OBSAH

SEZNAM OBRÁZKŮ

1. ÚVOD

Operační systémy pracující v reálném čase mají uplatnění tam, kde je vyžadována okamžitá odezva na vzniklé podněty. Využívá se jich v širokém spektru odvětví, např. v dopravě, elektrárnách, průmyslu apod. V současnosti jsou implementovány na různých platformách a úrovních hardwaru od mikrokontrolérů přes průmyslové automaty až po osobní a specializované počítače pro různé účely.

Tato práce se zabývá popisem vývojových prostředí LabVIEW a LabWindows/CVI s integrovaným modulem reálného času.

V první časti je uvedena definice systému reálného času a jeho použití, dále je proveden základní rozbor vybraných vlastností operačních systémů a popis operačního systému reálného času se zaměřením na operační systém VxWorks.

Ve druhé části je uveden stručný popis prostředí LabVIEW a LabWindows/CVI s modulem reálného času a popis vytváření aplikací reálného času v těchto prostředích.

Třetí část práce se zabývá návrhem metodiky porovnání vlastností systémů sběru analogových dat založených na OS WINDOWS a systému reálného času.

2. SYSTÉMY REÁLNÉHO ČASU

Nejzákladnějším požadavkem kladeným na systém reálného času je jeho včasná odpověď za všech možných okolností.

Obecně lze systém reálného času definovat jako informační systém, který zpracovává asynchronní vstupy a produkuje odpovědi v pevně stanoveném čase. Doba, kterou má systém k dispozici pro provedení úlohy, je známa předem. Na počtu vstupů a jimi vyvolané pracovní zátěži systému přitom nezáleží.

Volnější definice, která je blízká skutečným systémům, zní: Systém reálného času je takový informační systém, který zpracovává asynchronní vstupy a produkuje odpovědi v rozhodnutém a ohraničeném čase. Tato definice odráží skutečnost, že potřebný čas může být vypočítán ze zátěže systému. Navíc musí být tento čas ohraničený, což odpovídá pojmu nejdelší možná doba odezvy [8].

2.1 ROZDĚLENÍ SYSTÉMŮ REÁLNÉHO ČASU

V praxi není vždy nutné použít plně deterministický systém. Proto se systémy reálného času dělí na

2.1.1 Hard Real-Time Systems[1]

Požadavky hard RT nutí systém vždy splnit známý časový termín. Systému hard RT (např. systému řízení létajícího prostředku, jaderného reaktoru i mnoha jiných a běžnějších systémů) jsou tedy zadány časové krajní meze, které musí splnit, aby nenastala katastrofická selhání, jako je ztráta životů nebo zařízení.

2.1.2 Soft Real-Time Systems[2]

V systému soft RT mohou být konkrétní časové okamžiky zmeškány. Příkladem systému soft RT může být systém optimalizující spotřebu paliva v motorovém vozidle.

---

[1] Hard Real Time System – systém reálného času s tvrdými požadavky na dodržení doby odezvy

[2] Soft Real Time System – systém reálného času s měkkými požadavky na dodržení doby odezvy

2.1.3 Non Real-Time Systems

Na tyto systémy nejsou kladeny požadavky na čas pro splnění úlohy, hlavní smysl spočívá v provedení úlohy v průměrném čase (např. osobní počítače)

2.2 POUŽITÍ REAL-TIME SYSTÉMŮ

Oblast použití systémů pro řízení v reálném čase neustále narůstá. Kromě tradiční oblasti řízení technologických procesů, kde se jedná často o kritické úlohy reálného času např. antiblokovací brzdový systém ABS u vozidel, a informačních systémů pro práci v reálném čase např. rezervace místenek v dopravě, kde se jedná o nekritický systém reálného času. Rychle narůstá použití systému reálného času zejména v oblasti vestavěných systémů (Embedded Systems) [9].

3. ZÁKLADNÍ PARAMETRY A VLASTNOSTI OPERAČNÍCH SYSTÉMŮ

3.1 MULTITASKING

Obecné označení pro schopnost procesoru provádět několik úloh souběžně.

3.1.1 Kooperativní (nepreemptivní) multitasking

Výpočetní čas procesoru je přidělen jednomu programu na dobu jeho běhu. Po ukončení programu je procesor vrácen operačnímu systému. Ten ho pak přidělí jinému programu. Tato metoda má nevýhodu, že program nemusí procesor vrátit v dostatečně krátkém časovém úseku, což způsobí dojem, že ostatní programy nepracují. Ještě horší případ nastane ve chvíli, kdy program procesor nevrátí vůbec (např. zhavaruje). Tato situace vede ve většině případů k havárii celého systému.

3.1.2 Preemptivní multitasking

Výpočetní čas procesoru je přidělen programu pouze na určitou dobu a po jeho uplynutí je úloha pozastavena a operační systém přidělí výpočetní čas jinému programu. Z toho vyplývá, že nemohou nastat stavy uvedené u kooperativního multitaskingu. Nevýhodou tohoto řešení je vyšší náročnost na hardware počítače [6].

3.2 MULTITHREADING

Obecné označení pro schopnost programu sám sebe větvit - program se větví na tzv. vlákna (threads), která mohou běžet současně. Vzniká tak možnost paralelního provádění několika větví výpočtu v rámci jednoho procesu nad jedním adresovým prostorem (vícevláknové aplikace). Všechna vlákna (jedné aplikace) sdílejí systémové zdroje, např. paměť (na rozdíl od multitaskingu, kde jsou jednotlivé procesy zcela oddělené). Čas vláknům přiděluje OS (podle jejich priorit).

3.3 PLÁNOVÁNÍ ÚLOH

RTOS je založen na prioritním zpracování a přepínání úloh. Každá úloha má svou prioritu. S rostoucí prioritou roste také nutnost odbavení úlohy procesorem.

Rychlého odbavení lze dosáhnout přepínáním úloh. To znamená, že operační systém je schopen zastavit právě probíhající úlohu, která má nižší prioritu, a spustit úlohu s vyšší prioritou. Po skončení této úlohy operační systém pokračuje s prováděním předchozí úlohy.

Rozlišujeme tyto základní stavy úloh:

- Terminated[3] – v tomto stavu se nalézá úloha před a po provedení.
- Ready[4] – čeká již jen na přidělení procesoru
- Running[5] – procesor provádí programový kód úlohy
- Suspended[6] – je pozastaveno provádění úlohy z důvodu nesplnění nějaké podmínky nebo proto, že jí není přidělen zdroj

Nejpoužívanější metody plánování:

3.3.1 Round-Robin plánování

Při zpracování aktivního vlákna s nižší prioritou než má vlákno čekající ve frontě, je vlákno uloženo a vráceno zpět do fronty, poté je připraveno pro běh vlákno s vyšší prioritou. Round-robin pracuje buď v preemptivním nebo kooperativním multitaskingu. Pokud nastane situace, že jsou připravena vlákna se stejnou prioritou, každé vlákno bude spuštěno na definovanou dobu. Nelze tudíž přesně definovat čas zpoždění, protože vlákna mohou být různě náročná a mohou se zpracovávat delší dobu.

3.3.2 Prioritní plánování

Rozdíl od metody Round-robin je ve zpracování vláken se stejnou prioritou. Vlákno se provádí tak dlouho, dokud není ukončeno, nebo dokud není připraveno vlákno s vyšší prioritou. Nikdy není přepnuto vlákno vláknem se stejnou prioritou.

---

[3] Terminated = ukončen

[4] Ready = připraven

[5] Running = běžící

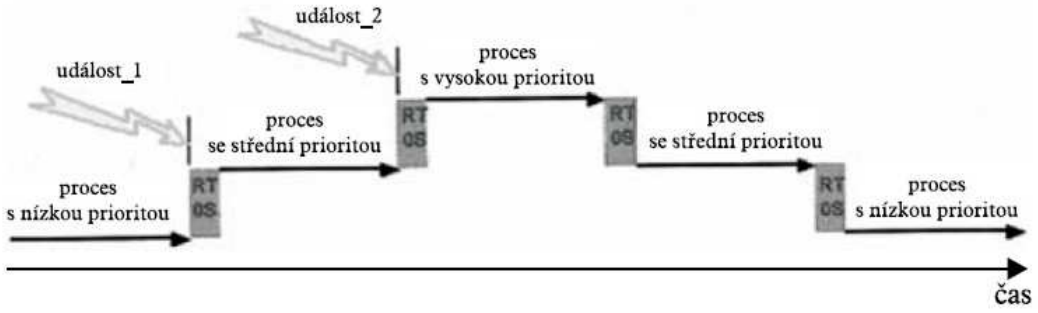
[6] Suspended = odložen

3.4 PŘEPÍNÁNÍ ÚLOH

Většina RTOS provádí plánování procesů na základě prioritního preemptivního plánování. Každému procesu softwarové aplikace musí být přiřazena priorita, nejvyšší hodnota priority představuje požadavek na nejrychlejší odezvu. Rychlá odezva je přirozeně zajištěna právě preemptivním plánováním procesů. Preempce znamená, že plánovač může pozastavit kterýkoliv proces v kterémkoliv bodě zpracování, jestliže přijde požadavek od jiného procesu na své okamžité zpracování.

Základní pravidlo, které řídí prioritní preemptivní plánování, je v kterémkoliv okamžiku následující, „proces s nejvyšší prioritou, který je připraven na zpracování je procesem, jehož zpracování musí být zahájeno“. Jinými slovy, jestliže jsou dva procesy připravené na zpracování, jeden s nízkou prioritou a druhý s vysokou prioritou, pak plánovač naplánuje na zpracování proces s vysokou prioritou jako první. Proces s nižší prioritou se dostane ke zpracování až tehdy, když proces s vyšší prioritou ukončí svou aktuální činnost [9].

Jestliže začne zpracování procesu s nižší prioritou a pak je připraven na zpracování proces s vyšší prioritou (např. při příchodu vnější události například sepnutí vypínače) bude prioritní preemptivní plánovač postupovat následovně: Umožní procesu s nižší prioritou ukončit právě probíhající assemblerovskou instrukci. (Neumožní dokončení celého řádku v kódu vyššího jazyka; ale umožní dokončení zpracování do následujícího tiku hodin.) Zpracování procesu s nízkou prioritou pak bude okamžitě zastaveno a umožní zahájit zpracování procesu s vyšší prioritou. Po ukončení zpracování procesu s vyšší prioritou proces s nižší prioritou bude pokračovat ve zpracování.



Obrázek 1 – Příklad prioritního preemptivního plánování [9]

3.5 LATENCE PŘERUŠENÍ

Latencí přerušení se rozumí časová prodleva, která uběhne mezi časem kdy vyvolané externí přerušení se dostane k procesoru a momentem, kdy je spuštěna obslužná funkce přerušení. Při popisu RTOS, ale zejména při návrhu RT aplikace je nutné přesně znát hodnoty této časové prodlevy.

3.6 GARBAGE COLLECTION

Garbage collection je pojem, který je znám od počátku šedesátých let minulého století. V tehdejší době byl Garbage collection („GC“) používán pro optimalizaci diskového prostoru na sálovém počítači. Později, kdy se začalo pracovat ve větší míře s databázemi, se „GC“ uplatňoval jako nástroj pro optimalizaci nebo defragmentaci databáze. S dalším vývojem programovacích jazyků se „GC“ se používal ve smyslu uvolňování dynamické paměti. Tato metoda poskytuje automatickou správu paměti, provádí uvolňování nepotřebné paměti s nízkou prioritou.

Pokud má nějaký programovací jazyk zabudovanou podporu „GC“, používají se především tyto dvě následující metody.

První, jednoduší, lze nalézt například v jazyku Perl a nazývá se „reference count GC“. U každého objektu je uvedeno, kolik ostatních objektů na něj navazuje. Pokud tento počet klesne na nulu, je objekt odstraněn. Tento způsob „GC“ je velmi rychlý. Má však jeden nedostatek a tím jsou „cyklické reference“. Ukazuje-li objekt

A na objekt B a naopak, a na tyto objekty (A a B) již žádné další objekty neukazují, měly by se v „GC“ smazat, což se nestane, protože mají „reference count“ 1.

Druhá modernější metoda, která je například v Javě, se nazývá „mark & sweep“. Je výrazně pomalejší než výše uvedená metoda, ale je to zatím nejdokonalejší známá metoda na čištění objektů. Spočívá v jednom kořenovém objektu, který představuje hlavní program. „GC“ začíná právě u tohoto objektu, označí všechny objekty od něj dosažitelné. Zbylé objekty, které nejsou dosažitelné, se smažou. Tato metoda je velmi náročná na výkon procesoru. Je proto kombinována s již zmíněnou starší metodou.

3.7 DYNAMICKÉ PŘIDĚLOVÁNÍ PAMĚTI

Determinismus se uplatňuje i v oblasti dynamického přidělování RAM paměti. Non-RTOS často využívá metody heap[7]. V jazyce C se pracuje s touto pamětí pomocí funkcí „maloc“ a „free“. Paměť lze alokovat až v okamžiku, kdy je potřeba a až je známo, kolik jí bude potřeba. V okamžiku, kdy již není paměť potřeba, lze ji vrátit systému. Pomocí volání „maloc“ si úlohy mohou dočasně půjčit danou velikost paměti od operačního systému. Jakmile úloha skončí, vrátí se paměť zpět operačnímu systému pomocí příkazu „free“. Operační systém vrátí tuto část paměti do celkového objemu paměti. Vrácená paměť může být použita jako část většího celku nebo ji lze rozdělit na menší bloky pro úlohy s menšími nároky. Neustálým rezervováním a uvolňováním paměti postupně dojde ke fragmentaci paměti. Fragmenty paměti jsou pak nepoužitelné pro požadavky dalších procesů - malé bloky paměti nelze spojit a vytvořit tak spojitou oblast. Operační systém může odmítnout novou aplikaci z důvodu nedostatku souvislé paměti. Tento problém lze vyřešit speciální metodou „garbage collection“. Bohužel způsob provozu toho softwaru je nedeterministický, náhodně přidává zpoždění při běhu, tudíž je nevhodný pro RTOS.

RTOS podporuje přidělení paměti bez fragmentace omezením velikosti paměťové oblasti, která bude využívána pro aplikace. Tato metoda je méně pružná

[7] heap = halda

než přidělování volných oblastí z celého paměťového prostoru, ale zamezí fragmentaci.

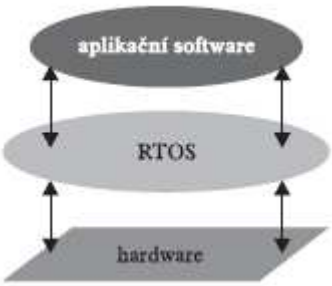
Metoda „memory pool“ přiděluje aplikacím přesně dané velikosti paměti, např. 4 nebo 8 velikostí z celkové sdílené paměti. Tato metoda zcela odstraní fragmentaci paměti při přidělení. Po vrácení bloku zpět do sdílené paměti se již nepřiřadí do definovaných velikostí a opět postupně dochází ke fragmentaci. Pro eliminaci této nevýhody se používá postup, který vrácený paměťový blok vypíše na seznam volných oblastí. Tímto postupem zamezíme fragmentaci nebo k rozdělování částí paměti do bloků [6].

4. RTOS – OPERAČNÍ SYSTÉMY REÁLNÉHO ČASU

4.1 ZÁKLADNÍ KONCEPCE RTOS [9]

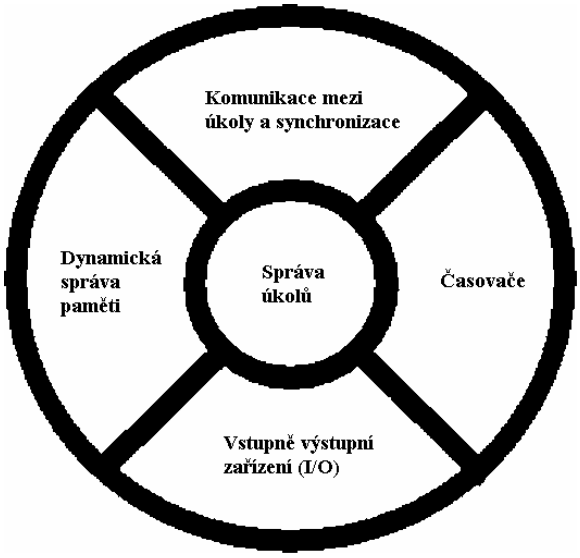
4.1.1 Jádro systému

Jádro systému považovat za část operačního systému, která poskytuje základní služby aplikacím běžícím na procesoru. Jádro RT operačního systému (RTOS) vytváří „abstraktní“ úroveň, která je skryta aplikačnímu software, jež zahrnuje např. detaily hardware procesoru (nebo sady procesorů), v době běhu aplikačních úloh. Takto vytváří interface mezi aplikačními úlohami a vlastním hardwarem počítače.



Obrázek 2 – Jádro RTOS jako abstraktní úroveň mezi aplikačním softwarem a hardwarem [9]

Jádro zahrnuje většinu nástrojů pro podporu reálného času, mezi které především patří rychlý multitasking, podpora přerušení, preemptivní a cyklické plánování úloh. Jádro je navrženo s minimální režii systému, která umožňuje rychlou a deterministickou odezvu na externí události. Výkonný mechanismus meziprocesorové komunikace umožňuje nezávislým úlohám koordinaci svých aktivit v systémech reálného času. Jádro RTOS poskytuje pět hlavních skupin základních služeb pro aplikační software(Obrázek 3):



Obrázek 3 – Základní služby jádra RT operačního systému [6]

4.1.2 Řízení procesů

Mezi základní služby jádra patří služby pro řízení procesů. Tato skupina služeb umožňuje vývojářům aplikačního software navrhnout aplikační software ve tvaru několika samostatných oddělených částí, z nichž každá zajišťuje samostatnou část aplikačního software, která má svůj cíl a v řadě případů i vlastní požadavky na časový limit v reálném čase. Služby z této skupiny umožňují spouštět procesy nebo vlákna a přiřadit jim příslušnou prioritu. Nejdůležitější službou RTOS z této skupiny je plánování procesů. Plánovač procesů řídí zpracování úloh aplikačního software, spouští je podle časového plánu odpovídajícím způsobem. Tato skupina služeb zahrnuje také rychlý a výkonný přerušovací systém a rutiny pro obsluhu chyb.

4.1.3 Meziprocesorová komunikace a synchronizace

Do druhé skupiny služeb jádra patří meziprocesorová komunikace a synchronizace. Tyto služby zajišťují předávání zpráv mezi procesy, se zabezpečením jejich předání před poškozením nebo ztrátou informací. Rovněž umožňují vzájemnou koordinaci procesů tak, aby účelně spolupracovaly jeden s druhým. Bez použití těchto služeb RTOS, mohou procesy komunikovat s porušenými informacemi nebo v jiném případě vzájemně se rušit mezi sebou.

4.1.4 Časovače

Většina jader RTOS poskytuje služby základních časovačů, jako je například možné zdržení (časový limit – deadline) nebo kontrola času prodlevy.

4.1.5 Řízení V/V jednotek

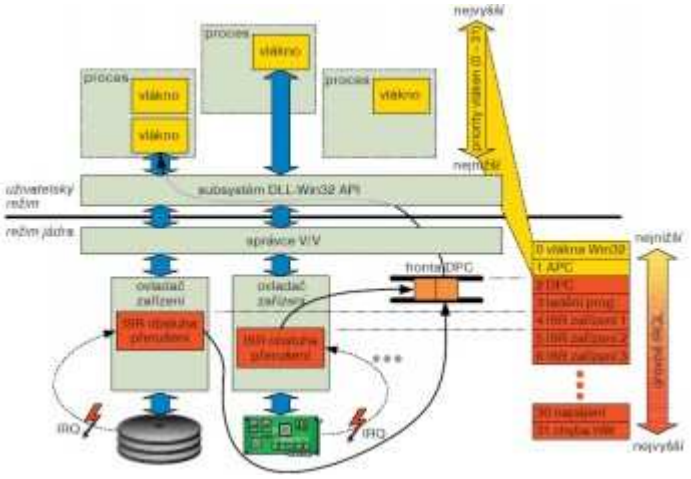
Většina jader RTOS, ale ne všechny, poskytuje skupinu služeb pro řízení V/V jednotek. Jsou-li tyto služby k disposici, umožňují pomocí ovladačů (driverů) standardizovaný přístup k mnoha hardwarovým jednotkám.

4.1.6 Dynamická alokace paměti

Jádro RTOS u některých systému poskytuje služby pro dynamickou alokaci paměti, pro kritické úlohy je ovšem někdy vhodnější statické přidělení paměti. Tato skupina služeb umožňuje procesům „vypůjčit si“ blok (oblast) paměti RAM pro dočasné použití v aplikačním softwaru. Často jsou tyto bloky paměti sdíleny mezi procesy, a takto umožňují rychlou komunikaci s velkým množstvím dat mezi procesy. Některé velmi malá jádra RTOS, které jsou určeny pro malé rozsahy paměti, neposkytují služby dynamické alokace paměti.

4.2 REÁLNÝ ČAS NA ARCHITEKTUŘE MS WINDOWS NT

Microsoft Windows je obvykle označován jako obecný operační systém, protože neumožňuje aplikacím pracovat na úrovni ovladačů s maskami přerušení a tím získat kontrolu nad operačním systémem.



Obrázek 4 – Prioritní uspořádání úloh v systémech MS Windows NT [7]

Systém byl původně navržen jako microkernel, ale realizován jako tzv. hybridkernel (struktura microkernelu, ale služby operačního systému běží z výkonnostních důvodů v *kernel módu* - privilegovaný režim). Na procesorech x86 využívá možnosti spouštění kódu s různými privilegii - tzv. Ring 0-3. Kernel mode je Ring 0 - má nejvyšší privilegia, běžný uživatelský kód pak běží v *user módu* - Ring 3 - nejnižší privilegia, Ring 1 a 2 není použit [10].

Bez výjimky všechny spuštěné aplikace (procesy) běží v uživatelském režimu, odkud nelze přímo přistupovat k hardwaru. Procesy jsou tak odkázány na ovladače zařízení, s nimiž komunikují prostřednictvím služeb subsystému Win32 API.

Každý proces (spuštěná aplikace) vlastní nejméně jedno prováděcí vlákno, které vykonává kód procesu. Jednotlivá vlákna všech procesů jsou střídavě zpracovávána procesorem. O tom, které vlákno má běžet, rozhoduje plánovač podle aktuálních potřeb vláken a přidělených priorit. Například vlákno s nejvyšší prioritou může být na stanovenou dobu uspáno nebo může čekat na uvolnění zdroje, čímž se umožní běh vláken s nižší prioritou. Na obdobném principu pracují časové a synchronizační funkce Win32 API (zkráceně časovače). Vlákno se jednoduše po vykonání příslušné operace uspí a čeká, dokud jej časovač znovu neprobudí.

Pro tyto účely systém nabízí několik nástrojů, které lze teoreticky nastavit s přesností až na jednu milisekundu. V praxi však uvedené přesnosti za určitých okolností dosahuje pouze multimediální časovač. Nicméně ani tento časovač nelze pro reálný čas použít. Důvodem je, že samotný plánovač podléhá ještě jinému prioritnímu uspořádání, tzv. úrovním IRQL (Interrupt ReQuest Level). Systém rozlišuje 32 těchto úrovní, kde nejvyšší prioritu má úroveň IRQL 31 a nejnižší IRQL 0. A právě plánovač (tím i veškerá vlákna) běží prioritně na nejnižší úrovni IRQL 0 nebo IRQL 1. Na vyšších úrovních běží všechny obsluhy přerušení a úlohy z fronty DPC (Deferred Procedure Call), které opět zakládají obsluhy přerušení. Zmíněné úlohy převážně nejsou kritické, avšak často bývají časově náročné.

Jinak řečeno, jakákoliv obsluha hardwarového přerušení, včetně jí odložených činností, je vykonána přednostně před libovolným vláknem. Probuzení vlákna se tak v závislosti na intenzitě úloh svázaných s obsluhou periferních zařízení

(spouštění aplikací, síťová komunikace, otevírání dokumentů apod.) může odložit až o stovky milisekund. Z toho je také zřejmé, že i když časovač probudí vlákno ve stanovený okamžik, neznamená to, že daná úloha se již bezodkladně vykoná celá, neboť kdykoliv může hardware opět generovat přerušení a tím nekompromisně odložit její vykonání.

Ze zkráceného popisu systému je zřejmé, že požadavky kladené na systémy reálného času systém Windows NT nenaplňuje. Přístup hardwaru není přímý a vyžaduje ovladač. Neexistuje spolehlivý časovač a úloha může být během své činnosti kdykoliv přerušena [7].

4.3 RTOS V PROSTŘEDÍCH NATIONAL INSTRUMENTS – WIND RIVER VXWORKS

VxWorks je operační systém pro řízení v reálném čase, který se vyznačuje RT mikro-jádrem wind, progresivní síťovou podporou, výkonným systémem souborů, řízením V/V jednotek, podporou C++ a runtimu. Základní funkce jádra mohou být doplněny připojením dalších komponent. Mikro-jádro zahrnuje většinu nástrojů pro podporu reálného času, mezi které patří především rychlý multitasking, podpora přerušení, preemptivní a cyklické plánování úloh. Mikro-jádro je navrženo s minimální režii systému, což umožňuje rychlou a deterministickou odezvu na externí události.

Patří k nejrozšířenějším operačním systémům reálného času zejména v oblasti průmyslových aplikací vestavěných systémů. Je příznivě hodnocen pro svou výkonnost, flexibilitu, kompatibilitu a rozšiřitelnost. Poskytuje velmi výhodné runtime prostředí pro vývoj vestavěných aplikací.

Je bezpečný i při použití v kritických aplikačních úlohách, například od antiblokovacího brzdového systému ABS u vozidel až po aplikace v meziplanetárním výzkumu. Systém je kompatibilní s řadou průmyslových standardů a lze jej provozovat na většině nejpoužívanějších základních jednotkách – CPU.

Vývojové prostředí Tornado II umožňuje snadný návrh vestavěných systémů a zahrnuje komplexní sadu křížových vývojových nástrojů a další vybavení včetně

komunikačních nástrojů pro spojení vývojového hostitelského systému s cílovým systémem pro aplikace.

5. VÝVOJOVÁ PROSTŘEDÍ NATIONAL INSTRUMENTS S REAL-TIME MODULEM

5.1 LABWINDOWS/CVI (ANSI C VÝVOJOVÉ PROSTŘEDÍ)

LabWINDOWS/CVI je profesionální integrovaný vývojový systém zaměřený na komplexní vývoj programů pro měřicí systémy na bázi standardizovaných rozhranní (IEEE 488, VXI, RS-232, ...) nebo zásuvných měřicích desek. V principu je založen na standardu jazyka ANSI C s rozšířením o knihovny podporující jak vlastní měření, tak matematické zpracování měřených dat a grafickou prezentaci výsledků. Značná pozornost je věnována problematice tvorby grafického uživatelského rozhranní, které je v současnosti samozřejmou součástí všech profesionálních programových aplikací.

Obrázek 5 – Logo LabWindows/ CVI

Systém je vybaven velmi kvalitními knihovnami funkcí orientovanými na podporu všech klíčových fází procesu automatizovaného měření (komunikaci s přístrojovými prostředky, sběr dat, jejich archivaci a zpracování, grafickou a numerickou prezentaci výsledků, přenos dat využívající počítačové sítě, atd.).

LabWindows/CVI Real-Time Modul rozšiřuje vývojové prostředí LabWindows/CVI o možnost vytvářet spolehlivé deterministické aplikace, které lze provozovat na hardwaru pracujícím v reálném čase. National Instruments poskytuje platformu pro vývoj aplikací pracujících v reálném čase, která kombinuje velmi flexibilní,vysoce výkonný software s robustním a modulárním hardwarem.

5.2 LABVIEW (GRAFICKÉ VÝVOJOVÉ PROSTŘEDÍ)

Grafický programovací jazyk LabVIEW™, platí ve světě měřicí a řídicí techniky za standard, s nímž jsou srovnávány ostatní programy. Toto prostředí slouží

pro vývoj kompletního systému, zajišťujícího řízení celého procesu sběru měřených dat, jejich analýzy a grafické prezentace. Sběr dat lze provádět z řady zařízení, zahrnujících přístroje vybavené sběrnicemi IEEE 488, VXI nebo sériovým rozhraním a zásuvných karet do osobních počítačů. Datové soubory jsou též přístupné ze sítě prostřednictvím protokolu TCP/IP, využitím dynamického předávání dat DDE mezi jednotlivými aplikacemi a z ostatních databázových programů pomocí SQL. Sebraná data lze analyzovat rozsáhlým matematickým aparátem, zahrnujícím knihovny pro generaci signálu, digitální filtry, statistiku, analýzu signálu v časové a frekvenční oblasti, regresní funkce, operace s poli a lineární algebru.

Obrázek 6 – Logo LabVIEW

Výsledkem použití takto komplexního systému pro tvorbu vlastních aplikací je výrazné zvýšení produktivity práce a pružnosti tvorby virtuálních přístrojů, jejichž výkon je na rozdíl od tradičních přístrojů omezen pouze výkonem použitého osobního počítače.

V LabVIEW™, pracujícím na platformách Microsoft Windows, Macintosh, SUN a HP-UX se programová aplikace vytváří v jednom okně formou blokového schématu, přičemž ve druhém okně vzniká odpovídající interaktivní přední panel virtuálního přístroje, mající vnitřní návaznost na program - blokové schéma. Pro tvorbu předního panelu slouží grafické uživatelské rozhraní, které obsahuje rozsáhlou knihovnu grafických objektů, pomocí nichž může panel získat podobu reálného přístroje. Samotná tvorba programu - blokového schématu je usnadněna tím, že zbavuje programátora starosti o řadu syntaktických detailů konvenčního programování. To umožňuje patentovaný model programování, založený na toku dat, nepoužívající lineární architekturu textově orientovaných jazyků. Protože pořadí vykonávání jednotlivých příkazů je v LabVIEW™ určeno tokem dat mezi bloky a ne sekvenčními řádky textu, lze vytvářet schémata, jež jsou vykonávána simultánně. Virtuální přístroje mají modulární uspořádání, kdy každý přístroj může pracovat

samostatně nebo být použit jako část jiného virtuálního přístroje. Každý z nich má vlastní ikonu s konektorem, umožňující hierarchickou stavbu programu kombinací propojených virtuálních přístrojů. LabVIEW™ je jediným grafickým programovým prostředím, které obsahuje kompilátor, generující optimalizovaný kód, jehož rychlost vykonání je srovnatelná s rychlostí kompilovaných programů v jazyce C.

LabVIEW Real-Time modul kombinuje grafické programování v LabVIEW s výkonností operačního systému reálného času, což umožňuje vytvářet deterministické aplikace. Vytvoříme virtuální přístroj (VI) v LabVIEW a spustíme VI na koncovém zařízení v reálném čase. Virtuální přístroj tedy může být spuštěn na platformě, která je určena pro řízení v reálném čase.

6. LABWINDOWS/CVI REAL-TIME: SPOLEHLIVOST A DETERMINISMUS V ANSI C

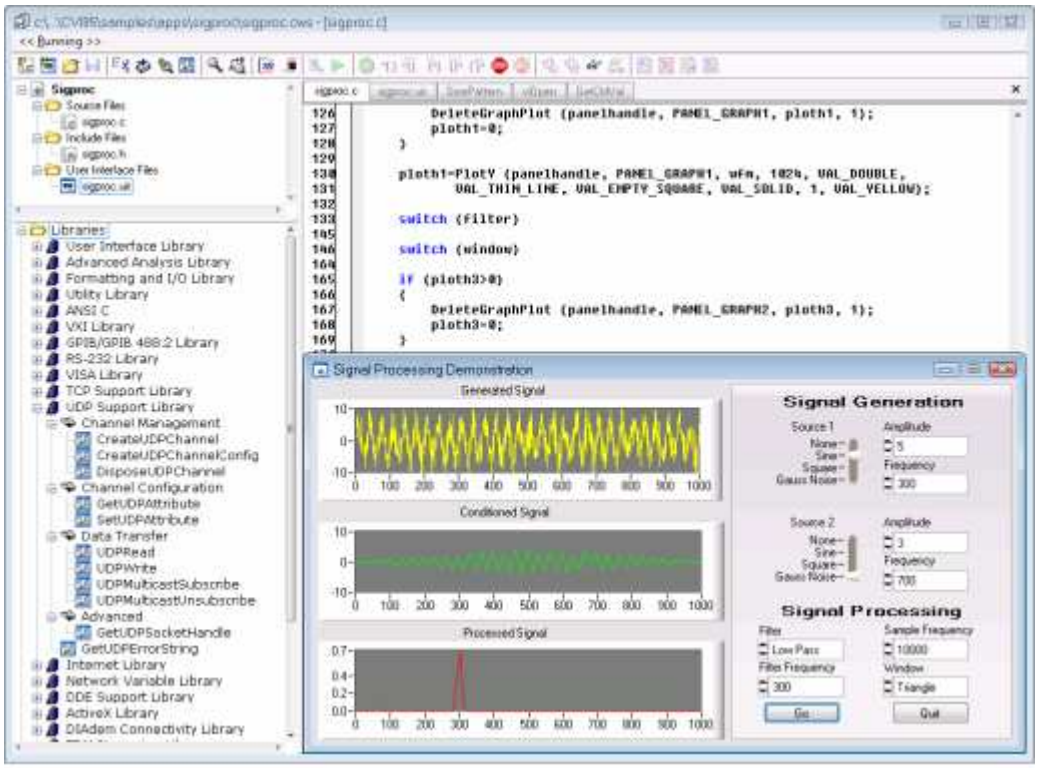
Zabudované programovací systémy vyžadují kvalifikované znalosti různých nástrojů, programovacích technik a možností integrace hardwaru a softwaru. To často znamená časově náročné studium pro inženýry, kteří potřebují systém reálného času používat, ale nejsou zkušení vývojáři aplikací v reálném čase. Navíc, zatímco roste nabídka produktů pro využití reálného času, je stále hodně práce v oblasti integrace a programování pro vývojáře, aby mohl software hladce spolupracovat s hardwarem.

Dalším obtížným úkolem v programování v reálném čase je práce s aplikacemi, které vyžadují nejen testování ale i deterministickou kontrolu. Použití nepropojených testovacích a kontrolních nástrojů vyžaduje další odbornou způsobilost na tomto poli a další integrační práci pro synchronizování těchto oddělených aplikací.

6.1 VÝHODY LABWINDOWS/CVI REAL-TIME

Modul LabWindows/CVI Real-Time poskytuje platformu, která zjednodušuje programování v reálném čase, integraci hardwaru a kombinaci testování a kontroly.

Již od své první verze A0 z roku 1987 je LabWindows/CVI dlouhodobě spolehlivým vývojovým prostředím v ANSI C. Od té doby LabWindows/CVI přidává moderní programovací technologie jako například ActiveX a .NET a zároveň se soustředí na stabilitu a zpětnou kompatibilitu. Protože LabWindows/CVI je založeno na otevřeném ANSI C standardu, lze v něm snadno tvořit nové aplikace stejně jako přenášet stávající kód do koncového RT hardwaru s minimálními modifikacemi. Tato možnost začlenit starší kód do vyvíjené aplikace snižuje dobu vývoje pro aplikace reálného času.



Obrázek 7: LabWindows/CVI poskytuje zabudované vývojové prostředí, ideální pro tvoření aplikací, které kombinují software a hardware. [3]

Vzhledem k tomu, že LabWindows/CVI Real-Time je těsně provázán s National Instruments hardwarem, je vývoj aplikací pro koncový RT hardware stejně jednoduchý jako vývoj aplikací pro hardware ve Windows. Použití existujících NI ovladačů (jako jsou NI-DAQmx a NI-VISA) umožňuje přístup k tisícům zařízení pro sběr dat a přístrojům bez toho, aby bylo nutné trávit čas výrobou vlastních ovladačů RT zařízení nebo se obávat o programování registrů.

6.2 KONCOVÝ REAL-TIME HARDWARE A I/O

Typický RT systém zahrnuje vývojový nástroj, takzvaný hostitelský počítač a výkonný nástroj, takzvaný koncový RT hardware. Obecně, je vyvinuta konkrétní aplikace v LabWindows/CVI na hostitelském počítači s obecným operačním systémem jako jsou například Microsoft Windows, ta je pak stažena do koncového RT hardwaru přes ethernet a následně je aplikace spuštěna na koncovém RT hardwaru (

Obrázek 8).

Obrázek 8: Použití LabWindows/CVI Real-Time pro vývoj, nahrávání a spuštění RT systémů. [3]

LabWindows/CVI Real-Time momentálně podporuje dva typy koncového RT hardwaru – PXI (PCI eXtensions přístroje) a desktopové počítače. PXI je robustní platforma založená na PC standardu určená pro měření a automatizaci systémů.

Typický systém se skládá z chassis[8], zabudovaného kontroléru a námi vybraného I/O modulu. Kontrolér má tutéž funkci jako stolní CPU ale v náročné, průmyslové podobě. Pokud je toto použito jako RT systém, LabWindows/CVI aplikace je stažena do zabudovaného procesoru na PXI kontroléru. Během toho, co je aplikace spuštěna, má přístup do dat na I/O modulech v systému. Jakmile je vše připraveno pro spuštění systému, je možno odpojit vývojový aparát a nechat systém spuštěný samostatně, tj. bez obrazovky, klávesnice nebo myši. [4]

Obrázek 9: PXI systémy jsou dostupné se 4, 8 nebo 18 sloty. [3]

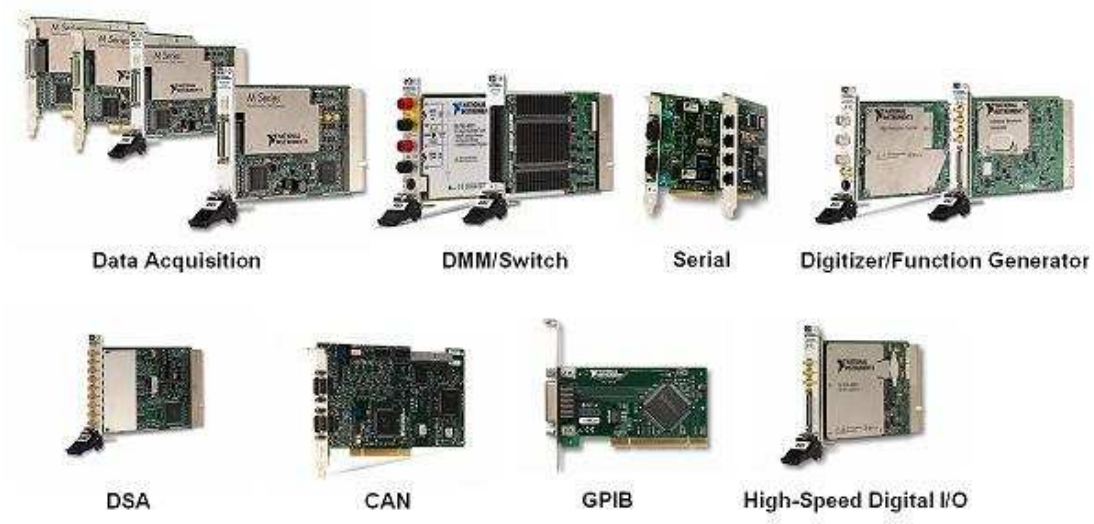
S LabWindows/CVI Real-Time je možné používat jako koncový RT hardware také desktopové počítače. S touto možností je využíván stávající PC hardware jako nástroj pro tvorbu spolehlivých a deterministických aplikací. Lze tak využít nejmodernější PC technologie, včetně PCI Express a Pentium 4 procesory pro dosažení výkonu v reálném čase na množství PCI hardwaru.

[8] chassis = šasi

Obrázek 10: Standardní počítač nebo single-board počítače jako koncový Real-Time hardware.

Podporované I/O

Aby byly plně využity výhody RT podpory, je nutné přidat do systému I/O. LabWindows/CVI podporuje celou škálu NI I/O, včetně zařízení pro sběr dat, sériových zařízení, digitalizátorů, generátorů funkcí, vysokorychlostních digitálních zařízení, DMM, přepínačů a CAN ovladačů. Komunikace s těmito zařízeními je umožněna pomocí plně funkčních ovladačů zařízení, obzvláště NI-DAQmx, NI-VISA, NI-CAN, NI-DMM, NI-Scope, NI-HSDIO a NI-Switch.



Obrázek 11: Množství NI I/O možností je dostupné s LabWindows/CVI Real-Time [3]

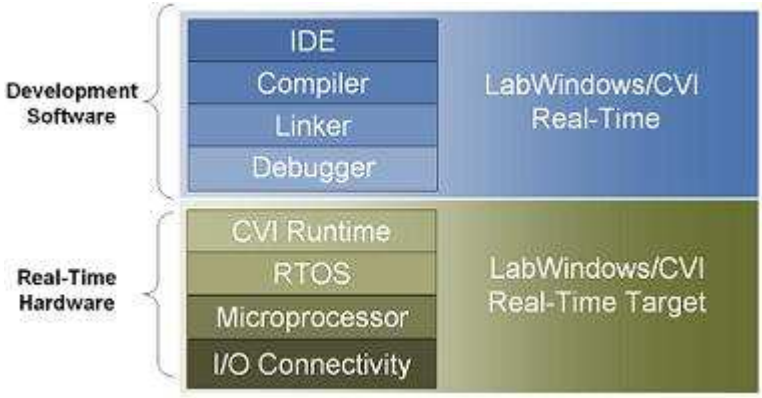
6.3 KLÍČOVÉ VLASTNOSTI

Zavedení LabWindows/CVI Real-Time modulu umožnilo využít vlastností LabWindows/CVI s mnoha RT vlastnostmi včetně těsného provázání softwaru a hardwaru, generování RT kódu, přímé spouštění na koncovém RT hardwaru přes ethernet, RT provázané knihovny, program na kopírování souborů, dálkové ladění, nastavení měření a automatizace (Measurement & Automation Explorer - MAX) a PID regulace.

6.3.1 Těsné provázání softwaru a hardwaru

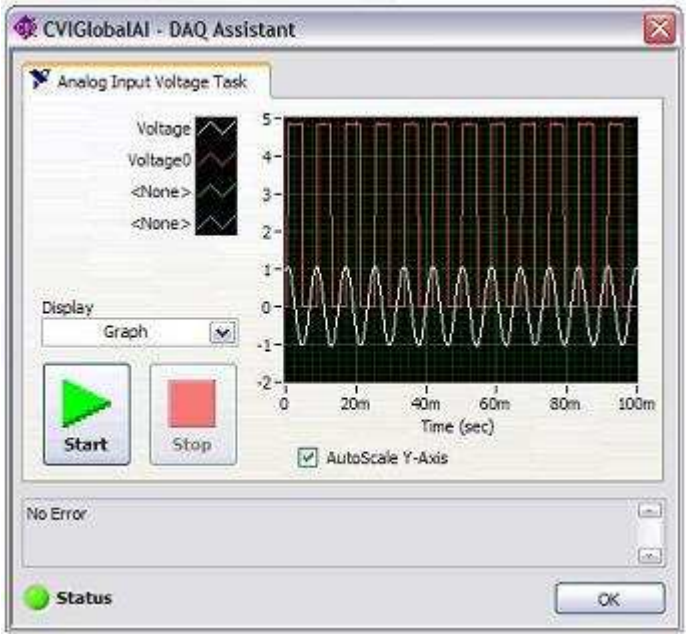
National Instruments poskytuje vysoce integrovanou RT platformu, která nabízí komerční předem hotové softwarové a hardwarové komponenty, které hladce spolupracují a tím zajišťují ucelená řešení. Každý RT hardware používá standardní komponenty jako jsou mikroprocesor, volatilní a nevolatilní paměť (paměť závislá a nezávislá na napájení), I/O sběrnicové rozhraní a operační systém reálného času.

Zabudovaný software se skládá z operačního systému reálného času, řídicí software a upravená verze LabWindows/CVI spouštěcího nástroje. Protože jsou všechny LabWindows/CVI Real-Time spouštěcí platformy založeny na této všeobecné architektuře softwaru a hardwaru, zapojení mezi těmito dvěma komponenty je velmi snadné.



Obrázek 12: National Instruments poskytuje ucelenou softwarovou a hardwarovou platformu pro RT [3]

Jako přídavek k zabudované NI-DAQmx knihovně, o které byla zmínka výše, LabWindows/CVI poskytuje DAQ asistenta, interaktivní rozhraní pro získávání dat (data acquisition driver framework). DAQ asistent interaktivně definuje měřicí úkol, určuje měřicí schopnost DAQ zařízení, které je specifikováno, a generuje moduly v C, které provádějí sběr dat. Úzká integrace DAQ asistenta a NI-DAQmx s LabWindows/CVI poskytuje inženýrům, kteří provádějí sběr dat, ten nejúčinnější a nejvýkonnější způsob provádění měření v ANSI C prostředí.



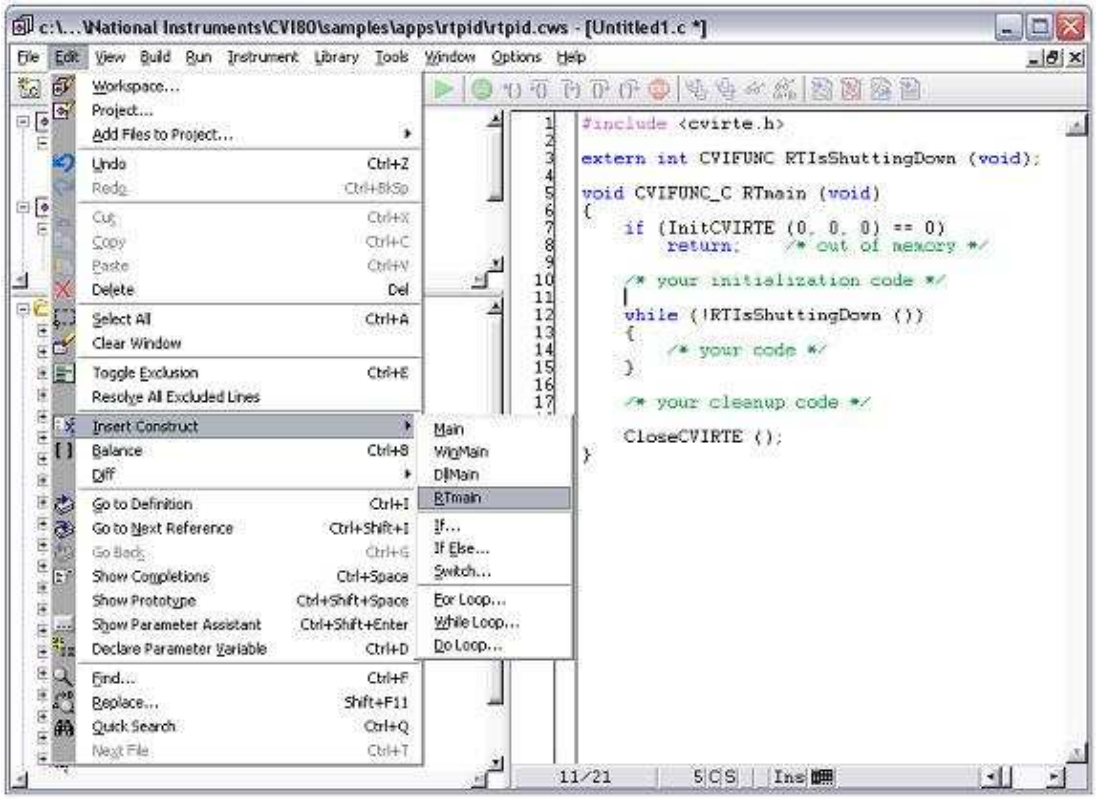
Obrázek 13: Vytvoření úlohy pro sběr dat pomocí DAQ asistenta [3]

LabWindows/CVI má vedoucí postavení v průmyslovém odvětví v oblasti ovládání a propojování přístrojů společně s NI LabVIEW softwarem napříč sítí přístrojových ovladačů (více než 4000 přístrojových ovladačů) od více než 200 prodejců. Tyto ovladače je snadné použít k naprogramování aplikací. LabWindows/CVI povýšila ovládání přístrojů na vyšší úroveň s přístrojovým I/O asistentem. S Přístrojovým I/O Asistentem je možné generovat kód pro komunikaci se zařízeními jako jsou Ethernet a GPIB přístroje bez použití přístrojových ovladačů.

Přístrojový I/O asistent nabízí jednoduché rozhraní pro rychlé prototypové aplikace a automatické rozbory přístrojových dat bez jakéhokoli programování. Můžeme snadno importovat kód generovaný do jakékoli existující aplikace a odpadá veškeré pracné psaní přístrojového propojení, základní komunikace a kód řetězových rozborů.

6.3.2 Generování RTmain( ) kódu

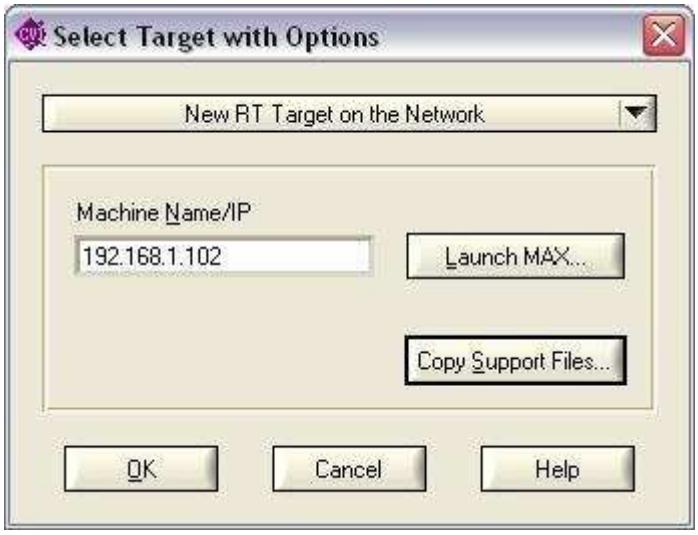
Generování RTmain() kódu nalezne uplatnění při automatickém tvoření šablon kódu pro aplikací pracujících v reálném čase. Šablona obsahuje vhodné startovací body pro přidání inicializačního kódu. Funkce RTmain() je automaticky nahrána a spuštěna po spuštění na koncovém RT hardwaru. Do aplikace se RTmain() funkce přidává pomocí položky v menu Edit » Insert Construct » RTmain.



Obrázek 14: Použití položky v menu Insert Construct pro automatické vytvoření RTmain() inicializačního kódu a šablony, aby bylo možné začít psát vlastní kód [3]

6.3.3 Přímé spouštění RT přes Ethernet

Spouštět aplikace přímo na koncovém RT hardwaru přes Ethernet je možné pomocí LabWindows/CVI Real-Time Select Target v okně Možnosti (Options). S touto možností je možné vybrat koncový RT hardware pomocí názvu přístroje nebo IP adresy bez nutnosti opouštět LabWindows/CVI vývojové prostředí. Pokud si nejsme jisti dostupností RT hardwaru, můžeme spustit Measurement and Automation Explorer (MAX), kde si zobrazíme dostupný RT hardware v dané síti. Po kliknutí na tlačítko run, LabWindows/CVI automaticky vytvoří RT naši aplikaci a zkopíruje je na koncový RT hardware přes ethernet.



Obrázek 15: Spouštění aplikace přímo na koncovém RT hardwaru přes Ethernet [3]

6.3.4 Real-Time knihovna vláken

Jedním z důležitých aspektů programování v reálném čase je řízení priorit jednotlivých vláken. Vlákno s vyšší prioritou nemůže být přerušeno vláknem s nižší prioritou uvnitř jednoho systému. Existuje sedm různých priorit vláken. Pro

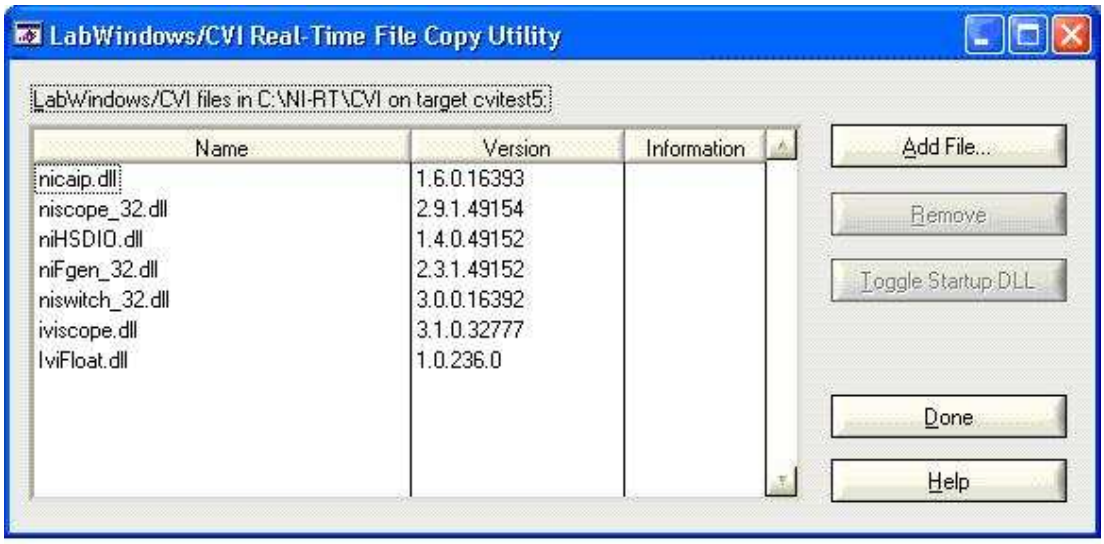
zjednodušení práce s vlákny zavádí LabWindows/CVI speciální funkce, které umožňují vytváření, přenos a rušení vláken.

```
// Create worker threads in new thread pool to run the high and low priority tasks
CmtNewThreadPool (2, &gThreadPool);
CmtScheduleThreadPoolFunctionAdv (gThreadPool, HighPriorityLoop, 0,
    THREAD_PRIORITY_ABOVE_NORMAL, 0, 0, 0, 0, 0);
CmtScheduleThreadPoolFunctionAdv (gThreadPool, LowPriorityLoop, 0,
    THREAD_PRIORITY_NORMAL, 0, 0, 0, 0, 0);
```

Obrázek 16: Nastavení priorit vláken [3]

6.3.5 Utilita pro kopírování souborů (File Copy Utility)

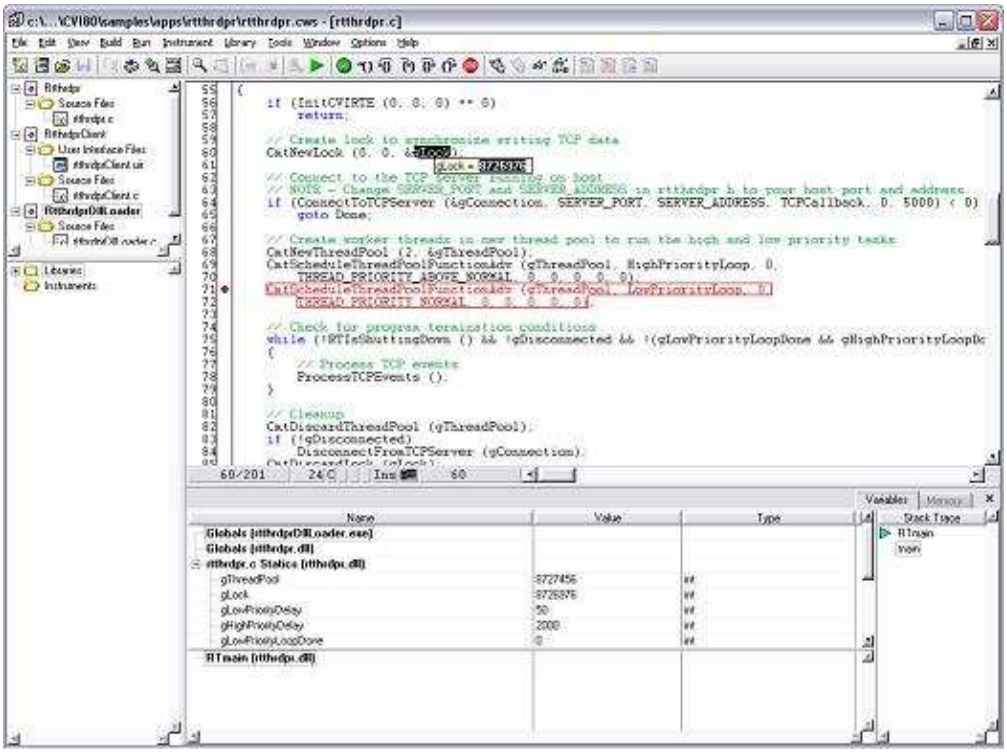
V momentě kdy vytvoříme naši RT aplikaci, LabWindows/CVI automaticky nahrají naši aplikaci a všechny staticky linkované DLL do koncového RT hardwaru. Nicméně, pokud si přejeme nahrát do RT hardwaru další soubory jako např. konfigurační soubory nebo dynamicky volané DLL, musíme použít utilitu pro kopírování souborů (File Copy Utility). Tato utilita také dokáže specifikovat, která DLL se má spustit po startu systému.



Obrázek 17: Použití utility pro kopírování souborů do RT zařízení [3]

6.3.6 Vzdálené odlaďování

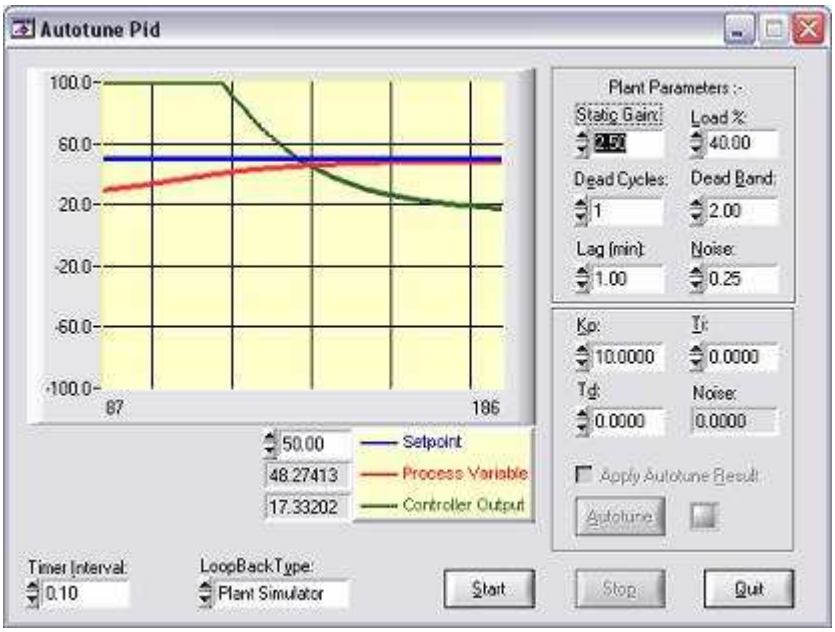
Po dokončení naší RT aplikace a jejím přenosu do koncového RT hardwaru, je dobré mít možnost odladění již připojeného systému přes I/O. S LabWindows/CVI lze jednoduše vytvořit DLL určenou pro odladění a tuto DLL poté odladit přímo z lokální instance LabWindows/CVI. Všechny známé vlastnosti jako krokování, body přerušení, sledování proměnných jsou plně funkční. Dokonce je možné měnit hodnoty proměnných v koncovém RT hardwaru přímo z lokální stanice s LabWindows/CVI. Všechny informace získané během odlaďování jsou automaticky přenášeny mezi lokální stanicí a RT hardwarem přes TCP.



Obrázek 18: Vzdálené odlaďování RT aplikace přes ethernet [3]

6.3.7 Nástroje pro PID regulaci

Modul LabWindows/CVI Real-Time obsahuje také nástroje pro pohodlnou PID regulaci, které umožňují vytvářet sofistikované regulační smyčky. Sada nástrojů obsahuje funkce jako externí reset zpětné vazby, algoritmus výpočtu regulační odchylky, apod. [2][3]



..

Obrázek 19: Sofistikovaná PID regulace [3]

7. PROGRAMOVÁNÍ V LABVIEW S MODULEM PRO ŘÍZENÍ V REÁLNÉM ČASE

Jak bylo zmíněno výše při návrhu systému pracujícího v reálném čase je třeba zvolit dvě hlavní komponenty:

- vývojový software
- hardwarovou platformu

Hardwarová platforma je stejná pro obě prostředí LabWindows/CVI i LabVIEW, proto se v této kapitole zaměříme zejména na grafické RT programování v prostředí LabVIEW s modulem Real Time.

7.1 ÚVOD DO MODULU REÁLNÉHO ČASU V PROSTŘEDÍ LABVIEW

Většina aplikací vytvořených v LabVIEW je provozována na běžných operačních systémech jako Microsoft Windows, Linux nebo Mac OS. Některé aplikace však vyžadují zpracování v reálném čase, které běžné operační systémy nemohou zajistit. Modul LabVIEW Real-Time rozšiřuje možnosti LabVIEW o schopnost práce v reálném čase.

Virtuální přístroje (VI) vytvořené v modulu Real-Time pro LabVIEW mohou být spuštěny v operačních systémech reálného času NI ETS a Wind River VxWorks.

7.1.1 Systémové komponenty modulu Real-Time

RT modul se skládá ze softwarových a hardwarových komponent. Softwarovými komponentami rozumíme LabVIEW, RT Engine, projekty a VI

vytvořené v LabVIEW. Hardwarovými komponentami jsou hlavní (hostitelský) počítač a koncový hardware pracující v reálném čase.

7.1.1.1Hlavní (hostitelský) počítač

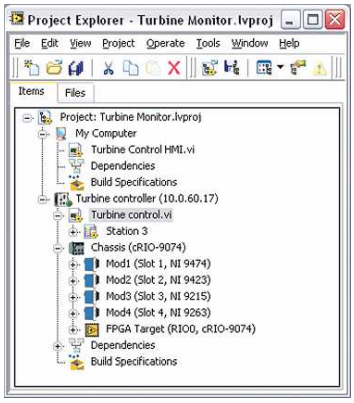
Hostitelský počítač je počítač na kterém je nainstalované grafické prostředí LabVIEW s modulem Real Time. Na tomto počítači se vyvíjí VI pro systém reálného času. Po vytvoření VI reálného času se vytvořený virtuální přístroj nahraje do koncového hardwaru pracujícího v reálném čase. Na hostitelském počítači může být spuštěn virtuální přístroj (VI), který komunikuje s virtuálním přístrojem (VI) spuštěným v koncovém hardwaru reálného času a který tak vytváří uživatelské rozhranní.

7.1.1.2LabVIEW

Virtuální přístroj (VI) se navrhuje v grafickém prostředí LabVIEW s modulem Real Time. Tento modul doplňuje LabVIEW o další nástroje pro vytváření, odlaďování a rozmisťování deterministických VI.

7.1.1.3LabVIEW Projects

LabVIEW projekty se používají pro seskupení souborů, pro vytváření samostatných RT aplikací, pro rozmístění a nahrávání VI a jiných souborů na koncový RT hardware. Při každém uložení projektu se vytváří projektový soubor (.lvproj), který obsahuje odkazy na projektové soubory, informace o konfiguraci, sestavení a rozmístění.



Obrázek 20: Projektový průzkumník [1]

7.1.1.4RT Engine

RT Engine je varianta LabVIEW, která běží na koncovém RT hardwaru. RT Engine zajišťuje chod VI, který je nahrán do koncového RT hardware. Poskytuje výkonnost deterministického řízení v reálném čase:

- RT Engine běží na operačním systému reálného času (RTOS), který zajišťuje včasné ošetření událostí exekučního systému LabVIEW i všech ostatních služeb.
- RT Engine běží pouze na RT Serii Hardwaru. Na koncovém RT hardwaru je možné provozovat pouze VI a ovladače přístrojů pro RT aplikace, aby nedocházelo k zbytečnému zatěžování procesoru.
- RT koncové zřízení nepoužívá virtuální paměť, která může způsobit nepředvídatelnou výkonnost.

7.1.1.5koncový RT hardware

Koncový RT hardware se odkazuje na RT serii hardwaru, na které běží RT Engine a VI vytvořené v LabVIEW. Síťové zařízení RT série je síťová hardwarová platforma s integrovaným procesorem a operačním systémem reálného času, na kterém běží RT Engine a VI. Pro komunikaci a ovládání VI na koncovém RT

hardwaru lze použít hostitelský počítač, který je připojený přes ethernet. Příklady série síťových RT hardwarů:

- NI RT Série PXI Controller
- NI CompactRIO Series
- NI RT Série FieldPoint and Compact FieldPoint
- NI 1450 Série Compact Vision System
- Desktop PCs jako RT koncové zařízení

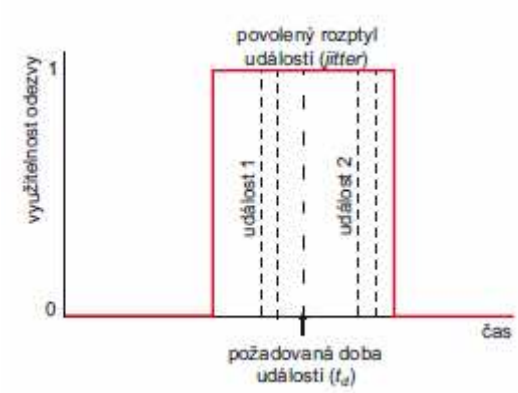
7.1.1.6USB paměťové zařízení

Modul reálného času zahrnuje podporu USB paměťových zařízení (jako flash disky nebo externí USB disky) připojených ke koncovému RT hardwaru. Takto lze ke koncovému RT hardwaru připojit externí disk a přistupovat na něj z VI běžícím na koncovém RT hardwaru. Každému připojenému zařízení je automaticky přiřazeno písmeno disku U:, každému dalšímu zřízení pak V:, W:, X:, atd.

7.2 VYTVÁŘENÍ DETERMINISTICKÝCH APLIKACÍ V MODULU REÁLNÉHO ČASU

Jak bylo uvedeno dříve, determinismus je charakteristika systému, která popisuje jak konzistentně systém odpovídá na externí události nebo vykonává operace v daném časovém limitu. Jitter[9] je mírou, ve které načasování provedení nesplňuje deterministická očekávání. Většina aplikací v reálném čase vyžaduje konzistentní chování časování uvnitř malého rozptylu hodnot.

[9] Jitter = povolený rozptyl události



Obrázek 21: Jitter – jeden ze základních pojmů systémů reálného času [11]

7.2.1 Vytvoření deterministické aplikace v modulu reálného času

Jednoprocesorový systém dosahuje multitaskingu (zpracování několika úloh současně) tím, že velmi rychle střídá na procesoru běžící procesy, takže uživatel počítače má dojem, že běží současně. Naproti tomu víceprocesorový systém je schopen dosáhnout opravdového paralelního zpracování procesů.

Zpracování ve více vláknech (multithreading) používá koncept multitaskingu v jedné aplikaci jejím rozdělením na menší úlohy, které jsou zpracovány v krátkém čase po různých vláknech systému. Vlákno je úplně nezávislý tok vykonání jedné úlohy uvnitř aplikace. Vícevláknová (multithreaded) aplikace zvyšuje výkonnost procesoru, protože procesor může v době, kdy není vytížen, pracovat na jiném vláknu. Např. v momentě, kdy aplikace čte z nebo zapisuje do souboru, vykonává I/O nebo čeká na odezvu uživatele, může procesor zpracovávat jinou úlohu.

Operační systém reálného času (RTOS) běžící na koncovém NI RT hardwaru používá kombinaci cyklické obsluhy a preemptivního rozvrhování k vykonání úlohy jednoho vlákna v rámci vykonávané aplikace. Cyklická obsluha obsluhuje vlákna se stejnou prioritou. Stejné díly procesorového času jsou rozděleny mezi vlákna se stejnou prioritou. Například vláknům s normální prioritou je přidělen čas 10ms.

Procesor vykonává úlohu po dobu 10ms a pokud ji nestihne dokončit musí úloha čekat na další přidělení procesorového času v dalším kole. Naproti tomu preemptivní rozvrhování znamená, že pokud má být zpracováno vlákno s vyšší prioritou v době kdy procesor pracuje na vláknu s nižší prioritou, je toto zpracování ihned přerušeno a procesor začne pracovat na vláknu s vyšší prioritou. Vlákno s nejvyšší prioritou (Time-critical priority) je vždy vykonáno přednostně.

Hodnota priority vláken, která jsou zpracovávána v LabVIEW, je mezi vysokou (High Priority) a nejvyšší (Time-critical priority) prioritou. Uvnitř VI lze nastavit relativní prioritu časové smyčky vzhledem k jiným časovým strukturám.

7.2.1.1Dělení úloh při vytváření vícevláknových aplikací

Správné vykonání deterministické aplikace závisí na tom, zda-li jsou její jednotlivé úlohy zpracovávány deterministicky, vždy ve vymezeném čase. Proto potřebují deterministické úlohy přidělení dostatečných zdrojů procesoru. Dělení na menší úlohy pomáhá zajistit, aby každá úloha mohla získat požadované množství procesorového času.

Abychom získali dostatečné množství procesorového času, je potřeba oddělit deterministické úlohy od všech ostatních úloh. Například jestliže daná aplikace sbírá data v pravidelných intervalech a ukládá je na disk, je potřeba zacházet s časováním a kontrolou získávaných dat deterministicky. Nicméně, ukládání dat na disk je přirozeně nedeterministická úloha, protože nelze odhadnou dobu čtení nebo zápisu dat na disk. Tato doba závisí především na samotném hardwaru a na dostupnosti hardwarových zdrojů.

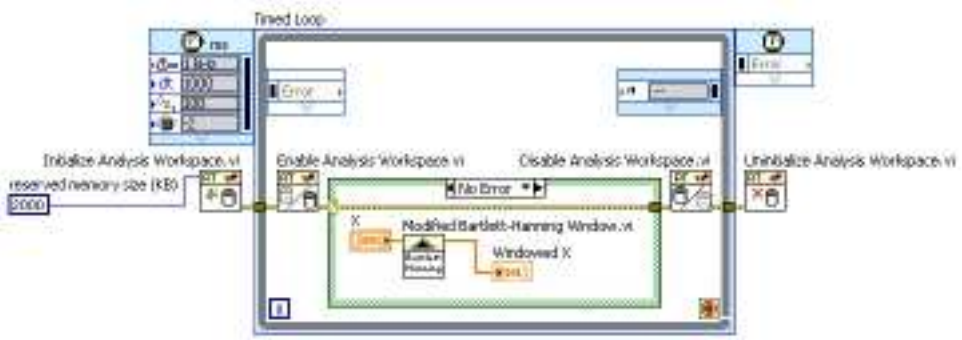
Ovládání a časování deterministických úloh lze zajistit časovými smyčkami nebo použitím různých VI s různými prioritami.

7.2.1.2Potlačení časového rozptylu použitím pracovního prostoru (workspace) při matematických operacích a zpracování signálu.

Mnoho analytických VI z palety matematických funkcí a zpracování signálu vyžaduje přidělení velkého množství zdrojů pro své vnitřní výpočty. Požadavek na větší množství zdrojů od operačního systému může způsobit časový rozptyl, který je nežádoucí. Pro potlačení tohoto časového rozptylu lze pomocí techniky pracovního prostoru (workspace) dopředu přidělit požadované množství zdrojů. K tomuto účelu se používají nástroje pro analýzu v reálném čase (Real-Time Analysis Utilities) v tomto pořadí:

1. Inicializace pracovního prostoru (workspace)
2. Spuštění VI s nejvyšší prioritou (Time-critical) nebo časové smyčky s vysokou prioritou (high priority)
3. Aktivace pracovního prostoru (workspace)
4. Provedení VI z palety matematických funkcí a zpracování signálu s nejvyšší prioritou (Time critical)
5. Deaktivování pracovního prostoru (workspace)
6. Ukončení VI s nejvyšší prioritou (Time-critical) nebo časové smyčky s vysokou prioritou (high priority)
7. Odinicializace pracovního prostoru (workspace)

Následující blokový diagram ukazuje typické použití pracovního prostoru (workspace)

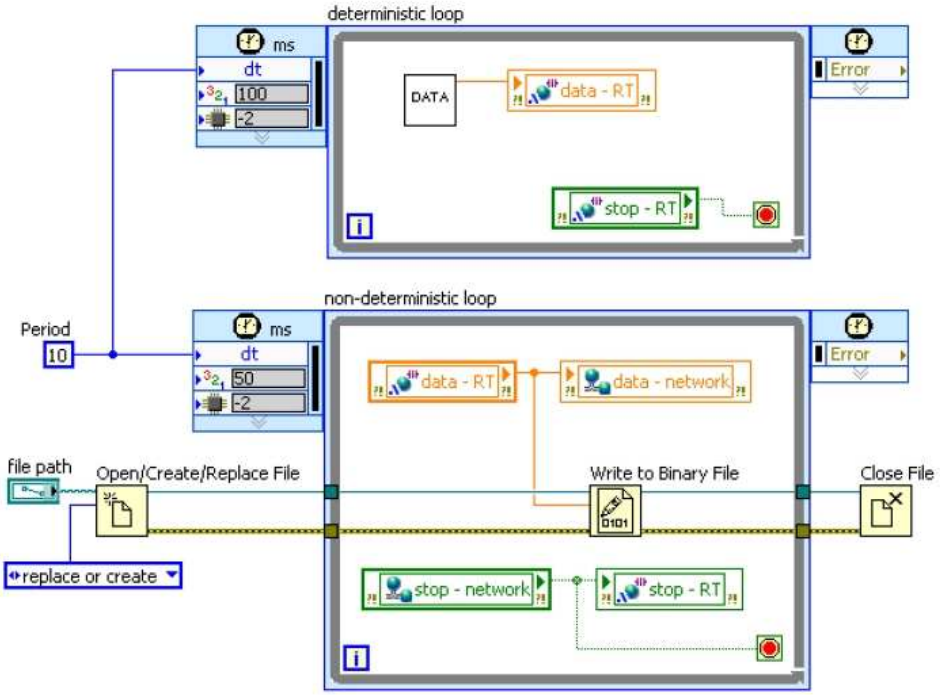


Obrázek 22: Typické použití pracovního prostoru [1]

Pozn. Ne všechny VI z palety matematických funkcí a zpracování signálu dokáží využít pracovní prostor pro potlačení nežádoucího časového rozptylu.

7.2.2 Vytvoření deterministické aplikace použitím časové smyčky

Abychom získali dostatečné množství procesorového času, je potřeba oddělit deterministické úlohy od ostatních úloh a umístit je do různých časových smyček uvnitř VI v koncovém RT hardwaru. Časová smyčka provede jednu iteraci smyčky podle periody a priority, které má přiděleny. Čím vyšší prioritu má smyčka, tím vyšší relativní prioritu má daná struktura k ostatním časovým strukturám v blokovém diagramu. Lokální nebo vzdálené sdílení dat celou sítí je zajišťováno pomocí deterministických komunikačních metod.



Obrázek 23: Použití deterministické a nedeterministické časové smyčky [1]

Uvedený blokový diagram ukazuje VI v koncovém RT hardware, který obsahuje dvě časové smyčky na řízení sběru dat a na záznam dat. Deterministická smyčka s nastavenou prioritou 100 obsahuje subVI, který zajišťuje sběr dat při každé iteraci smyčky. Deterministická smyčka sdílí sebraná data s nedeterministickou smyčkou použitím sdílené proměnné – „data“. Sdílené proměnné s aktivovanou pamětí FIFO pracující v reálném čase dokáží sdílet data deterministicky z VI běžícího na koncovém RT zařízení bez ovlivnění determinismu VI. Nedeterministická smyčka s nastavenou prioritou 50 zapisuje data ze sdílené proměnné „data“ na disk koncového RT zařízení.

7.2.2.1Nastavení periody časové smyčky

Vzhledem k preemptivní povaze časové smyčky se může stát, že si smyčka vyhradí pro sebe veškerý zdroj procesoru, tak, že nedovolí úlohám s nižší prioritou získat procesorový čas pro jejich potřebu.
Z tohoto důvodu je potřeba nastavit dostatečně dlouhou periodu časové smyčky, aby se stihla vykonat deterministická úloha a ještě zbyl čas pro vykonání úloh s nižší prioritou. Správným časováním smyčky lze takto uvolnit zdroje procesoru pro vykonání úloh s nižší prioritou v blokovém diagramu.

7.2.3 Vytvoření deterministické aplikace požitím VI s různými prioritami

Abychom získali dostatečné množství procesorového času, je potřeba oddělit deterministické úlohy od ostatních úloh a umístit je do různých VI. Takto vytvořeným VI lze přidělit různé priority a umístit je do různých kategorií realizovaných systémů. Podle těchto vlastností VI jim jsou přidělovány procesorové zdroje.

LabVIEW přiděluje každému VI vlákno podle priority a podle realizovaného systému do kterého patří. Podle tohoto rozdělení jsou pak jednotlivá vlákna obsloužena.

7.2.3.1Přidělení priorit VI

Prioritu VI lze změnit pravým klikem na VI v projektovém průzkumníku (Project Explorer). Každému VI lze přidělit jednu z následujících priorit:

- pozadí - background priority (nejnižší)
- normální - normal priority (defaultní)
- nad normální - above normal priority
- vysoká - high priority
- časově kritická - time-critical priority (nejvyšší)

Priorita každého VI je defaultně nastavena na hodnotu „normální“ (normal priority). Nicméně subVI dědí prioritu po VI, který jej zavolal.

7.2.3.2Přidělení VI do realizovaných systémů

Realizovaný systém, do kterého VI patří, lze změnit pravým klikem na VI v projektovém průzkumníku (Project Explorer). Každý VI lze zařadit do jedné z následujících kategorií:

- uživatelské rozhranní - user interface
- standardní - standard
- přístrojové I/O - instrument I/O
- sběr dat - data acquisition
- jiná 1 - other 1
- jiná 2 - other 2

Defaultně jsou všechny VI vykonávány v rámci „standardního“ realizovaného systému. Navíc lze subVI zařadit do stejné kategorie, ve které se nachází jeho volající VI.
Každý realizovaný systém (kromě uživatelského rozhranní – user interface) má vlastní frontu VI, která je vykonávána podle jejich priorit. Například předpokládejme, že v jednom systému se nacházejí tři VI se stejnou prioritou. Potom je vždy vybráno vlákno, které je na začátku fronty a je mu přidělen určitý procesorový čas. Po uplynutí tohoto času je vlákno zařazeno na konec fronty a začne se vykonávat další vlákno v pořadí. Pokud je vlákno dokončeno, systém jej vyřadí z fronty.

Systém uživatelského rozhranní (user interface) zajišťuje všechny úlohy týkající se uživatelského rozhranní. Ostatní systémy tyto úlohy nemohou vykonat. Pokud VI potřebuje obnovit uživatelské rozhranní, potom jeho systém předá odpovědnost systému uživalského rozhranní (user interface), který tuto úlohu zajistí.

7.2.3.3Nastavení priority VI

Vzhledem k preemptivní povaze VI se může stát, že si VI vyhradí pro sebe veškerý zdroj procesoru, tak, že nedovolí úlohám s nižší prioritou a FTP serveru na koncovém RT hardwaru získat procesorový čas pro jejich potřebu.
Z tohoto důvodu je potřeba periodicky nastavit VI tak, aby zbyl čas pro vykonání úloh s nižší prioritou bez ovlivnění determinismu systému. Správným časováním VI lze takto uvolnit procesorové zdroje.

7.3 INSTALACE A KONFIGURACE MODULU REÁLNÉHO ČASU

RT modul se instaluje na vývojový počítač nebo na hostitelský počítač a software se konfiguruje a instaluje na koncovém RT hardwaru.

NI Measurement a Automation Explorer (MAX – Měřicí a Automatizační průzkumník) zajišťuje přístup k NI zařízením a systémům. MAX je schopen komunikovat se zasíťovaným RT hardwarem, se zvanými vzdálenými systémy, umístěnými na stejné pomocné síti jako počítač, na kterém je MAX nainstalován. RT modulový software může být nainstalován na hostujícím počítači. Pak je možné používat MAX pro konfiguraci RT hardwaru a instalovat RT modulový software a ovladače na koncový hardware.

7.3.1 Instalace RT modulového softwaru

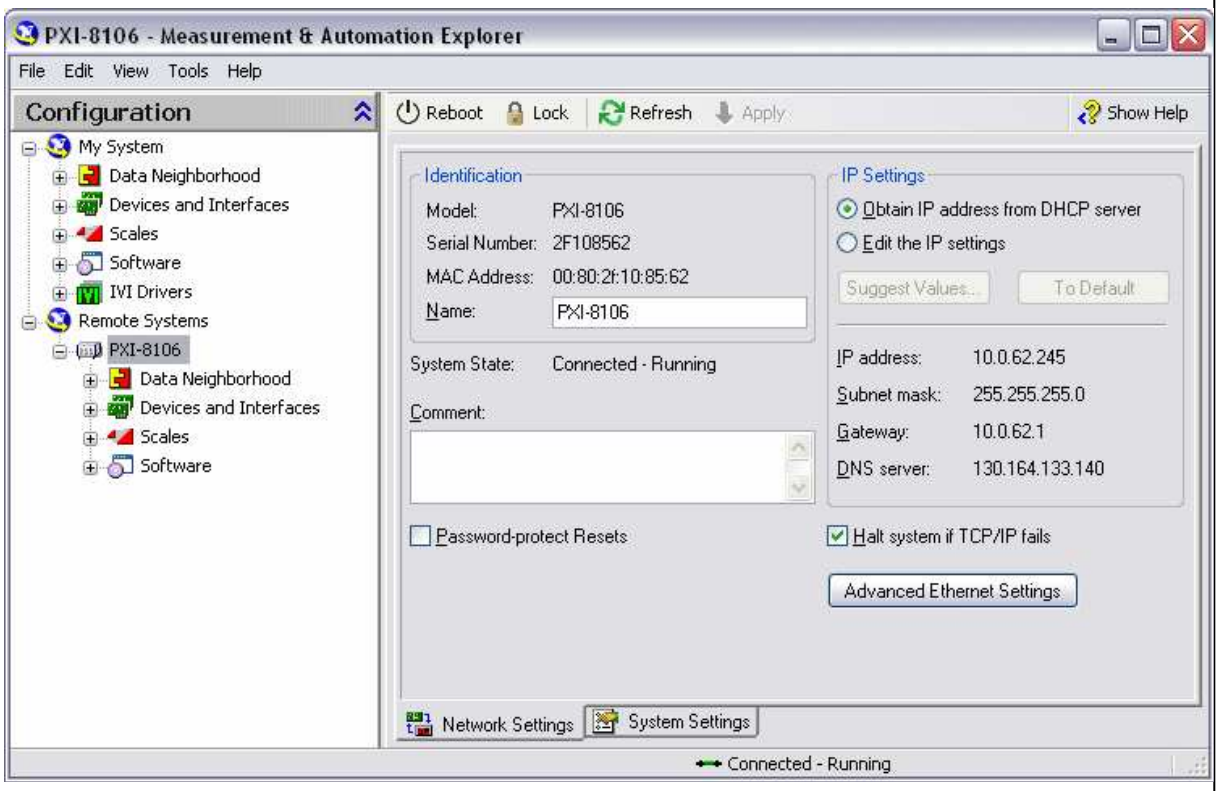
Musíme instalovat LabVIEW vývojový systém na hostující počítač dříve než se pokusíme instalovat RT modulový software a ovladače zařízení.

7.3.2 Konfigurace koncového RT hardware

Dříve než je možné nainstalovat software nebo ovladače na koncový RT hardware, je nutné použít MAX pro konfiguraci základního nastavení RT hardwaru.

Poznámka: Zasíťovaný RT hardware musíme připojit do stejné podsítě jako hostitelský počítač, ze kterého spouštíme MAX a nastavujeme počáteční konfigurace.

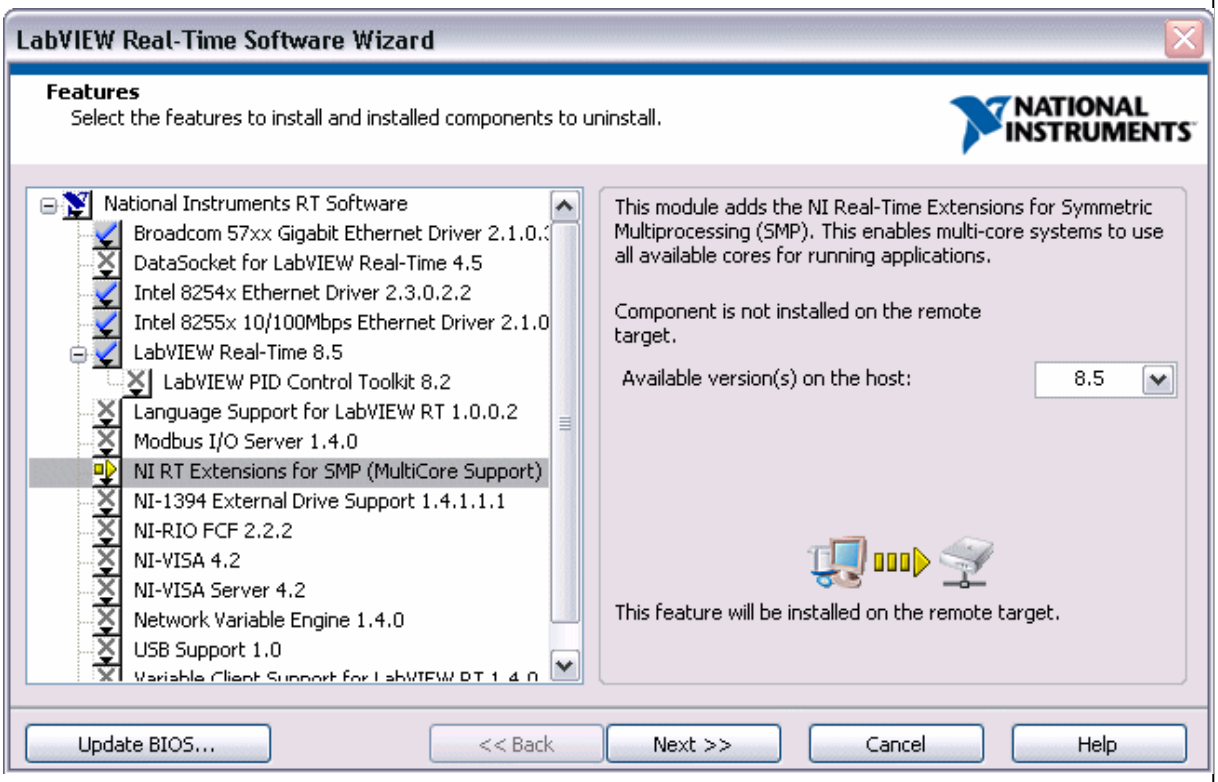
Obrázek 24 ukazuje RT Series PXI koncové zařízení, PXI-8106, konfigurované na automatické získávání IP adres z DHCP serveru, jak je ukázáno v IP Settings.



Obrázek 24: Konfigurace síťového nastavení RT hardwaru [1]

7.3.3 Instalace softwaru na koncový RT hardware

Použijte průvodce LabVIEW RT softwarem v MAXu pro instalaci softwaru a ovladačů z hostitelského počítače na RT koncové zařízení. V MAXu rozklikněte koncový hardware (target), klikněte pravým tlačítkem na Software pod koncovým zařízením a vyberte Add/Remove Software – Přidat/Odebrat software z menu. Tím se spustí průvodce LabVIEW RT software.



Obrázek 25: Instalace ovladačů a softwaru na koncový RT hardware [1]

Obrázek 25 ukazuje průvodce LabVIEW RT softwarem s LabVIEW RT modulovým softwarem vybraným pro instalaci na koncový RT hardware. Průzkumník LabVIEW RT softwarem zobrazuje všechny NI softwary a ovladače instalované na hostitelském počítači, které je možné nainstalovat na vybrané RT koncové zařízení.

7.3.3.1Instalace podpory mnohonásobného CPU

Aby bylo možné využít výhod, které poskytuje paralelní zpracování na mnohonásobných CPU systémech, používá se průvodce LabVIEW RT softwarem v MAX pro instalaci NI RT rozšíření pro SMP, jak je znázorněno na Obrázek 25. [1]

8. NÁVRH SYSTÉMU PRO SBĚR ANALOGOVÝCH DAT

Navržený jednoduchý systém pro sběr analogových dat, na kterém by šlo odzkoušet vlastnosti aplikace pracující v reálném čase se skládá z počítače, zásuvné PCI multifunkční karty a aplikace napsané v prostředí LabVIEW nebo LabWindows/CVI.

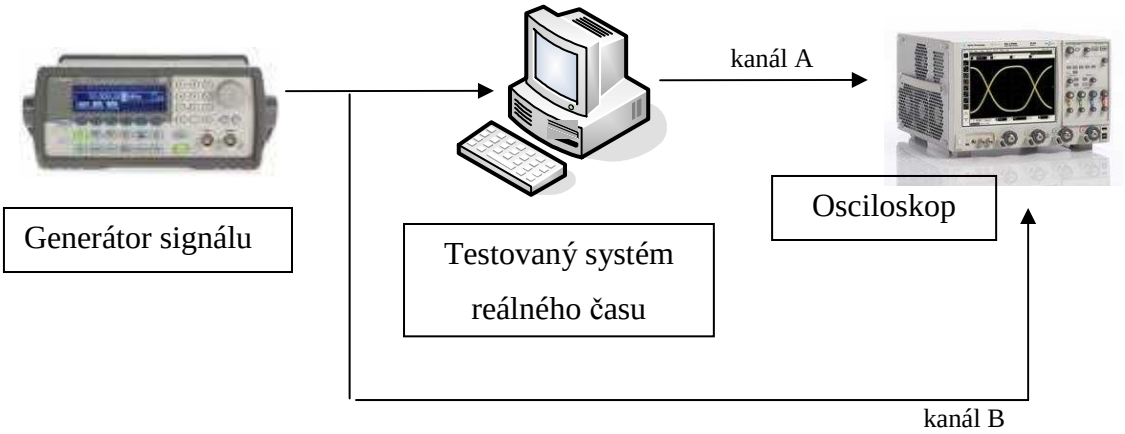
Naměřený analogový signál je na kartě zpracován – zesílen, navzorkován a převeden do digitální podoby pomocí A/D převodníku. Takto zpracovaný signál může být např. uložen do datových registrů nebo do paměti FIFO. Data jsou poté odeslána do počítače pomocí přerušení nebo pomocí DMA. Multifunkční karta je obvykle ještě vybavena čítačem a časovačem, který slouží např. k čítání impulsů.

Jako multifunční kartu lze použít např. NI X Series Multifunction Data Acquisition pro sběrnici PCI Express. *(model 6320, sběrnice PCI Express, počet analogových vstupů 16, max vzorkovací rychlost pro 1kanál 250 kS/s, maximální průchodnost 1 kanálu 250 kS/s, počet analogových výstupů 0, počet digitálních linek 24, Trigger – digitální)*

Obrázek 26: Multifunkční karta NI 6320

9. METODIKA SROVNÁNÍ OPERAČNÍCH YSTÉMŮ

Systém reálného času můžeme otestovat způsobem, kdy na vstup systému přivedeme vstupní signál (např. z generátoru) , který měříme osciloskopem a na výstupu systému měříme odezvu systému na druhém kanálu osciloskopu. Zapojení je schematicky naznačeno na následujícím obrázku.



;

Obrázek 27: Schéma zapojení pro ověření RT vlastností systému

Má-li tento systém provádět řízení nebo regulaci v reálném čase musí minimálně slpňovat:

- okamžitý přístup k hardwaru (I/O jednotce)
- přesný, rychlý a za všech okolností spolehlivý časovač, který zajistí opakované probuzení řídicí úlohy ve stanovený okamžik (vzorkovací perioda řízení)
- nepřerušitelnost celé periodické operace, stávající se ze získání regulované veličiny, výpočtu algoritmu regulace a aplikace akční veličiny

Všechny tyto požadavky jsou v operačním systému MS Windows realizovány pouze omezeně (viz kapitola 4.2)

K jednoduchému ověření chování v reálném čase při vstupně/výstupních operacích lze použít měření na vstupním portu multifunkční karty, na kterém je generován obdélníkový průběh (0 – 5V) se zvolenou frekvencí. Tento signál je poté přenášen na jiný výstupní port (napr. paralelní). Pokud dojde k většímu zatížení procesoru (např. při umělém zatížení procesoru) a frekvence na paralelním portu se nezmění, je to známkou správných real time vlastností systému. V opačném případě (změna frekvence nebo tvar výstupního signálu) nelze systém použít pro aplikace vyžadující deterministický přístup.

Čtení signálu z multifunkční karty lze vložit do nekonečné smyčky (Loop), která bude ukončena až s ukončením celé aplikace. Smyčku lze zpomalit pomocí funkce Wait (ms). Pro zápis hodnoty na paralelní port lze použít funkci OutPort.vi, která provádí fyzický zápis do výstupních registrů periférie PC. V nekonečné smyčce jsou tedy na port zapisovány hodnoty naměřené na vstupu. Naproti tomu v systému reálného času jsou funkce s časovou prodlevou nahrazeny deterministickými funkcemi pro časové prodlevy s definovanou dobou a prioritou (lze řešit např. Timed Loop a Wait Express.vi).

Oba testy by měly být provedeny v režimu bez zátěže CPU a se simulovanou zátěží CPU.

10. ZÁVĚR

Již více než 20 let vytváří grafický programovací jazyk LabVIEW společnosti National Instruments revoluci ve vývoji testovacích, měřicích a řídicích aplikací. I bez předchozích zkušeností mohou technici a vědci rychle a z hlediska nákladů efektivně používat měřicí a řídicí hardware, analyzovat data, sdílet výsledky a distribuovat systémy.

Stejných výsledků můžeme dosáhnout i s vývojovým prostředím v ANSI C LabWINDOWS/CVI. Nicméně programátor musí velmi dobře ovládat jazyk C a způsob programování ve Windows. Odměnou mu je potom větší volnost při psaní programového kódu.

Moduly pro programování v reálném čase jsou postaveny na stejných základech a i jejich konfigurování a ovládání je velmi podobné. Na základě firemní literatury lze tvrdit, že v obou vývojových prostředích lze dosáhnout velmi srovnatelných výsledků.

V závěru práce je navržen postup na ověření vlastností systému na RTOS a na operačním systému MS Windows XP. Z časových důvodů nebylo ověření vyzkoušenou na reálném systému. Nicméně na základě teoretické části této práce lze předpokládat, že systém na MS Windows bude mít horší deterministické vlastnosti než aplikace provozovaná na RTOS.

11. LITERATURA

[1] *Getting Started with the LabVIEW Real Time Module*, National Instruments™, 2009, 27s., 371375C-01

[2] *National Instruments.Getting Started with the LabWindowsc/CVI Real-Time Module*, National Instruments™, 2009, 30 s.

[3] *LabWindows/CVI Real-Time: Bringing Mission-Critical Reliability and Determinism to ANSI C Programmers*,
URL: <http://zone.ni.com/devzone/cda/tut/p/id/3591> [cit. 2009-12-10].

[4] *Using Desktop PCs as RT Targets with the Real-Time Module*, National Instruments™, 2009, 8s., 371857B-01

[5] ZEZULKA, F. *Prostředky průmyslové automatizace*, Brno, 161s., skripta VUT v Brně

[6] MĚŠŤAN R., *Doporučení pro volbu „real-time“ operačního systému pro 8-bitové mikrořadiče*, Brno, 2006, 32 s., Bakalářská práce na FSI VUT v Brně na Ústavu automatizace a měřicí techniky. Vedoucí bakalářské práce Ing. Pavel Houška, Ph.D.

[7] FOJTÍK D., *Realizace řízení v reálném čase pod Microsoft Windows 2000/XP*, AUTOMA č.11 (2004), Praha, FCC Public s.r.o.

[8] ŠUMBERA P., *Porovnávací test operačních systémů reálného času*, AUTOMA č. 08 (2000), Praha, FCC Public s.r.o.

[9] ČERNOHORSKÝ J., *Základní koncepce operačních systémů pracujících v reálném čase (RTOS)*, AT&T journal 12/2005, Bratislava, HMH s.r.o.

[10] HRDINA F., *Návrh programového rozhraní pro řízení mikropolohovací platformy z PC v reálném čase*, Praha, 2007, 63 s., Bakalářská práce na Fakultě elektrotechnické ČVUT v Praze. Vedoucí bakalářské práce Ing. Ondřej Holub

[11] ZEZULKA, F., *Průmyslový ethernet IV: Principy průmyslového ethernetu*, AUTOMA č.10 (2007), Praha, FCC Public s.r.o.