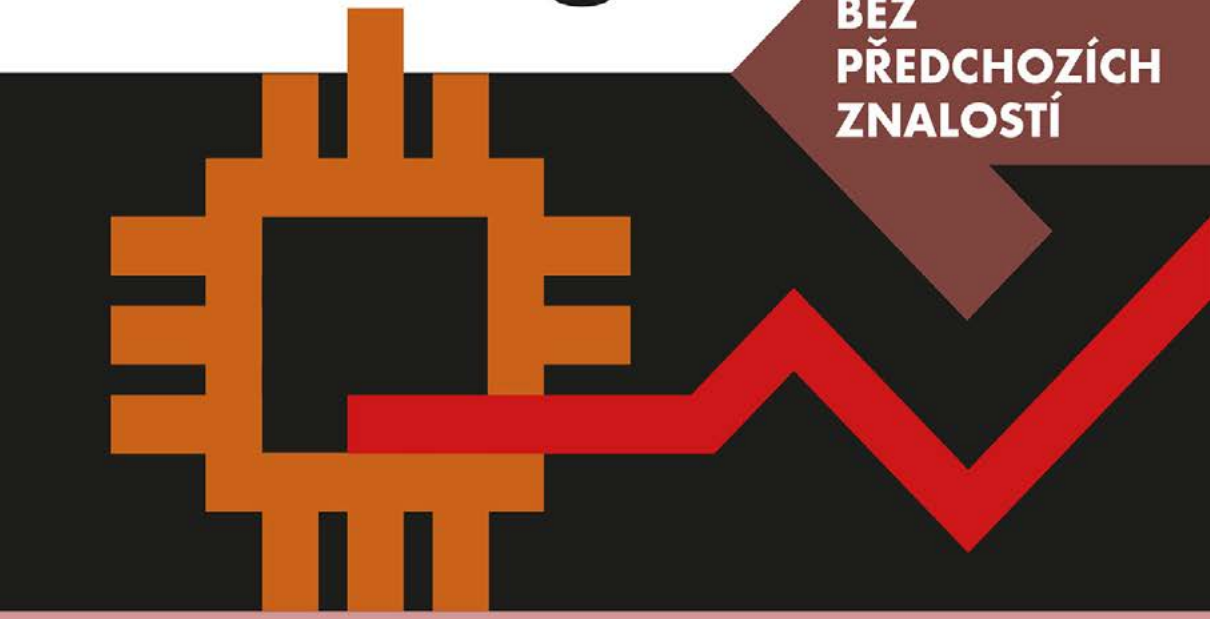
David Matoušek

PROGRAMOVÁNÍ MIKROKONTROLÉRŮ ATmega

BEZ PŘEDCHOZÍCH ZNALOSTÍ



Datové typy, operátory, řídicí struktury

Funkce, pole, ukazatele, práce s řetězci

Přerušení, čítače, časovače

Sériové komunikační jednotky USART, SPI a TWI

computer press

Programování mikrokontrolérů ATmega bez předchozích znalostí

Vyšlo také v tištěné verzi

Objednat můžete na
www.computerpress.cz
www.albatrosmedia.cz

David Matoušek

Programování mikrokontrolérů
ATmega bez předchozích znalostí – e-kniha

ALBATROS MEDIA

Programování mikrokontrolérů ATmega bez předchozích znalostí

Programování mikrokontrolérů ATmega bez předchozích znalostí

David Matoušek

computer press

ISBN tištěné verze 978-80-251-5042-9
ISBN e-knihy 978-80-251-5044-3 (1. zveřejnění, 2022) (ePDF)

Obsah

Předmluva

Kniha, kterou právě držíte v ruce, je určena zájemcům, kteří se chtějí naučit programovat mikrokontroléry rodiny ATmega pomocí jazyka C. Kniha provádí čtenáře od začátků programování až k pokročilým záležitostem, jako je použití čítačů/časovačů a sériových komunikačních sběrnic. Součástí postupného výkladu jednotlivých částí mikrokontroléru je i výklad jazyka C. To jistě ocení čtenáři, kteří s tímto programovacím jazykem nemají předchozí zkušenosti. Text je doplněn čtyřmi desítkami příkladů.

Prakticky se kniha zaměřuje na mikrokontrolér **ATmega328PB** s použitím vývojového kitu **ATmega328PB XMINI** (v ceně okolo 300 korun) a s vývojovým prostředím **Microchip Studio**. Prostředí je k dispozici zdarma.

V úvodní kapitole jsou vysvětleny základní pojmy ze světa mikrokontrolérů a je stručně popsán vývojový kit, rovněž je krok za krokem vysvětlena instalace vývojového prostředí Microchip Studio a nakonec je sestaven první program.

Ve druhé kapitole jsou nejdříve popsány datové typy pro práci s celočíselnými a reálnými proměnnými. Je vysvětleno použití funkce printf, která slouží pro formátovaný výpis proměnných. Také jsou vysvětleny a předvedeny základní matematické operace. V závěru je řešena problematika typové konverze a jsou popsány výhody používání konstant.

Třetí kapitola vysvětluje používání vývojových diagramů, funkci relačních a logických operátorů včetně sestavování podmínek, použití podmíněného příkazu if a cyklu typu while.

Čtvrtá kapitola se věnuje popisu funkce a praktickému použití digitálních vstupů/výstupů. Jsou vysvětleny bitové operátory, které umožňují přímé řízení jednotlivých vývodů portů mikrokontroléru, a tato znalost je pak zužitkována na řízení LED a čtení stavu tlačítka.

V páté kapitole se seznámíme s výhodami používání funkcí pro zpřehlednění a zkrácení zápisu programu. Jsou ukázány funkce předávající parametry hodnotou, ale i odkazem. Vysvětleny jsou též dopředná deklarace funkce a inline funkce. Závěr kapitoly představuje použití funkcí pro přehledné řízení dvoumístného 7segmentového displeje.

Šestá kapitola doplňuje možnosti programových konstrukcí o cykly do..while a for a uvádí též podmíněný příkaz switch. Tyto nové programové konstrukce jsou předvedeny formou příkladů.

Sedmá kapitola probírá použití polí a ukazatelů. Pole jsou vhodná pro hromadné zpracování dat (každý prvek pole má přiděleno pořadové číslo, takže průchod polem je možný pomocí cyklu). Ukazatele jsou používány pro předávání proměnných do funkcí a mají souvislost s poli.

Osmá kapitola vysvětluje pojem přerušení a obsahuje popis vstupů vnějšího přerušení. K tomu jsou připojeny dva příklady programového vytvoření PWM signálu, kdy je střída měněna buď obsluhou přerušení uživatelského tlačítka, nebo rotačního enkodéru.

Devátá kapitola je věnována výkladu funkce čítačů/časovačů. Jednotlivé příklady ukazují hardwarové generování PWM, použití periodického přerušení pro obsluhu LED a 2místného displeje, měření kmitočtu a časových intervalů.

Desátá kapitola popisuje funkci zabudovaného A/D převodníku. Pro praktické použití A/D převodníku je též vysvětlen datový typ výčet, který slouží pro rozlišení jednotlivých variant referenčního napětí a analogových kanálů. Jednotlivé příklady ukazují měření interního zdroje referenčního napětí, interního teplotního senzoru a externího napětí.

Jedenáctá kapitola ukazuje použití zabudovaného analogového komparátoru pro měření odporu potenciometru. Uvedený příklad lze snadno upravit na měření teploty nebo osvětlení (náhradou potenciometru za termistor nebo fotorezistor).

Kapitola 12 uvádí informace k datovým typům typu char a char*, které se v jazyce C používají pro reprezentaci znaků a řetězců. Jsou vysvětleny a pomocí krátkého příkladu také předvedeny základní operace se znaky a řetězci proto, aby mohly být použity v dalších kapitolách knihy.

Kapitola 13 popisuje jednotky USART, které zajišťují synchronní nebo asynchronní sériovou komunikaci. Obvykle je pomocí jednotek USART připojováno nějaké externí zařízení. Na příkladech je ukázáno použití jednotky USART0 pro odesílání a příjem znaků sériovým terminálem.

Kapitola 14 popisuje jednotky SPI, které zajišťují vysokorychlostní synchronní sériovou komunikaci. Obvykle jsou pomocí jednotek SPI připojovány specializované integrované obvody pro rozšíření schopností mikrokontroléru (například externí D/A převodníky, rychlé externí A/D převodníky nebo expandéry portů). Konkrétně je předvedeno použití expandéru typu MCP23S08 pro čtení stavu 4 digitálních vstupů a řízení 4 digitálních výstupů.

Kapitola 15 popisuje jednotky TWI, které vytvářejí synchronní sériovou sběrnici typu I^2C. Obvykle jsou pomocí sběrnice I^2C připojovány specializované integrované obvody pro rozšíření schopností mikrokontroléru. Konkrétně je předvedeno použití expandéru typu MCP23008 pro čtení stavu 4 digitálních vstupů a řízení 4 digitálních výstupů.

Šestnáctá kapitola je věnována výkladu zbývajících jednotek a subsytémů mikrokontroléru ATmega328PB. Jsou jimi hodinový subsytém, režimy snížené spotřeby (sleep modes), dohlížecí obvod WDT (watchdog), paměti typu E^2PROM a Flash, jednotky OCM a PTC, čítač/časovač 2 (může pracovat jako hodiny reálného času RTC)

V závěru jsou zařazeny dvě přílohy. Příloha A popisuje výrobu přídavné desky TEST328PB. Příloha B popisuje výrobu desky EXPANDER.

Seznam příkladů uvedených v knize:

PROG_01: první program (test vývojového kitu a sériové komunikace),

PROG_02: deklarace proměnných a jejich formátovaný výpis,

PROG_03: binární operátory pro celá a reálná čísla,

PROG_04: operátory inkrementace a dekrementace,
PROG_05: použití větvení pro hledaní maxima dvou čísel,
PROG_06: použití větvení pro omezení hodnoty na určený rozsah,
PROG_07: výpis čísel 1 až 10 pomocí cyklu while,
PROG_08: blikání LED,
PROG_09: ovládání digitálních výstupů pomocí bitových operátorů,
PROG_10: použití digitálních vstupů pro řízení programu,
PROG_11: zpřehlednění zápisu programu PROG_10,
PROG_12: univerzální funkce pro manipulaci s bity registrů mikrokontroléru,
PROG_13: použití inline funkcí a dopředné deklarace funkcí,
PROG_14: skupina funkcí pro ovládání displeje,
PROG_15: použití cyklu for a podmíněného příkazu switch pro ovládání displeje,
PROG_16: dekódování číslic displeje s použitím logických a bitových operátorů,
PROG_17: ovládání 7segmentového displeje s použitím pole konstant,
PROG_18: ukázka funkce, která předává parametry přes ukazatel,
PROG_19: ukázka funkcí, která pracují s poli celých čísel,
PROG_20: řízení PWM signálu pomocí vstupu vnějšího přerušení,
PROG_21: řízení PWM signálu pomocí rotačního enkodéru,
PROG_22: regulace jasu LED pomocí čítače/časovače 4 v režimu PWM,
PROG_23: blikání LED pomocí přerušení čítače/časovače 4 (normální režim),
PROG_24: obsluha dynamicky řízeného displeje čítačem/časovačem 4 (CTC režim),
PROG_25: měření kmitočtu čítačem/časovačem 3 přímou metodou,
PROG_26: měření periody a šířky impulzu pomocí čítače/časovače 1,
PROG_27: programově spouštěné měření vnitřních napětí mikrokontroléru,
PROG_28: spouštění A/D převodů záchytným registrem čítače/časovače 1,
PROG_29: měření odporu analogovým komparátorem a záchytným registrem,
PROG_30: ukázka práce se řetězci a znaky,
PROG_31: odeslání řetězce blokujícím způsobem,
PROG_32: odeslání a příjem řetězce přes přerušení,
PROG_33: vzájemná komunikace mezi jednotkami SPI0 a SPI1,
PROG_34: použití obvodu MCP23S08 pro obsluhu digitálních vstupů a výstupů,
PROG_35: použití obvodu MCP23008 pro obsluhu digitálních vstupů a výstupů,

PROG_36: programové přepínání hodinového kmitočtu mikrokontroléru,

PROG_37: příklad použití režimu Idle,

PROG_38: příklad použití WDT přerušení,

PROG_39: příklad použití E^2PROM,

PROG_40: příklad použití Flash pro uložení konstant.

Zdrojové kódy ke knize

Ze stránky knihy na *www.albatrosmedia.cz* si v sekci Ke stažení můžete přímo stáhnout archiv s ukázkovými kódy.

Errata

Přestože jsme udělali maximum pro to, abychom zajistili přesnost a správnost obsahu, chybám se úplně vyhnout nedá. Pokud v některé z našich knih najdete chybu, ať už chybu v textu nebo v kódu, budeme rádi, pokud nám ji nahlásíte. Ostatní uživatele tak můžete ušetřit frustrace a pomoci nám zlepšit následující vydání této knihy.

Veškerá existující errata najdete na stránce knihy na *www.albatrosmedia.cz* v sekci Ke stažení. (Nejsou-li žádná errata zatím k dispozici, není příslušný soubor dostupný.)

KAPITOLA 1

Úvod do programování mikrokontrolérů

V této kapitole:

- Základní pojmy
- Seznámení s vývojovým kitem ATmega328PB Xplained Mini
- Vývojové prostředí Microchip Studio
- První program

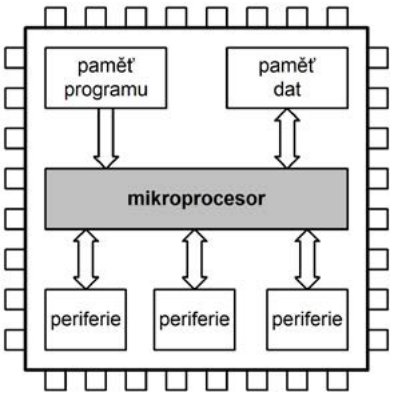
Mikrokontrolér (podle české terminologie správněji *mikrořadič*) je integrovaný obvod, který představuje „malý počítač". Jako historicky první mikrokontrolér se obvykle označuje typ Intel 8048, který pochází z 80. let minulého století. Obrovskými výhodami mikrokontrolérů jsou jejich vysoká spolehlivost a skutečnost, že jediný integrovaný obvod může po svém naprogramování provádět prakticky libovolnou činnost.

Mikrokontroléry nejsou na rozdíl od klasických stolních nebo přenosných počítačů vybaveny klávesnicí, myší a obrazovkou (připojení těchto zařízení k mikrokontroléru je však možné pomocí speciálních jednotek, které jsou v některých mikrokontrolérech dokonce zaintegrovány). Základní vstupně/výstupní zařízení mikrokontrolérů jsou například tlačítka, dotykové klávesnice nebo LED (svítivé diody) a displeje typu LCD. Mikrokontroléry jsou především vybaveny jednotkami pro měření elektrického napětí, odměr časových intervalů, vytváření různých signálů a komunikaci s okolními systémy pomocí sériových a paralelních rozhraní.

Výpočetní výkon mikrokontroléru souvisí s jeho schopností provádět matematické operace s čísly. Podle výpočetního výkonu tak rozlišujeme 8bitové, 16bitové, 32bitové a 64bitové mikrokontroléry. 64bitové a 32bitové mikrokontroléry dosahují výpočetních výkonů, které jsou srovnatelné s výkonem klasických stolních nebo přenosných počítačů, a nasazovány jsou především do mobilních telefonů. 8bitové a 16bitové mikrokontroléry jsou používány především v oblasti měření, řízení a regulace. Běžně je „potkáváme" ve spotřební elektronice.

Základní pojmy

Mikrokontrolér obsahuje řídicí jednotku nazývanou **mikroprocesor** (též centrální procesorová jednotka CPU), paměť programu, paměť dat a periferie.



Obrázek 1.1 Zjednodušené blokové schéma mikrokontroléru

Paměť programu obsahuje pokyny, které má mikroprocesor vykonat. Všechny tyto pokyny dohromady tvoří program, který mikrokontrolér vykonává. Paměť programu je obvykle realizována jako nonvolatilní paměť typu Flash (je to stejný typ paměti jako u Flash disků). Označení nonvolatilní znamená, že údaje obsažené v této paměti zůstanou zachovány i po odpojení napájecího napětí. Program tedy zůstane uložen beze změn až do chvíle, než mikrokontrolér přeprogramujeme.

Datová paměť slouží pro uložení výsledků běhu programu. Tato paměť je volatilní, tedy údaje jsou uchovány pouze po dobu připojení napájecího napětí. Jakmile je napájecí napětí odpojeno, údaje jsou ztraceny.

Periferie provádí komunikaci s okolím. Pomocí periferií může mikrokontrolér reagovat na pokyny uživatele nebo zobrazovat výsledky. Označení periferie je dáno historicky, protože v prvních počítačích byly tyto jednotky umístěny mimo integrovaný obvod mikroprocesoru. Současné mikrokontroléry jsou vybaveny desítkami periférií rozličných typů.

Zakoupený mikrokontrolér nemá určený žádný program. Teprve vlastním programem určíme, jakou činnost má mikrokontrolér provádět. Program vytvoříme pomocí vývojového prostředí.

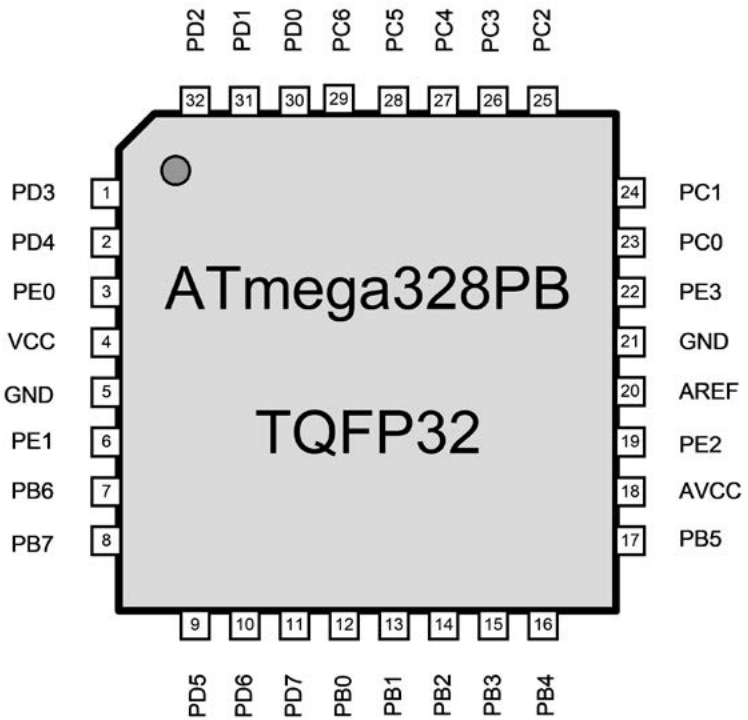
Vývojové prostředí je speciální počítačová aplikace, kterou spustíme na normálním („velkém“) počítači. Vývojové prostředí nám umožní zapsat zdrojový text programu, provést jeho překlad do podoby potřebné pro mikrokontrolér a pomocí programátoru nahrát program do programové paměti mikrokontroléru.

Vývojový kit je deska plošných spojů, na které se nachází samotný mikrokontrolér. Obsahuje také programátor, základní vstupně/výstupní zařízení (alespoň jednu LED a jedno tlačítko) a konektory pro připojení rozšiřujících desek. Připojení vývojového kitu k počítači je obvykle řešeno pomocí USB. Vývojový kit tak umožňuje rychlý úvodní krok do programování mikrokontrolérů. I úplný začátečník může okamžitě pracovat s daným mikrokontrolérem a nepotřebuje k tomu žádné další speciální vybavení. Vývojové kity jsou v současnosti poměrně levné a lze je pořídit v cenách od 300 korun (v některých případech i levněji).

Seznámení s vývojovým kitem ATmega328PB Xplained Mini

V této knize se zaměříme na mikrokontrolér s označením **ATmega328PB**, který je vyráběn v pouzdrech typu TQFP nebo VQFN pro tzv. povrchovou montáž (SMD). Výhodou jsou velmi malé rozměry: 7 × 7 mm nebo 5 × 5 mm.

Je to sice „jen" 8bitový mikrokontrolér, avšak má taktovací kmitočet až 20 MHz a je vybaven nadprůměrným počtem výkonných periferií. Celkově má 32 vývodů, viz obr. 1.2. Na tomto místě nebudeme podrobně popisovat jednotlivé vývody mikrokontroléru, tento popis bude uváděn postupně tak, jak se budeme seznamovat s jednotlivými jednotkami mikrokontroléru. Za povšimnutí nyní stojí pouze vývody PB5 a PB7, které jsou použity pro připojení LED a tlačítka vývojového kitu (viz níže).

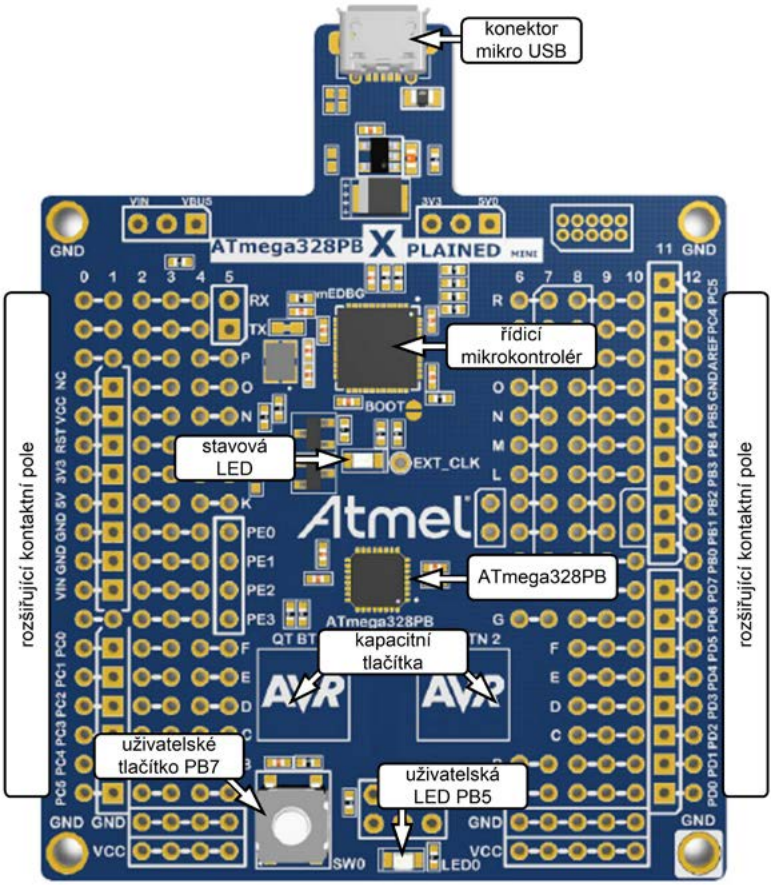


Obrázek 1.2 Pouzdro mikrokontroléru ATmega328PB typu TFQP32 (zvětšeno), překresleno podle [1]

Pro snadné použití mikrokontroléru bez nutnosti předchozích příprav (jako je výroba desky plošných spojů a její pracné osazovaní) budeme používat vývojový kit **ATmega328PB Xplained Mini** pod obchodním označením **ATMEGA328PB-XMINI**. Prodejní cena tohoto vývojového kitu se pohybuje okolo 300 korun.

Nejvýznamnější části vývojového kitu **ATMEGA328PB-XMINI** jsou (viz obr. 1.3):

- Samotný mikrokontrolér ATmega328PB.
- Konektor typu USB mikro B, kterým vývojový kit připojíme k počítači (slouží pro napájení a řízení vývojového kitu).
- Řídicí mikrokontrolér, který zajišťuje programování cílového mikrokontroléru.
- Virtuální sériový port pro komunikaci mezi mikrokontrolérem a počítačem.
- Stavová LED, která indikuje aktuální režim, v němž se vývojový kit nachází.
- Rozšiřující kontaktní pole, které umožňuje připojovat další jednotky k jednotlivým vývodům mikrokontroléru.
- Dvě kapacitní tlačítka.
- Uživatelské tlačítko připojené na vývod mikrokontroléru **PB7**.
- Uživatelská LED připojená na vývod mikrokontroléru **PB5**.

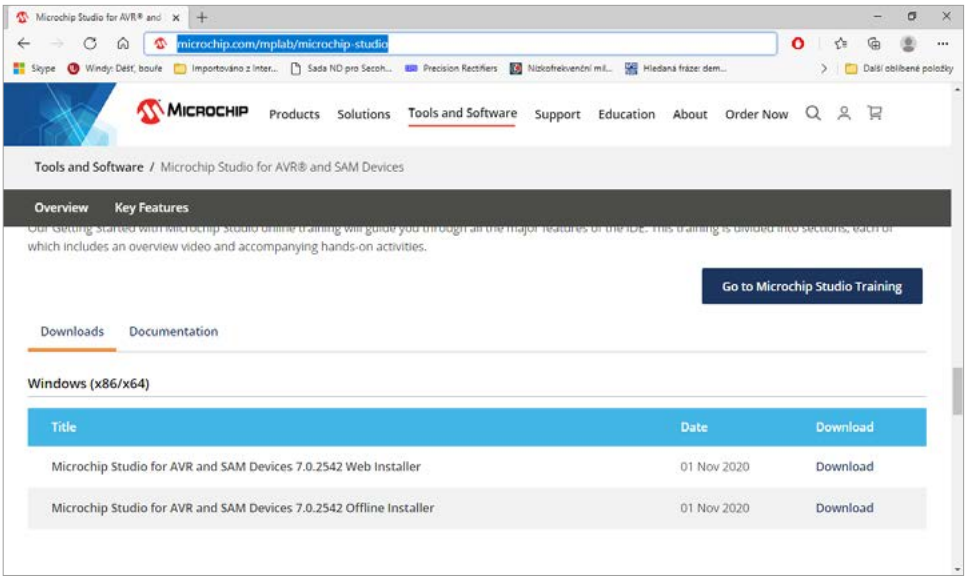


Obrázek 1.3 Vývojový kit ATmega328PB XMINI, překresleno s použitím [2]

Označení „uživatelské“ znamená, že dané tlačítko a LED můžeme libovolně používat v programech, které budeme vytvářet. Například můžeme číst stav tlačítka (zmáčknuto nebo nezmáčknuto) a podle toho měnit chování programu. LED můžeme rozsvítit nebo zhasnout a tím indikovat různé stavy programu.

Vývojové prostředí Microchip Studio

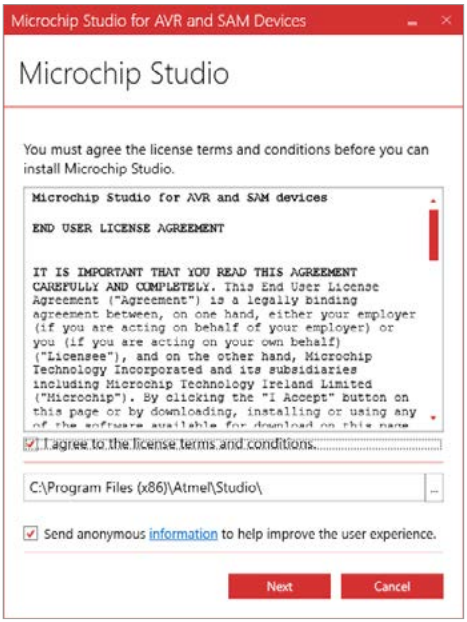
Pro zápis zdrojových textů programů pro mikrokontrolér ATmega328PB, jejich následný překlad a naprogramování mikrokontroléru používáme vývojového prostředí **Microchip Studio** (dříve Atmel Studio). Toto vývojové prostředí je možné stáhnou zcela zdarma ze stránek *https://www.microchip.com/mplab/microchip-studio* (viz obr. 1.4):



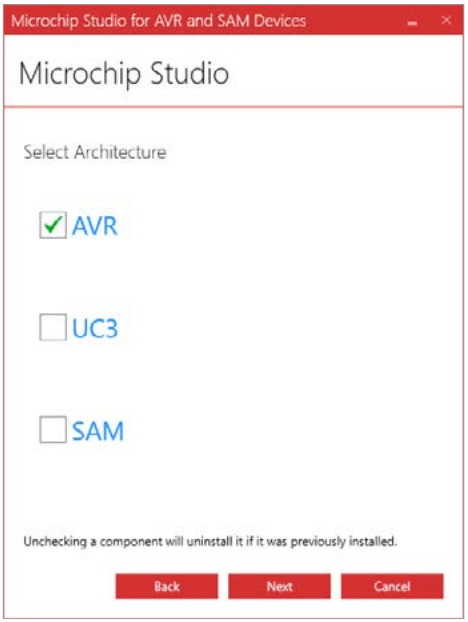
Obrázek 1.4 Výběr instalačního souboru na stránkách www.microchip.com

Po stažení instalačního souboru pomocí **Download** se zobrazí instalační dialog dle obr. 1.5. Potvrdíme licenční podmínky a pokračujeme stiskem tlačítka **Next**.

Následně se zobrazí dialog dle obr. 1.6. V něm zaškrtnutím příslušného políčka označíme pouze rodinu mikrokontrolérů **AVR** (UC3 a SAM jsou další rodiny mikrokontrolérů, které však nebudeme aktuálně používat, pro zkrácení instalace je tedy nevybereme). Poté pokračujeme stiskem tlačítka **Next**.



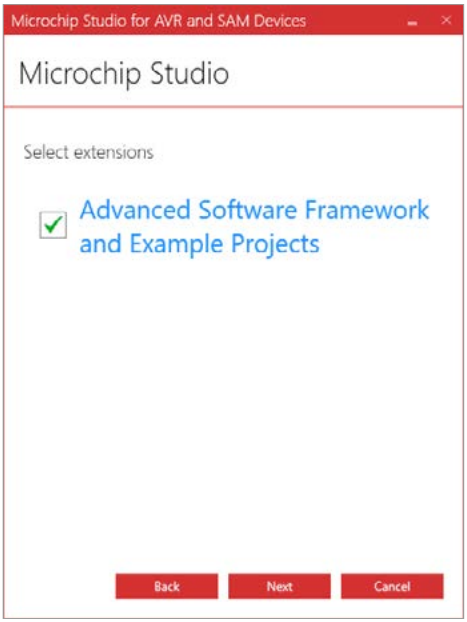
Obrázek 1.5 Potvrzení licenčních podmínek



Obrázek 1.6 Výběr rodiny mikrokontrolérů pro instalaci vývojového prostředí

Následně se objeví dialog dle obr. 1.7. Tam je třeba zaškrtnout políčko **Advanced Software Framework and Example Project**. Tato volba zajistí, že se budou instalovat soubory knihoven (tzv. framework) a vzorové projekty.

Následně jsou ještě zkontrolovány systémové požadavky a rozbíhá se instalace, která trvá dle výkonosti použitého počítače zhruba 5 až 10 minut. Viz obr. 1.8.

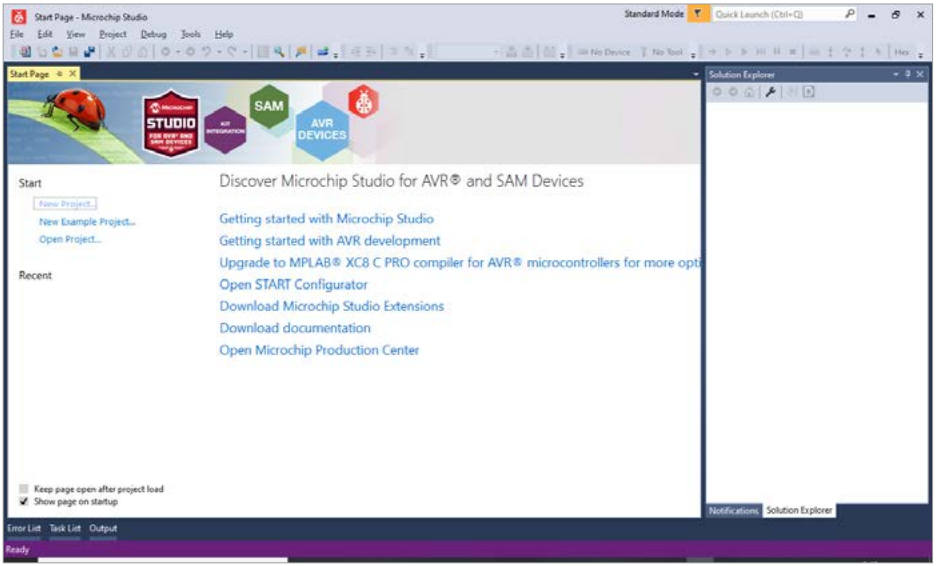


Obrázek 1.7 Potvrzení instalace knihoven a vzorových příkladů

Obrázek 1.8 Vlastní instalace

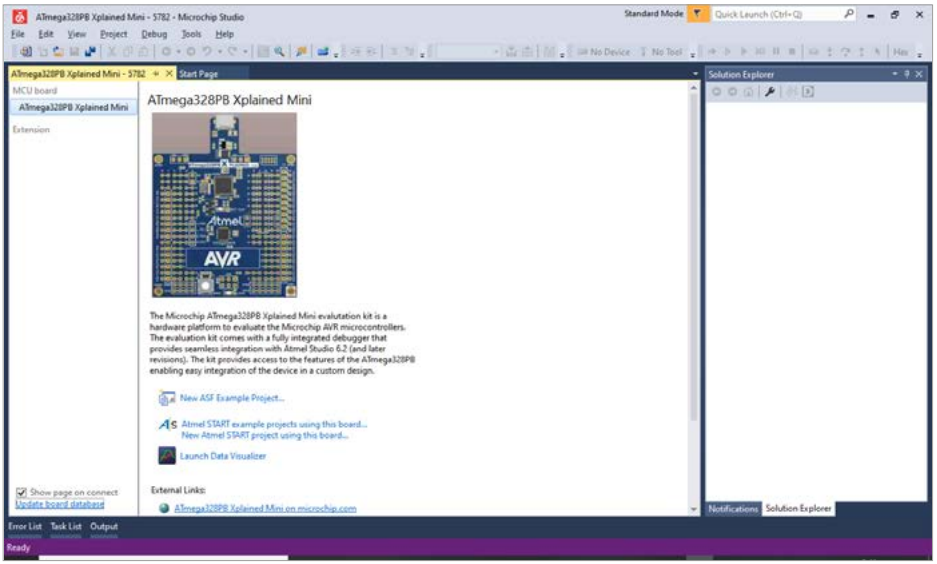
V průběhu instalace se může objevit další dialog s dotazem, zda mají být instalovány ovladače pro vývojové nástroje. Tento požadavek musíme potvrdit, jinak by nastal problém s používáním vývojového kitu, který tyto ovladače potřebuje.

Nakonec budeme informováni o úspěšném dokončení instalace a vývojové prostředí se následně spustí. Při prvním spuštění může být položen dotaz, zda budeme používat standardní, nebo rozšířenou variantu uživatelského profilu. Doporučujeme zvolit variantu **Advanced**. Situaci po prvním spuštění vývojového prostředí ukazuje obr. 1.9.



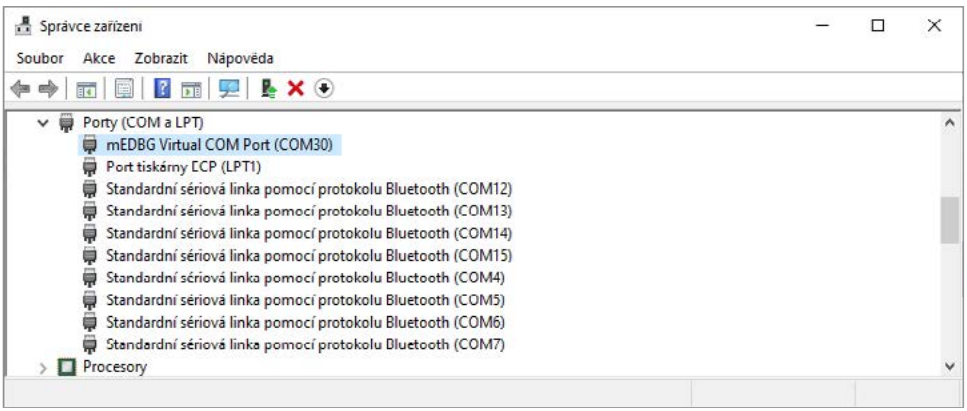
Obrázek 1.9 Vývojové prostředí Microchip Studio po prvním spuštění

Nyní je vhodné připojit vývojový kit **ATMEGA328PB-XMINI** USB kabelem k počítači. Úspěšné připojení bude ve vývojovém prostředí indikováno zobrazením další záložky, která ve svém záhlaví obsahuje poslední čtyřčíslí sériového výrobního čísla vývojového kitu, a na ploše této záložky je vyobrazen vývojový kit. Viz obr. 1.10.



Obrázek 1.10 Rozpoznání vývojového kitu ve vývojovém prostředí

Pro otestování vývojového kitu spustíme **Správce zařízení**. V jeho okně hledáme skupinu **Porty (COM a LPT)** a položku **mEDBG Virtual COM Port** (viz obr. 1.11). Označení v závorce odpovídá virtuálnímu sériovému portu, který poskytuje vývojový kit pro sériovou komunikaci mezi mikrokontrolérem a počítačem (na obr. 1.11 je to COM30, ale označení bude v praktickém případě odlišné).

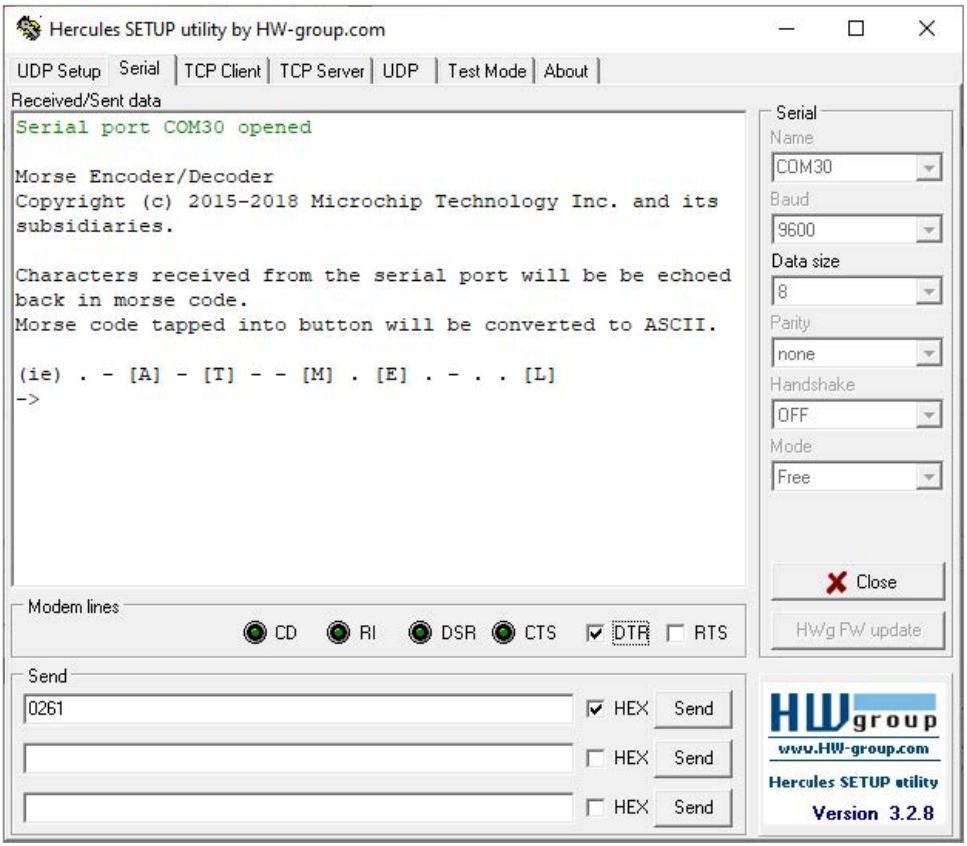


Obrázek 1.11 Zjištění čísla virtuálního sériového portu pro komunikaci mezi mikrokontrolérem a počítačem

Pro sledování sériové komunikace na straně počítače potřebujeme tzv. sériový terminál. Pro tyto účely použijeme program **Hercules**, který je k dispozici zcela zdarma na stránkách *https://www.hw-group.com/cs/software/aplikace-hercules-setup*.

Tento program lze přímo spustit, není nutná jeho instalace. Po spuštění vybereme záložku **Serial**. Poté nastavíme **Name** (název sériového portu, například COM30), **Baud** (přenosová rychlost, ponecháme výchozí hodnotu 9600 Bd), **Data size** (velikost přenášených dat, ponecháme výchozí hodnotu 8 bitů), **Parity** (typ parity používaný pro zabezpečení přenosu, ponecháme výchozí hodnotu none), **Handsake** (řízení toku dat, ponecháme výchozí hodnotu OFF), **Mode** (ponecháme výchozí hodnotu Free). Nakonec zmáčkneme tlačítko **Open** a poté zaškrtneme políčko **DTR**.

Poté již sériový terminál vypisuje znaky, které vysílá mikrokontrolér virtuálním sériovým portem do počítače. Vývojový kit je totiž z výroby nastaven tak, že tuto sériovou komunikaci provádí program v mikrokontroléru. Výsledek dle obr. 1.12 se může poněkud lišit od výpisu, který budete pozorovat.



Obrázek 1.12 Sériový terminál vypisuje znaky přijaté z mikrokontroléru.

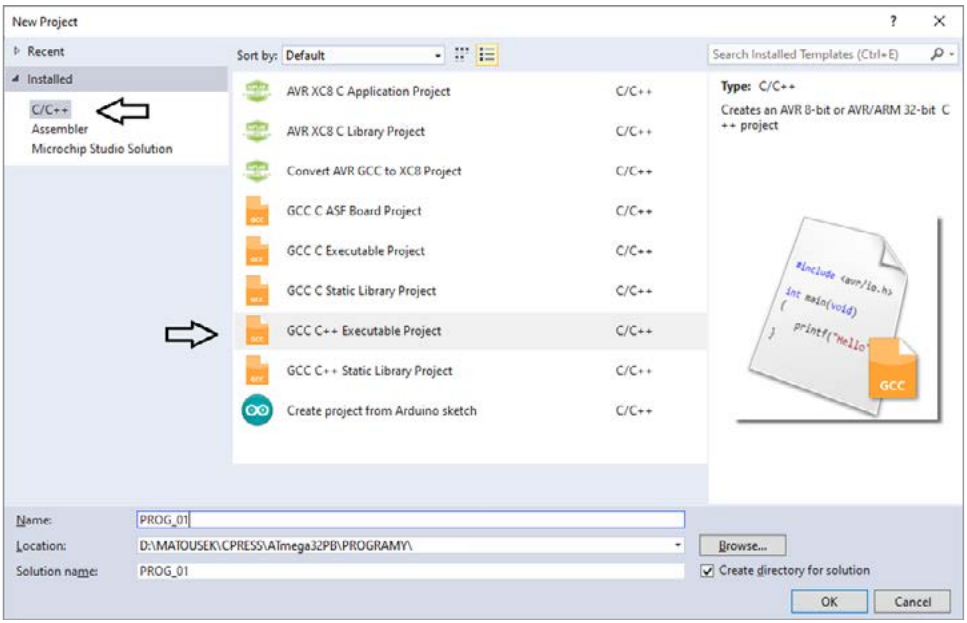
Instalace vývojového prostředí Microchip Studio a stažení programu Hercules představuje všechny softwarového nástroje, které budeme v této knize používat pro vývoj jednotlivých programů mikrokontroléru ATmega328PB.

Nyní tedy můžeme přistoupit k vytvoření ukázkového programu.

První program

Pro každý program, který budeme vytvářet, musíme nejdříve založit tzv. projekt. **Projekt** je skupina souborů a příslušných nastavení, která jsou nutná pro úspěšný překlad zdrojového souboru.

Ve vývojovém prostředí **Microchip Studio** proto vyhledáme položku nabídky **File → New → Project**. Následně se zobrazí dialog dle obr. 1.13. Pro vytvoření projektu v jazyce C ve vývojovém prostředí **Microchip Studio** vybereme variantu **C/C++** a poté **GCC C++ Executable Project**.

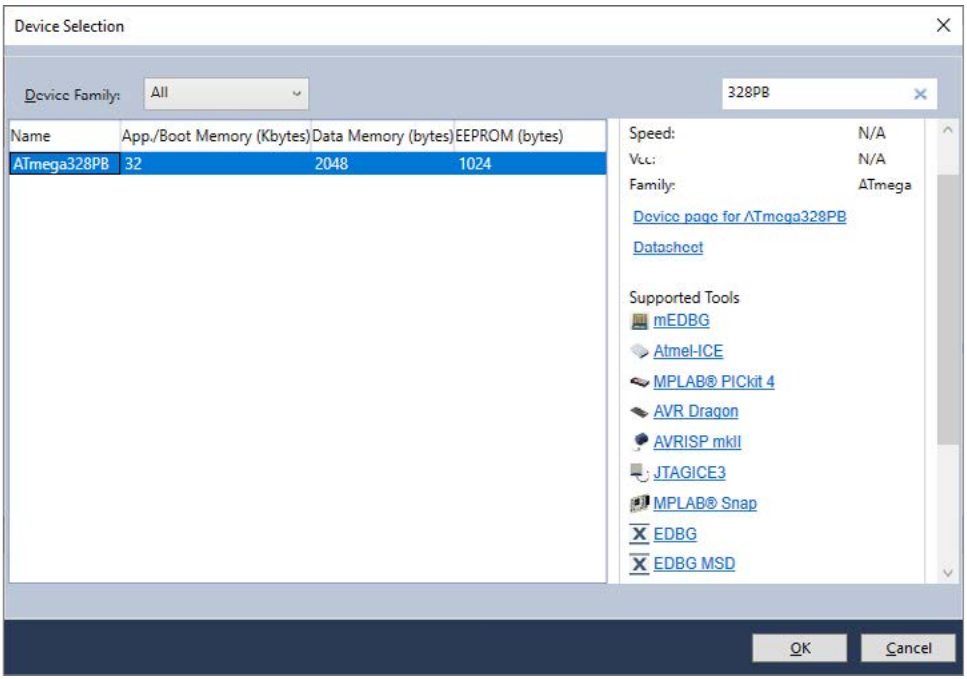


Obrázek 1.13 Dialog pro založení nového projektu

Poznámka: Variantu C++ používáme pro usnadnění některých operací, kód však bude zapsán na úrovni jazyka C.

Je také nutné zadat složku pro ukládání souborů, výběr složky provedeme pomocí tlačítka **Browse** (umístěno dole). Poté zadáme název projektu v poli **Name** (zde **PROG_01**) a zaškrtneme políčko **Create directory for solution** (tím je zařízeno, že se soubory projektu budou umísťovat do zvláštní složky).

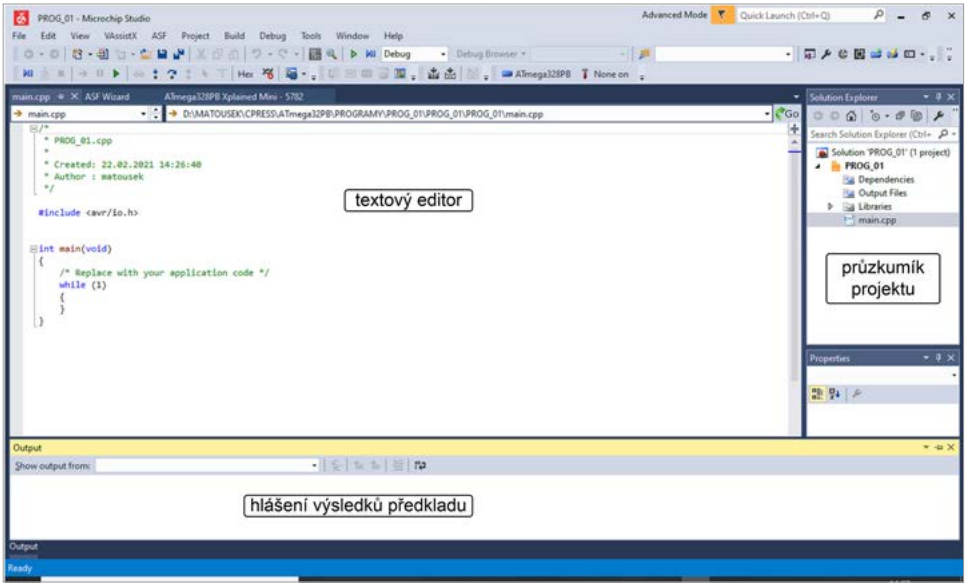
Po provedení všech nastavení dle obr. 1.13 pokračujeme stiskem tlačítka **OK**.



Obrázek 1.14 Výběr typu cílového mikrokontroléru

V následně zobrazeném dialogu dle obr. 1.14 vybereme mikrokontrolér **ATmega328PB** (stačí do editačního pole vpravo nahoře zapsat část označení mikrokontroléru, například zapíšeme pouze 328PB a výpis dostupných typů se zúží). Vytváření projektu dokončíme tlačítkem **OK**. Výběr správného typu mikrokontroléru je nezbytný, musí být přesně vybráno ATmega328PB, jinak bude při programování hlášena chyba.

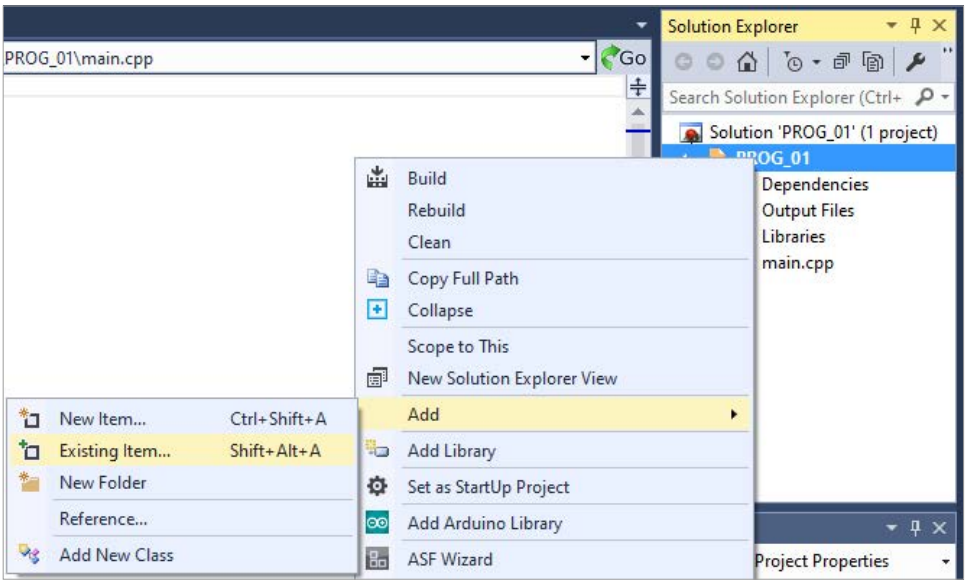
Následně se nageneruje předpřipravený zdrojový text, který posléze upravíme. Situaci po dokončení operace vytvoření nového projektu ukazuje obr. 1.15. Na tomto obrázku jsou také označeny nejdůležitější části vývojového prostředí: **textový editor** (pro zápis zdrojového textu), **Solution Explorer** (průzkumník projektu, který slouží pro organizaci zdrojových souborů projektu) a okénko s hlášením výsledků překladu.



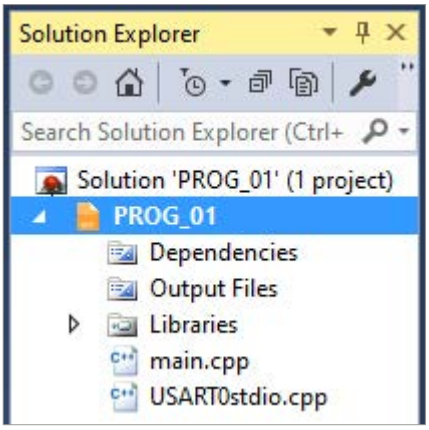
Obrázek 1.15 Vývojového prostředí a jeho nejdůležitější části

Pro zjednodušení sériové komunikace s počítačem přidáme do projektu soubor **USART0stdio.cpp** (je k dispozici v elektronické příloze této knihy).

Tuto akci provedeme kliknutím pravým tlačítkem myši v okně **Solution Explorer** na položce odpovídající názvu projektu. V zobrazené místní nabídce vybereme **Add → Existing Item**. Alternativně lze použít zkratovou kombinaci kláves **Shift + Alt + A**. Pomocí dále zobrazeného dialogu vyhledáme soubor **USART0stdio.cpp**. Výsledek by měl odpovídat obr. 1.17.



Obr. 1.16 Přidání existujícího souboru do projektu

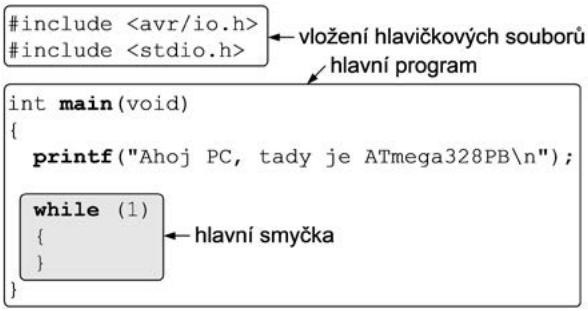


Obr. 1.17 Stav po přidání souboru USART0stdio.cpp do projektu

Nyní upravíme zdrojový text souboru **main.cpp**.

Zdrojový soubor **main.cpp** (již podle svého označení) obsahuje zápis hlavního programu. V úvodu je třeba pomocí direktiv `#include` vložit tzv. hlavičkové soubory, které obsahují informace o používaných funkcích. V níže uvedeném příkladu používáme konkrétně funkci **printf** (print formatted) z hlavičkového souboru **stdio.h**, tato funkce slouží pro výpis textu na tzv. standardním výstupu. V našem případě je pomocí souboru **USART0stdio.cpp** přesměrována na sériovou linku. Další detaily k funkci `printf` jsou uvedeny v kapitole 2.

MAIN.CPP:



PROG_1

Samotný hlavní program je označen jako funkce `main`.

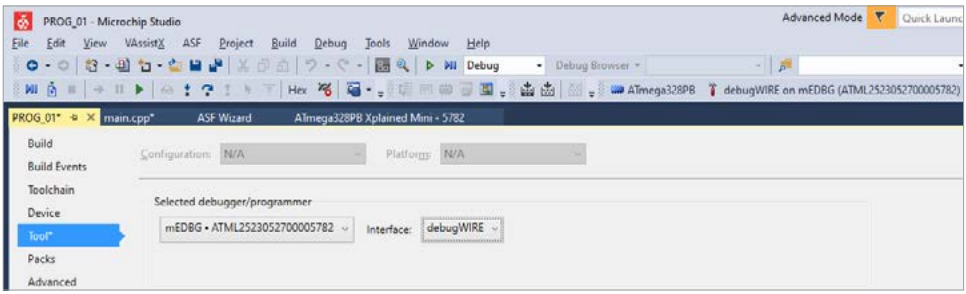
První příkaz hlavního programu odpovídá vyvolání funkce `printf` s tím, že se vypíše (tedy odešle sériovou linkou) příslušný text. Speciální znak `\n` (čteme: *zpětné lomítko en*) zajistí ukončení řádku (myšleno n jako new line, tedy nový řádek).

Druhým příkazem je tzv. **hlavní smyčka** programu realizovaná cyklem `while`. Do hlavní smyčky programu zapisujeme příkazy, které se mají neustále opakovat. Nyní je smyčka prázdná. To znamená, že po výpisu textu se program zacyklí a mikrokontrolér vyčkává na případný reset nebo přeprogramování.

Jazyk C používá blokovou strukturu, příkazy se sdružují do bloků začínajících levou složenou závorkou { a končící pravou složenou závorkou }.

Poznámka: Přestože hlavní smyčka neobsahuje žádné příkazy, musíme ji v programu ponechat. Mikrokontrolér totiž musí stále vykonávat nějaké příkazy. Nemůže se zastavit. Prázdná hlavní smyčka zajistí neustálý běh programu. To je velký rozdíl proti programům klasických počítačů, které mohou být kdykoli ukončeny. Avšak i v klasickém počítači stále běží operační systém. Tedy ani procesor klasického počítače se nemůže zastavit.

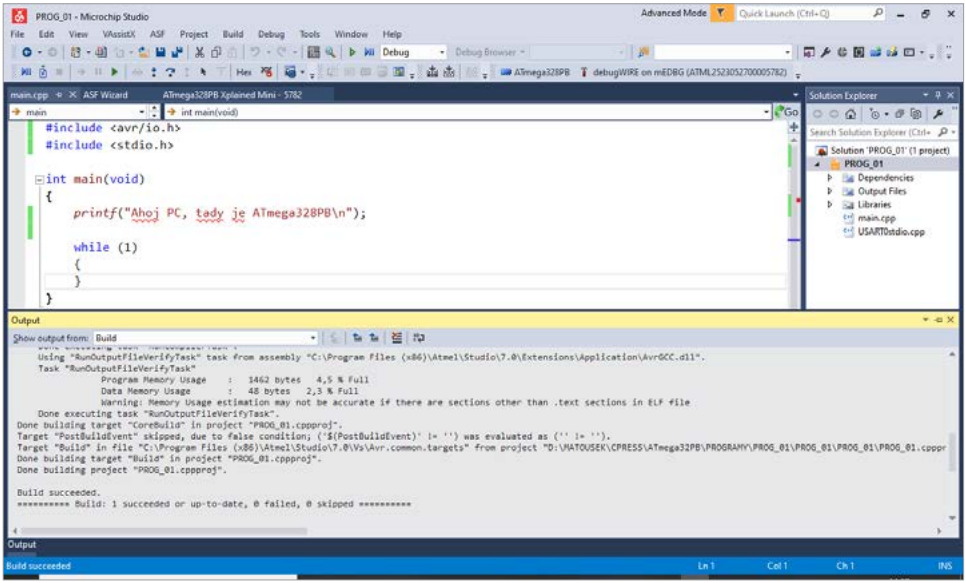
Proto, aby byl vývojový kit použit pro naprogramování mikrokontroléru, vyhledáme položku nabídky **Project → Options**. Zobrazí se nová záložka s titulkem odpovídajícím názvu projektu **PROG_01** (na obr. 1.18 je umístěna zcela vlevo). V ní vyhledáme položku **Tool** a pomocí komboboxu **Selected debugger/programmer** zvolíme připojený vývojový kit. Poté pomocí druhého komboboxu **Interface** vybereme programovací rozhraní **debugWIRE**.



Obrázek 1.18 Nastavení vývojového kitu pro programování mikrokontroléru

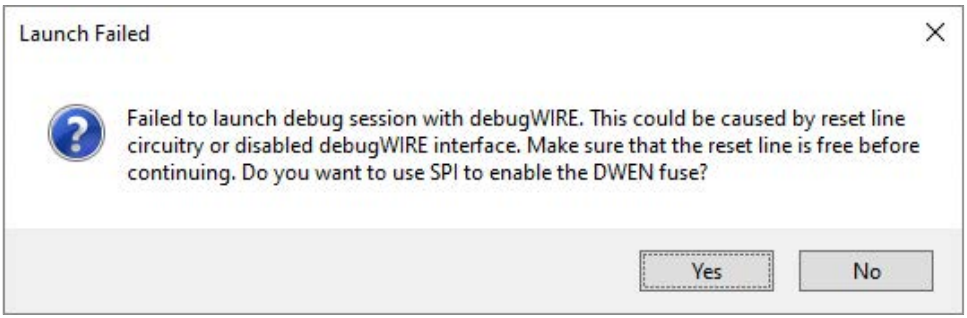
Tuto záložku můžeme nyní již zavřít nebo se pouze přepneme zpět do záložky editoru označené **main.cpp** (na obr. 1.18 druhá zleva).

Překlad programu vyvoláme pomocí nabídky **Build → Build Solution (F7)**. V případě úspěšného překladu obdržíme jako výsledek hlášení o úspěšném sestavení programu dle obr. 1.19 dole.



Obrázek 1.19 Program je úspěšně přeložen.

Nyní již přikročíme k naprogramování mikrokontroléru. Vyhledáme nabídku **Debug → Start Without Debugging (Ctrl + Alt + F5)**. Při prvním výběru rozhraní **debugWIRE** je požadováno potvrzení, že souhlasíme s přepnutím programovaného mikrokontroléru do režimu **debugWIRE** dle obr. 1.20 (výchozí režim totiž odpovídá použití rozhraní **ISP**). Potvrdíme stiskem tlačítka **Yes**.

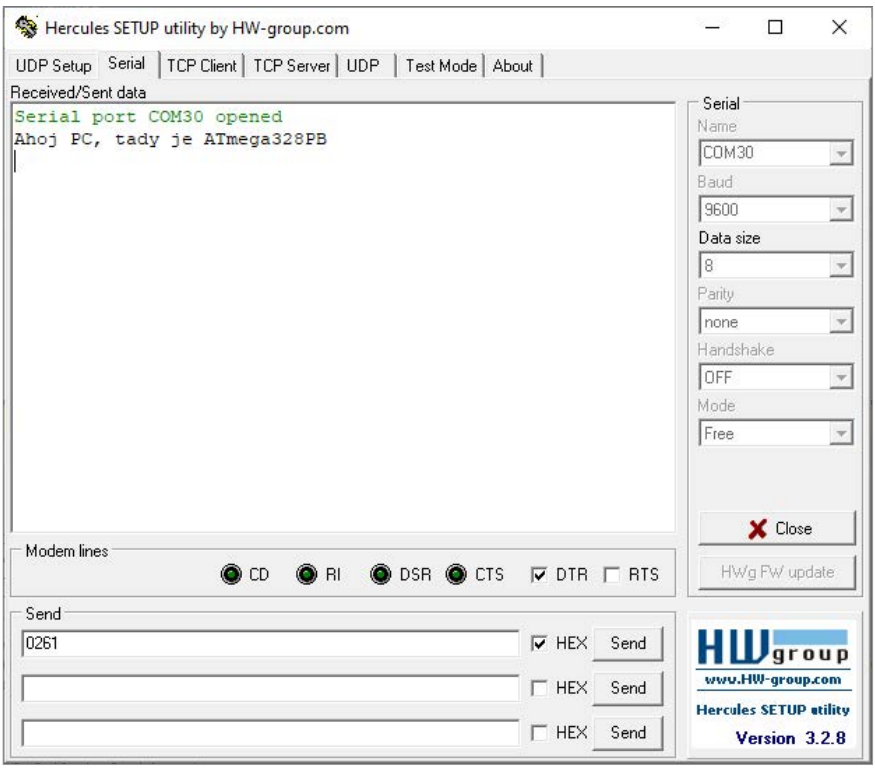


Obrázek 1.20 Dotaz na povolení rozhraní debugWIRE

Po úspěšném navázání komunikace mezi počítačem a vývojovým kitem se uskuteční programování mikrokontroléru. Průběh programování je indikován ve spodní liště vývojového prostředí.

Program **Hercules** nastavený dle obr. 1.12 zobrazí text, který jsme vypsali/odeslali pomocí funkce **printf**. Viz obr. 1.21. Nezapomeňte zaškrtnout pole **DTR**.

V případě, že program Hercules spustíme až po naprogramování mikrokontroléru, neuvidíme žádný text. Je to způsobeno tím, že mikrokontrolér odešle text pouze jednou. Po opakovaném naprogramování mikrokontroléru by se již text měl objevit.



Obrázek 1.21 Výsledek výpisu funkcí printf v prvním příkladu

Výše popsaný postup programování budeme používat při vytváření dalších programů. V následujících kapitolách budou na příkladech postupně probírány jednotlivé jednotky mikrokontroléru ATmega328PB společně s výkladem programovacích konstrukcí v jazyce C.

KAPITOLA 2

Základní datové typy jazyka C

V této kapitole:

- Bity a bajty
- Rozdělení základních datových typů
- Celá čísla
- Reálná čísla
- Deklarace proměnné
- Funkce printf
- Základní aritmetické operace s celými a reálnými čísly
- Unární aritmetické operátory
- Stručně o typové konverzi
- Priorita a asociativita dosud probraných operátorů
- Konstanty

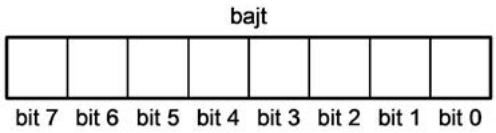
Titul knihy Niklause E. Wirtha „*Algoritmy + datové struktury = programy*“ nám sděluje, že pro úspěšnou tvorbu programů se musíme naučit zapisovat **algoritmy** (algoritmus je přesný postup, jak vyřešit daný typ úlohy) a používat **datové struktury**.

V této kapitole se zaměříme na základní datové typy jazyka C. To nám dovolí vytvářet programy, které budou například schopny provádět matematické operace. Realizace algoritmů na základě jazyka C je náplní dalších kapitol.

Bity a bajty

Základní jednotkou informace je jeden bit. **Bit** může reprezentovat pouze jednu ze dvou hodnot: 0, nebo 1. Tyto dvě hodnoty mohou být prakticky chápány jako vypnutý stav (0) nebo zapnutý stav (1) nějakého zařízení nebo jednotky. Bitová hodnota 0 nebo 1 též může také reprezentovat stav nějaké jednotky (například stisknuté tlačítko nebo uvolněné tlačítko).

Pro reprezentaci hodnot ve větším rozsahu používáme bajt a jeho násobky. Rovněž platí, že paměť je přístupná nikoli po jednotlivých bitech, nýbrž po bajtech nebo jeho násobcích. **Bajt** (byte) je skupina osmi bitů. Každý z bitů, který je součásti tohoto bajtu, může nabývat hodnoty 0 nebo 1. Proto je rozsah čísel reprezentovaný jedním bajtem 0 až 255. Násobky velikosti bajtů jsou prakticky omezeny na mocniny čísla 2. Rozpoznáváme tedy obvykle 1bajtové, 2bajtové, 4bajtové nebo 8bajtové datové typy.



Obrázek 2.1 Bity a bajty

Rozdělení základních datových typů

Datový typ určuje *obor hodnot* (v jakém rozsahu proměnná uchovává hodnoty), *interpretaci obsahu* (číslo, znak apod.) a *množinu přípustných operací* (například čísla lze sčítat nebo porovnávat).

Důvody používání různých datových typů vyplývají především z omezené velikosti datové paměti mikrokontrolérů (obvykle se snažíme ušetřit každý bajt paměti, takže používáme datový typ co nejmenší velikosti) a z požadavků na rychlé provádění operací (údaje uložené v delších datových typech obvykle vyžadují také časově náročnější zpracování).

Mezi základní datové typy patří:

- *celočíselné datové typy* mohou reprezentovat pouze celočíselné hodnoty,
- *reálné datové typy* mohou reprezentovat desetinná čísla,
- *znakové datové typy* pro uchování jednoho znaku (viz kapitolu 12),
- *odvozené datové typy:* například pole nebo ukazatel (viz kapitolu 7).

Celá čísla

V programech pro mikrokontroléry používáme především celočíselné údaje, a to hlavně proto, že jejich zpracování je nejrychlejší.

Základní rozdělení celočíselných typů je na *celá čísla se znaménkem* a *celá čísla bez znaménka*.

Celá čísla se znaménkem mohou reprezentovat kladné i záporné hodnoty v určeném rozmezí. Základním typem této skupiny je `int`.

Celá čísla bez znaménka mohou reprezentovat pouze kladné hodnoty včetně nuly v určeném rozmezí. Základním typem této skupiny je `unsigned`.

Typy odvozené z typů **int** a **unsigned** mají užší nebo naopak širší obor hodnot proti těmto základním typům.

Pro lepší orientaci ve velkém množství dostupných datových typů budeme v této knize používat datové typy se snadno zapamatovatelnými názvy typu: `intX_t` (pro celá čísla se znaménkem) a `uintX_t` (pro celá čísla bez znaménka). Symbol `X` značí velikost datového typu v bitech a platí, že X může být 8, 16 nebo 32. Jsou to tudíž 1bajtové, 2bajtové nebo 4bajtové celočíselné datové typy. Pro použití těchto datových typů musíme do zdrojového textu vložit

hlavičkový soubor **stdint.h** (jako *standard integer*). Proto na začátku zdrojového textu v jazyce C zapíšeme direktivu:

```
#include <stdint.h>
```

Jednotlivé typy definované v souboru **stdint.h**. jsou uvedeny formou tabulky 2.1.

První sloupec odpovídá označení datového typu, které budeme používat. Druhý sloupec je původní označení datového typu v jazyce C. Následují sloupce udávající velikost datového typu v bajtech (tedy kolik zabere místa v paměti) a číselný rozsah.

Poslední sloupec určuje příponu, kterou je třeba použít při zápisu číselných hodnot větší šíře. Jako základní šíře se uvažuje 16bitové celé číslo se znaménkem (`int16_t`). Hodnoty přesahující tento rozsah je nutné opatřit příponou. Přípona je tvořena podle jednoduchých pravidel: *U – unsigned*, *L – long*, *UL – unsigned long*. Například hodnotu 16 milionů bez znaménka je třeba zapsat jako `16000000UL`.

Tabulka 2.1 Datové typy pro celá čísla definované v souboru stdint.h

Datový typ	Původní název	Velikost	Číselný rozsah	Přípona
`uint8_t`	unsigned char	1 bajt	0 až 255	*není*
`uint16_t`	unsigned	2 bajty	0 až 65 535	`U`
`uint32_t`	unsigned long	4 bajty	0 až 4 294 967 295	`UL`
`int8_t`	char	1 bajt	–128 až +127	*není*
`int16_t`	int	2 bajty	–32 768 až +32 767	*není*
`int32_t`	long	4 bajty	–2 147 483 648 až +2 147 483 647	`L`

V naprosté většině případů dáváme přednost datovým typům bez znaménka, protože hodnoty, se kterými pracujeme, nejsou obvykle záporné (například počet impulzů nebo kmitočet nejsou nikdy záporná čísla).

Datový typ je vhodné volit přiměřený k rozsahu hodnot, se kterými pracujeme. Načítáme-li údaje z 8bitového registru mikrokontroléru, použijeme nejčastěji typ `uint8_t`. Tomuto typu budeme dávat obecně přednost, protože 8bitová architektura použitého mikrokontroléru s ním dokáže pracovat nejrychleji.

Zápis hodnoty celého čísla

Překladače jazyka C pro mikrokontroléry podporují zápis celých čísel ve dvojkové, osmičkové, desítkové a šestnáctkové soustavě podle těchto pravidel:

- **Dvojková** (binární) soustava: zápis začíná znaky `0b`, platné jsou číslice `0` a `1`. Příklad: 0b10010001 je 8bitová hodnota zapsaná ve dvojkové soustavě odpovídající desítkové hodnotě 145. Používání dvojkové soustavy je při programování mikrokontrolérů velmi časté.
- **Osmičková** (oktalová) soustava: zápis začíná číslicí `0`, platné jsou číslice `0` až `7`. Příklad: 037 je osmičková hodnota (odpovídající desítková hodnota je 31), kdežto 37 (bez úvodní nuly!) je desítková hodnota. Osmičková soustava je v současnosti používána zřídka, uvá-