

18. Čítač/časovač a přerušení

| | |
|----------------|-------------------------|
| Created | @April 26, 2025 7:11 PM |
| Tags | Done |
| Kdo vypracoval | Jirka |

Čítač

Čítač je zařízení, které počítá nebo odpočítává (a někdy také zobrazuje) kolikrát proběhla určitá událost nebo proces. Princip čítače je nárůst nebo pokles hodnoty o jednotku vzhledem k předchozí hodnotě při příchodu čítacího impulsu:

$$N_{(t)} = N_{(t-1)} + 1$$

nebo

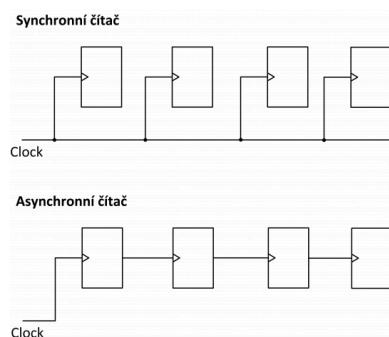
$$N_{(t)} = N_{(t-1)} - 1$$

kde N = hodnota čítače, t = aktuální stav po příchodu čítacího impulsu, $t-1$ = časový okamžik před příchodem čítacího impulsu

Čítač určený pro rychlé čítání elektrických impulsů, řádově s frekvencí kHz, MHz až GHz. Podle konstrukce je možno čítače rozdělit na **binární** (dvojkové), **oktanové** (osmičkové), **desítkové** (dekadické) a **hexadecimální** (šestnáctkové). Z hlediska matematického zpracování impulsů se čítače dělí na **vzestupné** - zvyšuje svou hodnotu (inkrementační operace) a **sestupné** - snižuje svou hodnotu (dekrementační operace).

Synchronní čítače: Je jich potřeba více, ale projde rychleji.

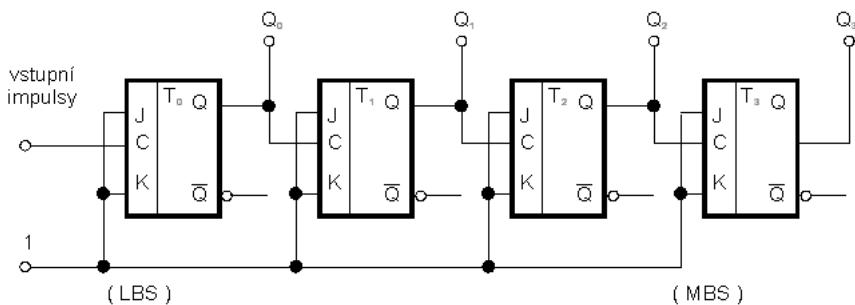
Asynchronní čítače: Dochází k zpoždění, ale je jich potřeba méně.



Elektronický čítač se většinou realizuje pomocí elektronických **logických obvodů** (především pomocí **klopných obvodů**). V takovém případě mírá vstup, na který se přiveze časový vstup označovaný **C / CLK**, dále vstup nulování hodnoty – **RESET** okamžitě vynuluje co je na čítači, vrátí do původní hodnoty, nastavení čítače na přednastavenou hodnotu – **SET**. Na výstupech je binárně nebo v BCD (desítková v 0 a 1, jak nešetří místo, čtveřice bitů pro jednu cifru 32 = 00110010).

Užití čítačů v obvodech

Asynchronní čítač vpřed

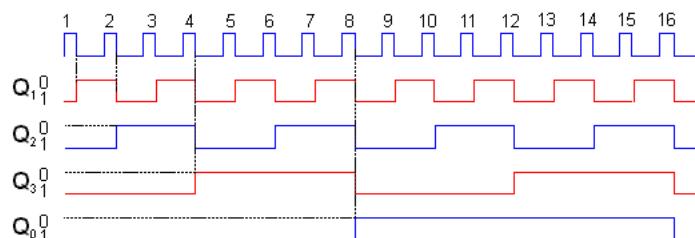


Řetězec klopných obvodů T. Klopné obvody byly vytvořeny pomocí obvodu J-K připojením obou vstupů na logickou 1. (O obvodech J-K v kapitole 7.) Jednotlivé klopné obvody mění stav výstupu při každé úběžné hraně na svém hodinovém vstupu. Překlápení obvodů se tedy řídí v podstatě dvěma pravidly:

1. Výstup Q0 obvodu T1 mění svůj stav při každé úběžné hraně vstupních impulů,
2. Všechny ostatní výstupy mění svůj stav právě když předcházející klopný obvod mění stav výstupu Q z 1 do 0.

Vidíme, že stav výstupů Q0 - Q3 je přesně binární reprezentace čísla, udávajícího pořadí vstupního hodinového impulsu.

Mluví o tom také pán z TEF na odkazu na konci.



Švihla o tom mluvil na hodině

Časovač

Časovač obecně slouží k asynchronnímu řízení vykonávání strojového kódu. Hardwarové časovače využívají operační systém pro řízení multitaskingu a plánování dalších důležitých událostí, které jsou závislé na reálném čase. (Časovač může být umístěn ve specializovaném integrovaném obvodu, v čipsetu nebo přímo v procesoru.)

Obvykle jsou to čítače, které program nastaví na nějakou hodnotu, která se následně automaticky snižuje. V okamžiku, kdy dosáhne nuly, je vyvoláno přerušení, které je doručeno procesoru. Časovač umožňuje procesoru zpracovávat během měření času jiné úlohy, čímž snižuje režii systému a zvyšuje jeho výkon.

Časovač může pracovat v **jednorázovém režimu**, kdy je přerušení vykonáno po vypršení časovače nebo v **periodickém režimu**, kdy je přerušení vyvoláváno periodicky po uplynutí nastaveného fixního času (například každou 1 milisekundu).

Časovač se v počítači používá pro řízení multitaskingu (vykonávání více věcí najednou). Operační systém rychle střídá procesy a tak působí, že běží najednou.

Druhy

Hodiny reálného času (RTC)

Hodiny reálného času jsou integrovaný obvod, který udržuje informaci o aktuálním čase v počítačích a dalších elektronických zařízeních. Nepoužívá se pro periodické vyvolávání přerušení. V současné době je RTC na základních deskách integrován do čipsetu.

Programmable Interval Timer (PIT)

je nejstarší používaný časovač v počítačích IBM PC kompatibilních (starý počítač). Obsahuje tři čítače: časovač číslo 0 používají operační systémy jako systémový časovač, časovač číslo 1 je z historických důvodů použit pro obnovování paměti RAM a časovač číslo 2 pro PC speaker.

Spíše pro zajímavost

High Precision Event Timer (HPET)

Umožňuje vyšší a přesnější rozlišení času (například pro synchronizaci multimédií). Na rozdíl od RTC a PIT je jeho programování efektivní. Jeden HPET blok obsahuje 3 až 32 časovačů a bloků může být až osm. Časovač přičítá a po dosažení hodnoty nastavené v registru vyvolá přerušení.

Protože je počet hardwarových časovačů v počítači omezen a není možné předpokládat, že budou stačit pro všechny účely, je možné používat jen jeden časovač, který obsluhuje operační systém a vytváří k němu softwarovou časovou frontu. Hardwarový časovač se pak používá k odměření času zbývajícího do první události v časové frontě.

Popis fronty

Ve frontě jsou události řazeny podle času, takže nejbližší událost je první. Po uplynutí času nastaveného v časovači do první události je časovačem vyvoláno přerušení, které vyvolá obsluhu přerušení uvnitř operačního systému, která obsluhuje čekací frontu. V obsluze přerušení je z fronty odebrána událost, která přerušení vyvolala, časovač je nastaven na čas zbývající do následující události ve frontě a aktuální (vyřazená) událost je předána k vyřízení.

Přerušení

Přerušení je metoda pro asynchronní obsluhu událostí, kdy procesor přeruší vykonávání sledu instrukcí, vykoná obsluhu přerušení, a pak pokračuje v předchozí činnosti. (Něco musí udělat přednostně.)

Obsluha přerušení

Přijde-li do procesoru signalizace přerušení, je v případě, že obsluha přerušení je povolena, nejprve dokončena právě rozpracovaná strojová instrukce. Pak je na zásobník uložena adresa následující strojové instrukce, která by měla být zpracována, kdyby k přerušení nedošlo. Pak je podle tabulky přerušení vyvolána obsluha přerušení, která obslouží událost, kterou přerušení vyvolalo. Obsluha přerušení je zodpovědná za to, aby na jeho konci byl uveden stav procesoru do stavu jako na jejím začátku, aby výpočet přerušené úlohy nebyl ovlivněn, což se z důvodu vyšší rychlosti obvykle dělá softwarově (některé procesory umožňují uložit svůj stav pomocí speciální strojové instrukce). Na konci obsluhy přerušení je umístěna instrukce návratu, která vyzvedne ze zásobníku návratovou adresu a tak způsobí, že z této adresy bude vyzvednuta následující strojová instrukce. Přerušená úloha tak až na zpoždění nepozná, že proběhla obsluha přerušení.

Tabulka přerušení umožňuje, aby procesor mohl rozlišit více různých přerušení (rozlišených čísla), ke každému vyvolat odpovídající obsluhu přerušení (podprogram) a aby šlo jednotlivé obsluhy umístit na libovolná místa v paměti.

Typy přerušení

Vnější přerušení je označováno podle toho, že přichází ze vstupně-výstupních zařízení (tj. z pohledu procesoru přicházejí z *vnějšku*). Vnější přerušení jsou do procesoru doručována prostřednictvím řadiče přerušení (obvod, který umožňuje stanovit prioritu jednotlivým přerušením).

Vnitřní přerušení vyvolává sám procesor, pokud dojde k problému při zpracování strojových instrukcí, címž umožňuje operačnímu systému na tyto situace reagovat. Jedná se například o pokus o dělení nulou, pokus o provedení neexistující instrukce, porušení ochrany paměti, nepřítomnost matematického koprocesoru, výpadek stránky a podobně.

Softwarové přerušení je speciální strojová instrukce. Tento typ přerušení je na rozdíl od druhých dvou typů synchronní, je tedy vyvoláno zcela záměrně umístěním příslušné strojové instrukce přímo do prováděného programu. (Přijde signál přerušení.)

(Původní text, který tu byl před jirkou)

IRQ = Interrupt ReQuest

řeší problém více inputů

když přijde input do procesoru když něco vykonává tak signál nebude přečten

Maskované přerušení znamená, že ho bude bude procesor ignorovat i když je aktivní.

NMI = Non Maskable Interupt

Nelze ho ignorovat

Nejvyšší priorita - zastaví co právě dělá procesor

Při výpadku napájení nebo při chybách, které nelze hned tak spravit (např.: chipsetu, chyba v paměti nebo poškození dat)

Doporučuji tento odkaz na maskování, je v angličtině, ale je to dobře popsáno

How do you use interrupt masking and unmasking to control interrupt requests in microprocessor?

Learn how to use interrupt masking and unmasking to control interrupt requests in microprocessor, using hardware or software methods. Improve efficiency and responsiveness of your microprocessor.

 <https://www.linkedin.com/advice/0/how-do-you-use-interrupt-masking-unmasking-control>



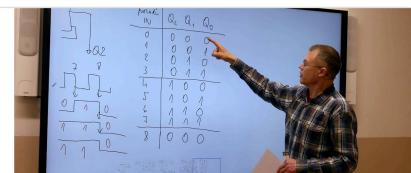
Zdroje

Odkaz na pána z TEF. Je to možná k ničemu, ale v nejhorším máš o čem mluvit u maturity

Ek4r - Suchý - Klopné obvody a čítače3 - čítač

Online výuka SPŠE Fr. Křížka Praha - www.skolakrizik.cz

 <https://www.youtube.com/watch?v=UcZEEfGSAQA>



Odkaz na prezentaci z nějaké školy. Je to učňák, ale dobré.

www.dubno.cz

http://www.dubno.cz/images/stories/dum/8.sablona/24/pdf/VY_32_INOVACE_CTE_2.MA_18_%C4%8C%C3%ADta%C4%8D%C4%8D%C4%80asynchron%C3%AD,%20synchrone%C3%AD.pdf

Můžete se kouknout i na knížku, kterou doporučuje Švihla

20 Čítače | Hradla, volty, jednočipy

Musím se vám k něčemu přiznat. Když jsem jako desetiletý chlapec dostal do ruky katalog integrovaných obvodů Tesla a viděl jsem obvody, které se jmenují „čítače“, tak jsem si říkal: „Asi udělali chybu, asi tam mělo být napsáno počítače,“ a fakt jsem se těšil, že tohle už je to ono. No, není. Trochu mě to mrzelo, ale jen chvíliku, brzo jsem totiž zjistil, že čítače jsou opravdu šikovné součástky.

 https://maly.gitbook.io/hradla-volty-jednocipy/20_citate

www.zilog.com

<https://www.zilog.com/docs/z80/um0080.pdf>

Elektronická učebnice - ELUC

Mikrokontrolér obvykle obsahuje jeden nebo více časovačů/čítačů, které lze programově konfigurovat. Časovače a čítače umožňují programu mikrokontroléru např. měřit časové intervaly, detektovat počet impulsů za určitý časový interval apod.

👉 <https://eluc.ikap.cz/verejne/lekce/869>

Detekce náběžné hrany:

RC členy

Nejjednodušší RC členy, tzv. poločlánky, jsou složeny z rezistoru a kondenzátoru – proto RC. Vždy je jedna součástka v sérii (v cestě signálu) a druhá paralelně k výstupu

👉 <https://www.souep.cz/wp-content/ucitle/kulhanek/OPVK%202009/RC/RC%20cleny.htm>

Difference between Maskable and Non Maskable Interrupt - GeeksforGeeks

Maskable interrupts can be selectively enabled or disabled for lower priority tasks, while non-maskable interrupts are critical and cannot be ignored, ensuring prompt handling of high-priority events.

👉 <https://www.geeksforgeeks.org/difference-between-maskable-and-non-maskable-interrupt/>



Text o užívání čítačů - originál, v textu upraven

https://physics.mff.cuni.cz/kfpp/skripta/elektronika/kap6/6_7.html

Old jirkovo bez předělání

Předělané podle Robba