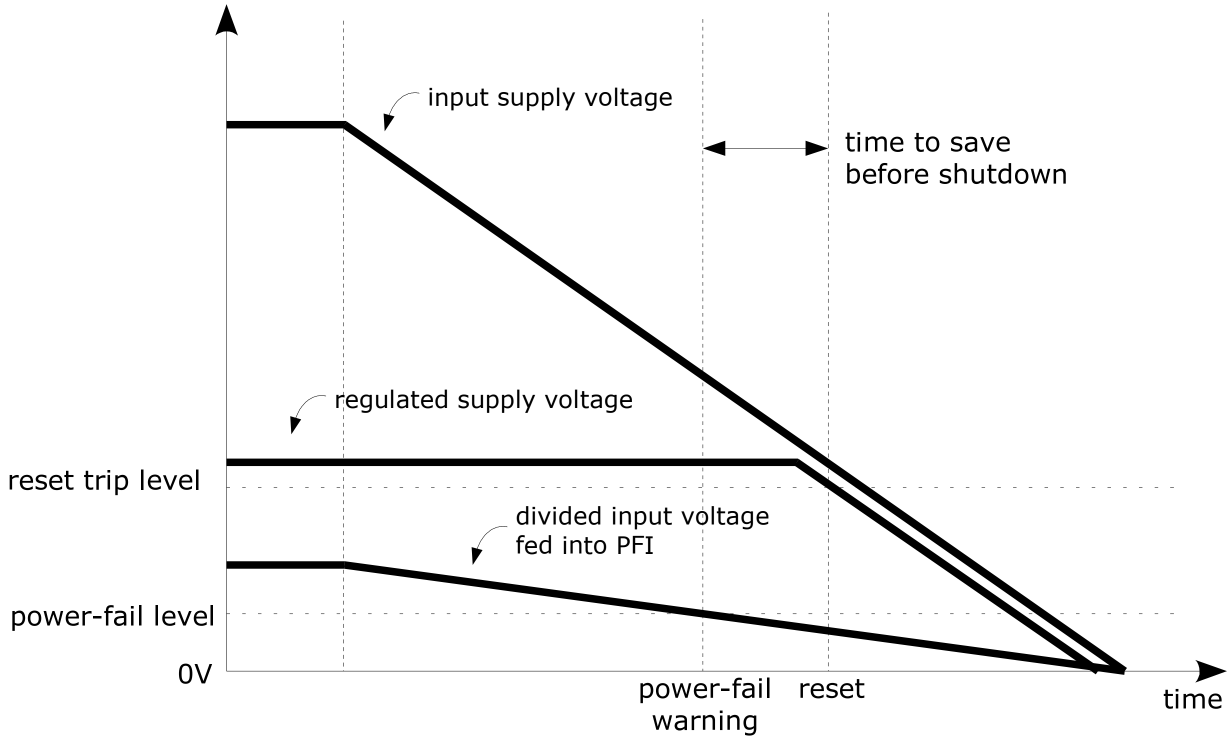
There might be other sources of reset on a microcontroller-based board, or an external reset might be needed; so a reset input on the reset IC can help to “merge” more such sources of reset. Another reason to have a reset input is to facilitate a pushbutton for manual reset. Reset inputs usually are active low, and have built-in pullups.

A nice (and patented) trick is employed in some of the newest Intersil reset ICs: the reset pushbutton is connected directly to the reset output, sparing down a pin from the package.

1.7.2 Early power-failure warning

In many applications, the microcontroller is powered from a voltage regulator, which is supplied by a higher unregulated voltage source. With a proper arrangement, it is possible to keep the regulated voltage at an appropriate level for some short but defined time, even when the primary power source was switched off. In that case, the unregulated voltage starts to fall, and if this decrease is detected, the microcontroller can be warned and can perform critical data save and/or housekeeping.

To accomplish such a feature, the reset IC can contain an another comparator, connected to the same voltage reference as the basic reset comparator, and usually to an extra input pin (PFI - powerfail input - to be connected to the monitored unregulated supply rail through a suitable voltage divider). The comparator output goes to a dedicated pin (PFO - powerfail output), to be connected to an interrupt pin of the microcontroller. The interrupt will then handle the early powerdown warning.



1.7.3 Battery switchover

In systems incorporating a battery-backuped SRAM, during undervoltage condition when the processor and/or address decoder and similar circuitry might operate incorrectly, it is vital that no spurious write occurs. As this function involves V_{CC} monitoring, it is natural to integrate it into a reset/supervisor IC.

Typically, such IC has a chipselect input, where the microcontroller's or address decoder's output is connected, and a chipselect output, driving the memory's chipselect. The reset IC gates the chipselect output when V_{CC} gets too low. Often, the memory itself is powered via the reset IC, which provides on a pin a glitchfree transition from external V_{CC} to battery supply.

The reset IC is powered from the battery, too, and even if it is designed to draw as little current as possible (a few μA), it has to be taken into account when determining the battery lifetime.

1.7.4 Timekeeping, integrated memory, other

From battery switchover there is only a small step to have integrated battery-powered devices: real-time clock and battery-backuped SRAM. In a somewhat bizarre combination some high-end reset ICs incorporate also EEPROM (FRAM), although it is not related to any of the functions of the reset IC itself.

1.7.5 Multiple voltage sensing, power sequencing

Modern microcontrollers and other high-integration ICs often use multiple voltages for their operation - usually lower voltage for their core to reduce power consumption at high operating frequencies, and higher voltage for IOs to have high noise immunity. Or, multiple supply voltage circuits may be used in a system for various reasons. For systems with multiple supply voltages, there are reset ICs available, which monitor all of these voltages and issue reset/warning if any of them fall outside a specified range.

In such systems, sometimes the different supply voltages have to be ramped up and down in a particular order. The circuitry accomplishing this task often issues also a "power good" signal, which in fact is used to reset the involved microcontroller and/or surrounding peripherals.

1.7.6 Watchdog

Watchdog is the component most often integrated to reset ICs. The whole chapter 2 is devoted to watchdogs.

1.8 What about resets integrated into microcontrollers?

Newer microcontroller tend to have integrated reset circuits. However, it seems that it is not easy to implement a proper reset circuit especially in cheap digital IC process. On the other hand, there is a big pressure from marketing point of view to have the "magic" acronyms such as POR (power-on reset) and BOD (brownout detector) in the datasheets. Therefore, some of the lower-end microcontroller have a rather crappy implementation of internal reset, which cannot be relied upon in more critical applications. The main warning sign is usually inadequate documentation of these features in the datasheets (sometimes even nothing more than "has POR and BOD").

A good implementation of internal reset, besides having properly set trip voltage levels and perhaps also fusable options for duration, may also after reaching proper supply voltage and expiring the delay, hold the microcontroller in reset for a number of oscillator clock occurs (to enable oscillator stabilisation); and it should have an output to reset the surrounding circuitry.

Even if a microcontroller has an integrated reset, an external reset circuit might be required by the peculiarities of given application or by norms and legislation requirements.

1.9 Did we forget something?

Almost.

A microcontroller is never alone. It is built into a circuit, a board, a system, which might issue signals to the outer word, to other systems, or they might control powerful actuators. And all these might be subject to similar requirements - a defined safe state upon powerup, powerdown and any insufficient power condition.

In such case, the reset signal has to be distributed across the whole system.

2. Watchdog

> ***The watchdog is the embedded fire extinguisher.***
> ***You do not want to need it. But you want it there.***
> *Neil Kurzman, http://www.8052.com/forum/read.phtml?id=113613*

2.1 What is watchdog?

Watchdog is a circuit which should bring the microcontroller (and surrounding circuitry) into a known safe state, in case that the microcontroller does not work as intended ("runs away").

This is usually accomplished by a timer circuit, which during normal operation is periodically restarted from the microcontroller; and after not receiving the restart within a predetermined time, times out and issues a reset signal.

What problems are watchdogs expected to solve?

Watchdogs should act in case the microcontroller, or the system as a whole, "hangs". However, this expression is too vague and we need to be more precise in description of causes and symptoms. This analysis will then result in requirements that can be put on the watchdogs.

2.1.1 Software runaway

'Hung' system can be due to a **runaway program**, because of a **software bug** (a very common cause of program runaway is e.g. stack overflow), or a **hardware glitch** (which can be caused e.g. by crappy hardware design, unexpectedly high electromagnetic noise, impact of cosmic or other radiation...). In this case, the microcontroller in fact does not stop (except if it incidentally executes an instruction leading to halt, such as setting bit 1 of PCON register in 8051 which causes entering the powerdown mode) - but it **executes code in unexpected way**. It can stay in a loop infinitely; execute from data areas; execute code in an invalid way e.g starting in middle of a multibyte instruction; execute from uninitialized or even nonexistent memory areas. It can be in any if the above state, yet interrupts (e.g. from timer) may function normally. Or, the program itself may function normally but some of the SFR bits may be in unexpected state (e.g. interrupts may get spuriously disabled).

In these cases, the microcontroller will most probably recover to normal operation applying hardware **reset** signal. A great idea is to invoke a (nonmaskable) **interrupt** by the watchdog activation so that the state of microcontroller can be recorded in some way to facilitate further debugging; however, it must be arranged so, that it won't make things worse, and will work under any circumstances (i.e. does not assume anything on the microcontroller's internal state - e.g. stack integrity - implicitly). But even in this case, it is vital that **after a certain period a full reset is performed** by the watchdog, as the interrupt itself may not be able to recover from a faulty state - e.g. if an internal flipflop unaccessible from user software is in a wrong state.

As nothing can be assumed on what will be executed, it is important to arrange the **watchdog restart** so that it has a **low probability of occurring from a runaway software**. There are two ways how the restart can occur - either the restart code itself is executed by the runaway microcontroller, or executing some random code incidentally results in the sequence needed for watchdog restart. The first mechanism require that the **watchdog restart requires to pass through various parts of the software** during normal operation, the second require that the **restart requires to perform multiple steps** in a given order (e.g. toggle a pin within a given time window, or write two different values to a register in given order, perhaps also within a given time frame). One additional requirement is, that the **watchdog cannot be disabled from software**.

2.1.2 Hardware lockup

Another way how a system can "hang" is a **hardware lockup**. This can occur in many ways - it may be such that a standard reset clears the lockup; but also in a way which requires power cycle; and even as a result of permanently damaged circuit.

Standard watchdogs can cope only with the first type of lockup, but it is not unthinkable that in some applications the watchdog circuitry is capable of power cycling, being able to resolve also the second type of lockup, or could accomplish a more complex task leading to system recovery, for example to notify a control center sending an SMS.

In any case, it is often vital that not only the controlling microcontroller, but also the system's peripherals are secured in some defined state, in case of any type of fault (think e.g. of powerful machines, they should perform **emergency stop** for security reasons rather than run uncontrolled). So, a good watchdog should have an **output** signal to enable this sort of operation. It is also a good idea to **light up an indicator**, when a watchdog event occurs.

Another requirement resulting from this mode of failure is, that the watchdog should be much more resistant to causes of hardware lockups than the microcontroller it protects. This also means, that it **should not depend** in any way **on the proper operation of the microcontroller**. A very common type of watchdog violating this rule is a watchdog integrated into the microcontroller's chip, clocked by the microcontroller's clock, which is absolutely useless if the clock fails (which is quite common, being a sensitive analog circuit usually involving a crystal, which is a quite complex electromechanical device).

The latter requirement has implications also to the power supply arrangement - for example, the operation voltage range of a good watchdog has to cover the whole operation voltage range of the microcontroller and system it protects.

2.2 The constituents in detail

2.2.1 How does it time?

One of the most rudimentary incarnations of hardware watchdog is a traditional **monostable flip-flop** (a.k.a. single-shot), where the timing is provided by a capacitor charged via a suitable resistor. When the voltage on capacitor reaches a certain value, the output circuit (comparator) generates a reset. The restart pulse simply discharges the capacitor to start over the timing - charging. To change the timeout value, the capacitor or charging resistor should be changed.

Modern watchdogs in integrated circuits usually implement a different source of timing. They usually have a digital **counter**, clocked by an integrated ring- or RC-oscillator, which is reset by the restart pulse, and generates the reset output upon counter rollover.

The timeout value of external watchdogs can usually be selected between a couple of fixed values by tying one or more of its input pin high, low, or leaving them floating.

Internal watchdogs sometimes also have a register determining the rollover timeout, and may be clocked from the microcontroller's own clock rather than an independent oscillator (which is an incorrect practice as explained above).

A relatively common practice is, that the initial time delay is significantly longer than the subsequent timeout intervals. This is to facilitate initial setup of the microcontroller.

The exact timing value determines, how fast after a fault can the watchdog bring the circuit back into a controlled state; but also determines, how often the watchdog restart has to be accomplished in the software.

2.2.2 How is it restarted?

The traditional method of watchdog restart is simply by applying a **pulse**. This is not satisfying in cases where the runaway program could pulse the watchdog so the watchdog would never get activated.

As the developers want to keep the interface between the microcontroller and watchdog as simple as possible (i.e. a single pin), and they also don't want to loose resources in the program by a more sophisticated "protocol", the most commonly employed "higher security" scheme with external watchdogs is a **windowed** restart pulse - the pulse cannot be too short nor too long, otherwise it is ignored.

The equivalent of the latter method in the internal watchdog is, where the restart is bound to **successive write of two different values** into a register, possibly within a timing window, too.

Whichever the method of restart is, it is the way of how the restart is implemented in software, which ultimately determines the coverage of various possible modes of failure. A badly used watchdog restart method can degrade the best possible hardware scheme quite easily. For example, if the whole sequence of watchdog restart is unconditionally performed in a periodic timer interrupt, it will continue to restart even if the "main" program sticks erroneously.

A good practice is to bind a single watchdog restart to all the event the program has to perform within a periodic loop. A simple example is, when with the simple external watchdog the pin set is performed in the "main" program, while the clear is performed in the periodic timer interrupt.

2.2.3 What action does it take?

As already mentioned, a good watchdog, besides resetting the microcontroller and its peripherals, provides also a means for fault diagnostics. Although it is anticipated that the watchdog will get activated only in case of serious hardware fault, the more usual case is a software error, or just simply a sequence of events and stimuli, which leads to longer than expected program execution between consecutive watchdog restarts.

Internal watchdogs usually provide a flag in a register, reading which the microcontroller can distinguish, whether a reset has occurred due to watchdog timeout, or some other reason (e.g. undervoltage). However, once a watchdog reset occurs, the program context is already lost, so the only thing which can be done is to keep a counter of watchdog resets (versus resets of other reasons), and perhaps light up an indicator to notify the user that a watchdog event happened.

The best solution would be a watchdog triggering first a special interrupt - which could save the program context for further analysis - then performing the reset inadvertently after a short, fixed time.

For **development/debugging** reasons, it is sometimes good to have the watchdog disabled in a controlled environment. Although external watchdog can be cheated by restarting them from a periodic signal source (with '51s, the ALE signal is often (ab)used for this), it is perhaps better to have the output of watchdog disconnected from the actual reset input, and have it perhaps connected to a LED to indicate potential watchdog problem.

2.3 How does such watchdog look like?

Watchdogs can have various forms, as there may be various requirements put on them, depending on the particular application. There are "standard" watchdogs manufactured either as standalone IC ("external" to the microcontroller), or integrated into the microcontroller it is intended to protect ("internal"). However, there is **no universal solution** which would fit all requirements, and even the ready-made watchdogs can be used in a variety of ways.

2.3.1 Software watchdogs

Sometimes, function **similar to watchdog** is implemented purely in software. A usual incarnation of such is, when a flag is set often enough in the "main" part of the software, and is checked and cleared in a periodic (e.g. timer) interrupt; with a reset-like action taken by the interrupt when it encounters the flag unset. Although such scheme **does not cover** many of the abovementioned **requirements** and is fragile and might fail for a variety of reasons, it requires no extra hardware and can be added also as an afterthought with relatively little effort.

2.3.2 Internal watchdogs

Today, most microcontroller contains a **builtin watchdog**. However, they often are implemented in a way that reduces they usefulness. The commonly occurring problems are:

- watchdog can be disabled easily
- watchdog restarts by a single operation (SFR write)
- watchdog is clocked from the microcontrollers own clock
- there is no output to reset the peripherals upon watchdog event

On the other hand, internal watchdog, if properly implemented, can be very valuable while cheap. The restart mechanism can be bound on a complex sequence of events, such as multiple writes of different values in given order to a SFR perhaps even within a given time window - this is not easy to achieve using an external watchdog and decreases the chance of unwanted restart witch runaway program.

2.3.3 External watchdogs

These are often integrated together with reset circuits, and generate a reset when they time out. Usually, the watchdog is restarted by a pulse on one of its pins. More sophisticated models have to be pulsed within a time window, have adjustable timing and have a dedicated output.

2.3.4 Other, less usual forms of watchdog

There are various watchdogs realized in a rather sophisticated fashion, ensuring function of a relatively complex system by observing its external behaviour. For example, an unattended network device (server, router) may be periodically ping-ed (or its functionality otherwise tested) by a remote device acting as the watchdog; which upon lack of timely response may invoke a remote reset of the supervised device, often via an independent communication channel (e.g. GSM). Note that even this incarnation of watchdog has all three constituents: restart mechanism, timer and output, although of a rather complicated form.

2.3.5 What to do when even more paranoia is needed?

In mission critical systems, where significant assets and/or human lives are at stake, watchdog even of the most sophisticated form is not an adequate security measure. In these cases, redundant systems come into consideration. This involves partial or full redundancy of sensors and actuators, power supply redundancy, multiple redundancy of the control system with mutual checking of the computation results and polling-based decision on actions and/or shutdown of a faulty subsystem, optional fallback to simpler backup control system etc. This is a very wide topic, but the underlying principle is simple: one has to ensure a consistent answer to the question - what happens if this component fails? - for every component.

2.4 What else can be done for “code safety”?

There are quite a few measures which can help to “safe” a “runaway” microcontroller. Unfortunately, most of them have to be implemented in the hardware of the microcontroller (or its core), only a few depend on the choice of user.

Oscillator stop detection resembles a true watchdog in its function - basically it is an independently clocked timer, and if no oscillator clock occurs within a timeout, it issues a reset pulse.

Timed access to critical SFRs prevents spurious writes to SFRs, which would have “suicidal” consequences - for example would result in FLASH corruption or erasure. To effectively write into such SFR, two writes into two distinct SFRs have to be performed (the first of them being the “enabling” write), both within a short time window (successively after each other).

In some higher-end microcontrollers, the circuitry used for on-chip debugging can be used also to set up traps for **stack, program counter and/or data pointer runaway**, resulting in a special (often unmaskable) interrupt being triggered. This is a simpler yet effective version of complex memory management units in the “adult” processors, where each process runs within constraints of virtual memory assigned to it by the operating system.

One of the simplest measures which can be accomplished almost in each case, is to **fill up unused code memory** by a pattern, executing which (by a runaway program) would lead to a “safe harbour”. Care must be taken to chose the pattern so, that it leads to the desired result regardless of at which part of a possibly multibyte instruction does the erroneous execution start. Some microcontrollers have their opcode arranged so, that executing from a blank memory (usually containing 0xFFs) results in a specific interrupt or in reset. In some microcontrollers, similar effect is also caused by accessing non-existent memory (especially if external memory access is prohibited by security fuses, or simply by design).

A similarly simple provision is to fill up unused interrupt vectors by call or jump to a fault-resolving routine.

2.5 Conclusion

Watchdog may be a valuable tool in building a secure system, but its usage is a matter of compromises. There is no universal solution nor guidelines. It is expensive - not the hardware, but writing the appropriate software for efficient restart (and perform the associated analysis). It even can do more harm than good in certain situations - it may cover up errors and give false feeling of safety.

So, usage of watchdog is matter of tradeoffs, with one of the choices to consider is to not use a watchdog at all.

3. ICs

As reset and watchdog circuits are relatively simple and low integration analog circuits, almost every analog integrated circuit manufacturer makes a range of reset/watchdog ICs. They range from simple 3-terminal ("three-legger") reset-only circuits with fixed timeout, to complex circuits, sensing multiple voltage and providing multiple, mutually sequenced outputs. The watchdog function, if implemented, can be also simple, fixed timeout, or more complicated, "programmable", providing a longer timeout after reset and then shorter timeouts during runtime, etc. There are also ICs combinating reset/watchdog with other similar functions, e.g. battery handling/switchover; or with completely unrelated functions, e.g. real-time clock and EEPROM memory.

The reset timeout is often "programmable" by either capacitor, resistor, or by connecting one or more pins to certain voltage level, or leavinging it floating. On the other hand, the trip voltage is always fixed, although many circuits have standalone sense input (i.e. don't derive the trip level from the supply voltage), so there the actual tripping level can be adjusted by using an appropriate voltage divider.

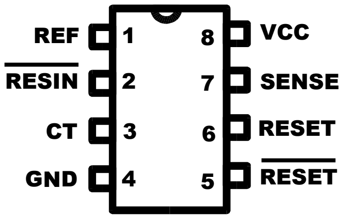
Most modern reset ICs are manufactured in CMOS technology and are correctly working with VCC as low as around 1 V. Some of the datasheets guarantee that outputs are floating when VCC falls too low, so that an appropriate pull-up or pull-down resistor may ensure correct logic level of reset and other output signals even when VCC is below these voltages.

In spite of this diversity, there is a set of popular reset/watchdog ICs produced by multiple manufacturers. They usually share the same number and differ in manufacturer-specific prefix, so in the following we will denote them only by the number and the reader has to look up the specific part number at the list of reset ICs of the given manufacturer. They are often listed under "power management", "system supervisors" and similar fancy group name. Some of the most popular of them - by no means a complete list - are described in a few sentences below.

..7705

This is a rather old reset IC, maybe one of the first reset ICs at all. In fact, it is a family if reset ICs with various trip voltages - the last two digits in the name indicates this - the 5V one, 77**05**, being the most commonly occuring one.

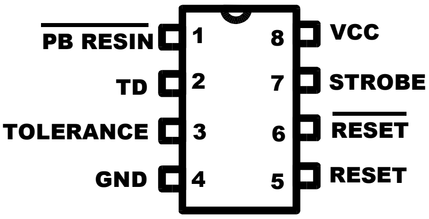
Originally, 7705 is built in bipolar technology. The reset pulse timing is determined by an attached capacitor. As a curiosity, the voltage sensing circuit fires a thyristor which discharges the timing capacitor; maybe as a consequence of this the circuit is reported to be rather sensitive to "spikes" in power supply. It has a dedicated "sense" input and a pushbutton (more precisely, a TTL-level) input. It has both active-low and active-high outputs, and this is one of few, if not the only, IC, where the active-high output is open-collector (actually a PNP transistor). Unfortunately, correct operation is guaranteed only from around V_{CC} > 3.5 V. It typically occurs in DIP-8 and SO-8 packages.



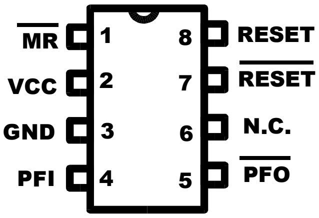
There is also a CMOS variant of the 7705, often carrying the letter C either in the middle of numbers or in the prefix. It has a lower power consumption, works from lower supply voltage (around), but note that its active-high output is push-pull.

..1232

A combination of reset and watchdog, dedicated for 5 V supply. Does not have a sense input, but two trip points can be selected through TOLERANCE input: 4.5 V and 4.75 V. Three watchdog timeout delays beweer 150 ms and 1200 ms can be selected through setting the TD pin high, low or leaving it floating. Has a debounced pushbutton input, and both active-low (open-drain) and active-high (push-pull) outputs. Comes in DIP-8 or SO-8 packages.

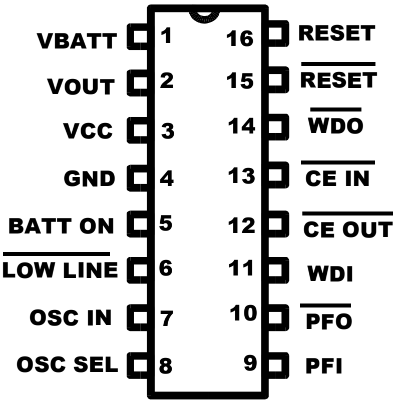


..707



A combination of reset and power-fail circuit. Does not have a sense input and has a fixed nominal trip voltage of 4.63 V, but there is a variant under ..708 marking, which is manufactured in several variants with different fixed trip voltages. Has a pushbutton input pin /MR with an internal pullup, and both active-low and active-high outputs, both in push-pull configuration. The independent power-fail circuitry compares input PFI with a 1.3 V reference, and Comes usually in DIP-8 or SO-8 packages.

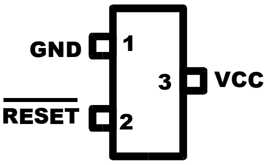
...69x



This is a has-it-all family of reset/watchdog ICs. The V_{CC}-low detecting circuit not only triggers the reset (output in both polarities), but also has a dedicated /LOW_LINE output. Watchdog, refreshed from any transition at /WDI input, has a longer timeout after reset, and not only contributes as a reset source, but also has a separate /WDO output. Not only can the reset/watchdog timeout be adjusted in two steps by setting the OSC_SEL pin, but the internal timer can be overdriven by external clock sourced to the OSC_IN pin. There is a provision to source a SRAM from V_{OUT} pin, which switches over from V_{CC} to battery connected to V_{BATT} pin, when V_{CC} falls below threshold; this is also indicated on a dedicated BATT_ON output. There is another provision for SRAM content protection: a chip select, routed from pin /CE_{IN} to pin /CE_{OUT} through the chip, is gated during low V_{CC}. There is also an independent power-fail detector, comparing the PFI input with a 1.3 V nominal reference voltage, and outputting the "result" on /PFO pin.

Even an IC packed with features as the ..69x do not have every possible option. They lacks pushbutton input and the trip voltage is fixed, to nominally 4.65 V and 4.4 V (depending on particular model). The complexity requires a big number of pins, the "odd number" circuits (..691/..693/..695) come in DIP-16, SO-16 and TSSOP-16 packages. The "even number" circuits (..690/..692/..694) are simplified versions lacking the chipselect-gate, adjustable timing, and separate /WDO and BATT_ON and /LOW-LINE outputs, thus they fit into 8-pin package, but they still provide battery switchover and the separate power-fail circuitry.

...809/810



Contrary to the previous, this is the simplest form of reset IC, the "three-legger". Comes in many variants with different fixed reset trip points, suiting 5V, 3.3V, 3V and other power voltage systems. The reset duration is fixed to around 200ms, but there are similar circuits available at various manufacturers with different delay time, or no delay time at all (i.e. "pure" supply voltage monitors). The ..809 has active-low output, while the ..810 has active-high output, both push-pull.

These and similar ICs come almost invariably in the tiniest SMD packages, typically in SOT-23. One of the few through-hole-packaged "three-leg" reset circuit is the MCP100/101 combo manufactured by Microchip.