Forth interpreter v MCU

Wek pred nejakou dobou písal o „kecaní“ s MCU [1] ako o procese, kedy je MCU „zavesený“ iba na sériovej linke a bez použitia akýchkoľvek iných programovacích metód je možné tvoriť, upravovať a ladiť program, vykonávaný v rezidentnom BASIC interpreteri. Toto je v zásade možné (v segmente „malých MCU“) len u intepretovaných jazykov, ako je napríklad BASIC, Forth, Java alebo Python [2].

Použitie intepreteru je tu možné preto, lebo interpreter má menšie pamäťové nároky, povedzme, jednotky kB programovej pamäte, RAM obmedzuje množstvo premenných alebo veľkosť interpretovaného programu (hoci, teraz mi na stole beží BASIC interpreter v AtTiny2313). Naproti tomu kompilátor, editor a debugger majú o niekoľko dekád väčšie pamäťové nároky, čo je pre väčšinu MCU mimo rozsah ich možností, a ak by to aj tak nebolo, tak to výrazne obmedzí ich použiteľnosť – pretože netreba zabúdať, že na takom systéme musí okrem vývojových nástrojov bežať aj samotný užívateľský program. Je zrejmé, že na počítače triedy malých MCU je vhodné nasadenie jednoduchšieho jazyka, hoci ani implementácia neúplného Java VM [3] alebo Pythonu [4] nie je vylúčená, tíško zamlčujúc jej účelnosť. Keď som prýkrát skúšal BASIC-52 vo wek-ovej verzii, napadlo mi jeho zaujímavé využitie (na ktoré odkážem ku koncu tohto článku) a začal som hľadať aj iné, podobné možnosti. Do oka mi padol projekt FlashForth, ktorý je orientovaný na PIC MCU a – ako jeho názov naznačuje – je to interpreter jazyka Forth.

Forth

Jazyk Forth má za sebou dlhú históriu, ale aj napriek tomu je pomerne málo známy. Je to pravdepodobne spôsobené popularitou jazykov BASIC a C, ktoré sú mimochodom bližšie spôsobu chápania bežných ľudských bytostí, zatiaľčo Forth je bližší vnútorným procesom počítačov – z toho vyplýva jeho jednoduchosť (na implementáciu). Ale jeho komplikovanosť sa prejavuje len na prvý (no, niekedy aj na druhý, tretí...) pohľad a po istom čase analyzovania jazyka človek príde na to, že je to veľmi logický a jednoduchý jazyk. Nebudem popisovať jeho štruktúru, pretože by táto bola popísana už mnohokrát [5], [6].

FlashForth

Projekt FlashForth (ďalej len FF, autor Mikael Nordman) [7] je jednoduchý Forth interpreter, ktorý definície slov zapisuje priamo do FLASH pamäte použitého PIC, čím je zabezpečená pomerne veľká rýchlosť vykonávaného programu. Varianty FF existujú pre PIC18F aj pre dsPIC30F. Vybral som si variant pre dsPIC30F - síce to nie je moja obľúbená rodina PIC-iek, mal som dôvodné podozrenie, že na túto úlohu budú praktickejšie tieto výkonnejšie typy než PIC18F.

Ako na vec?

Celé je to pomerne jednoduché. FF je navrhnutý tak, aby jeho portovanie na akékoľvek dsPIC30F bolo jednoduché. Obsahuje dva súbory – ff30_46.s (pre verziu 4.6) a ff30.inc. Na prispôsobenie sa použitému HW je určený súbor ff30.inc.

Podstatné nastavenia sú tieto:

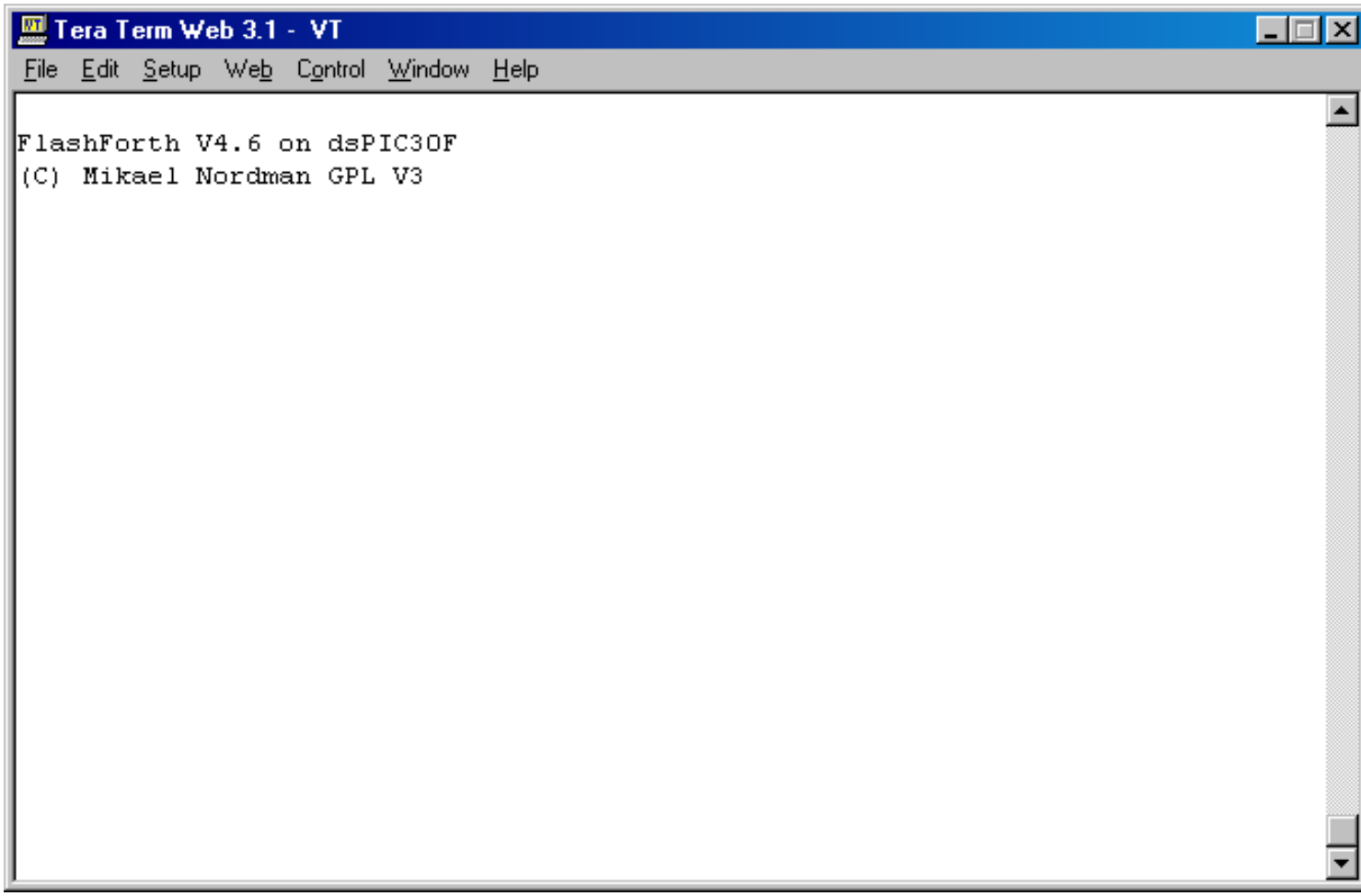
```
1.
   config __FOSC, HS          ;16MHz Xtal

2.


3.
   .equ FREQ, 16000000          ; Clock (Crystal)frequency (Hz)

4.
   .equ BAUDRATE, 2400          ; Serial baudrate
```

Prvé je nastavenie typu zdroja hodín pre MCU – toto je súčasť konfiguračných bitov. Je treba voliť podľa datasheetu, v mojom prípade ide o 16MHz kryštál, tak som zvolil HS. Je možné aj použitie interného PLL alebo interného oscilátora – pretože pre teploty –40 až 85 stupňov Celzia v kombinácii s variáciou napájacieho napätia od 3 do 5,5V má nekalibrovaný interný oscilátor maximálne 2%-nú odchylku od frekvencie 7,3728MHz, čo by malo na komunikáciu cez USART stačiť. Ďalej je treba nastaviť frekvenciu použitého zdroja hodín a požadovanú rýchlosť sériovej linky. Tým by malo byť všetko pripravené na „asembláž“ a po nej by mal zostať .hex súbor obsahujúci binárku, ktorú treba naprogramovať do MCU. Ak je pripravené všetko ostatné – teda plošný spoj s PIC-kom, MAX232 alebo podobným RS232 linkovým budičom a trochou bižutérie (obligátne blokovacie kondenzátory a rezistor na MCLR pine).

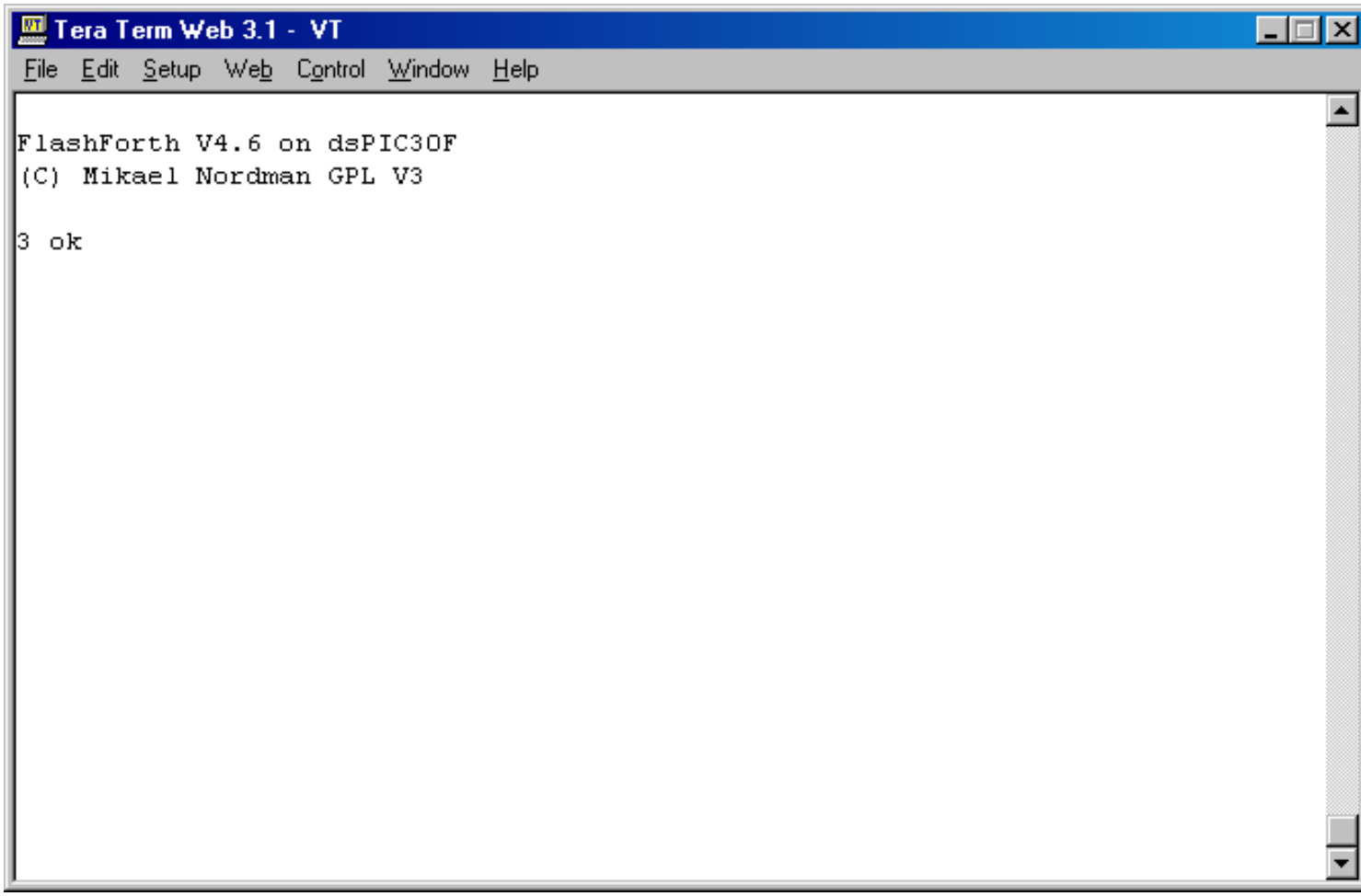
Treba nastaviť obľúbený terminál na požadovanú baudovú rýchlosť, nastaviť Xon-Xoff handshaking, a môžeme sa pustiť do programovania.

Na počiatku bolo slovo...

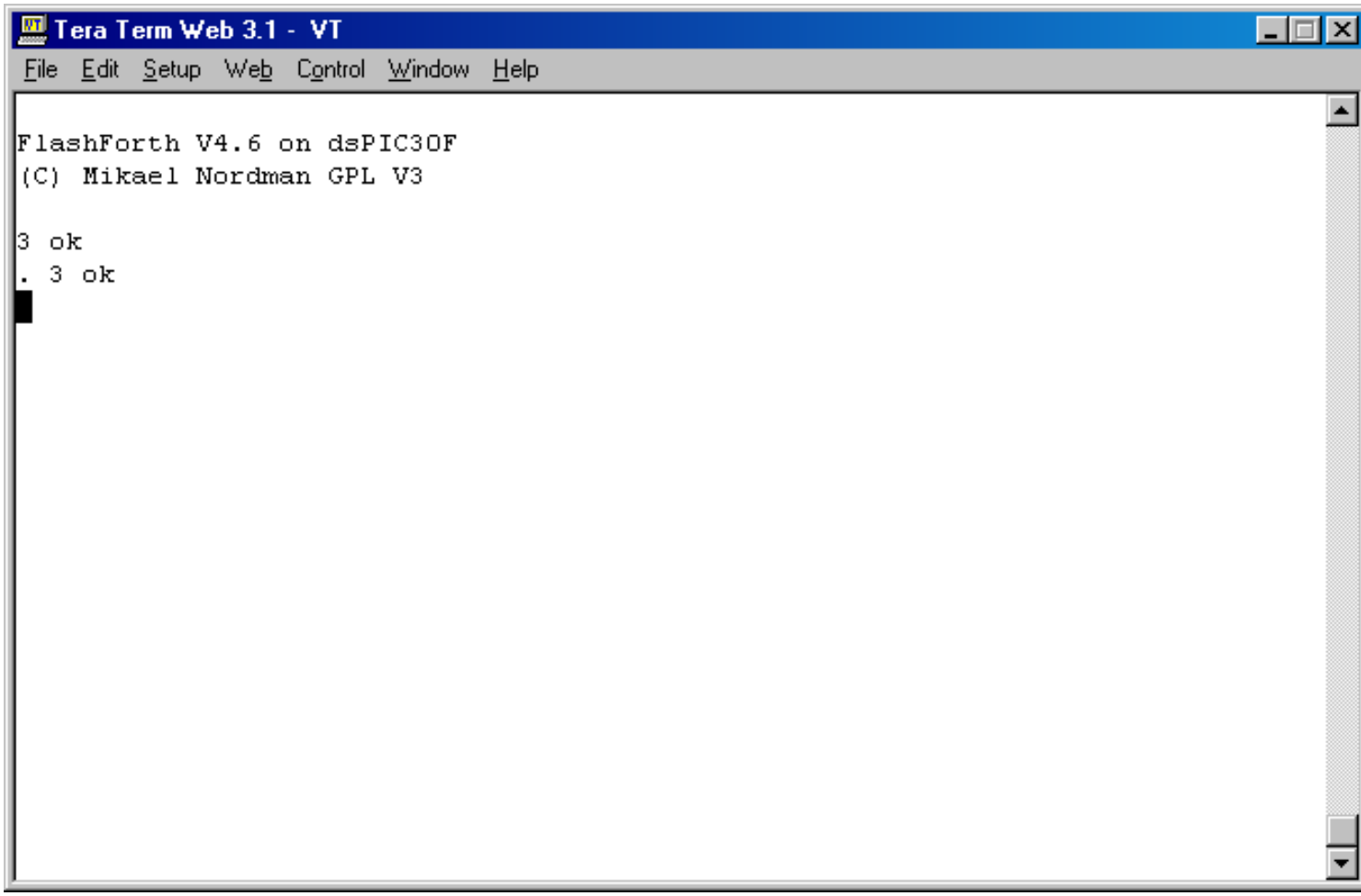
Ak je všetko v priadku, po resete sa ohlási FF asi takto:



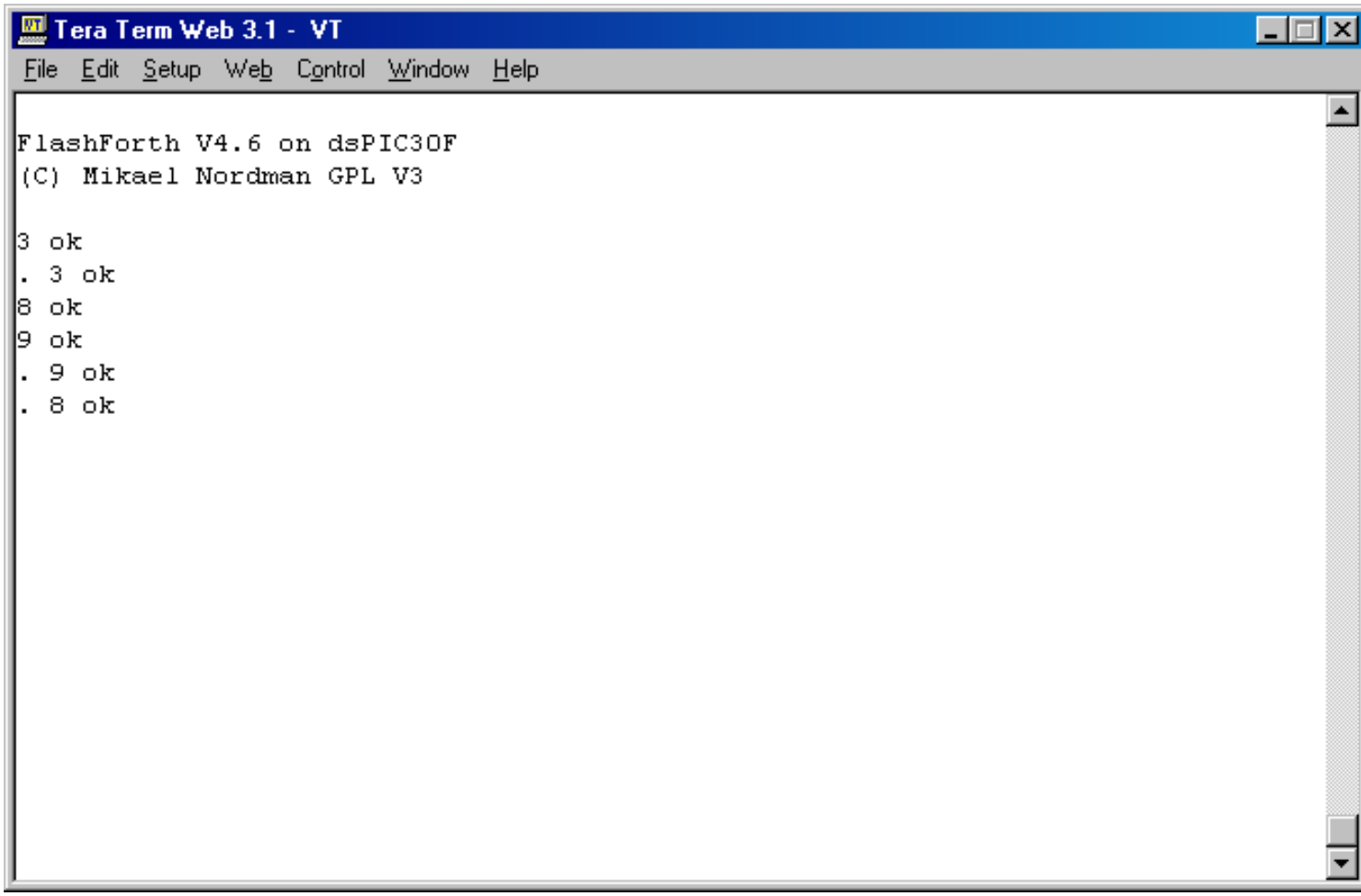
To znamená, že FF funguje a je pripravený na komunikáciu. Napríklad - vložme nejaké číslo na vrchol zásobníka. Stačí napísať „3“ a stlačiť Enter. FF odpovie „ok“, čo znamená, že vykonal čo mal.



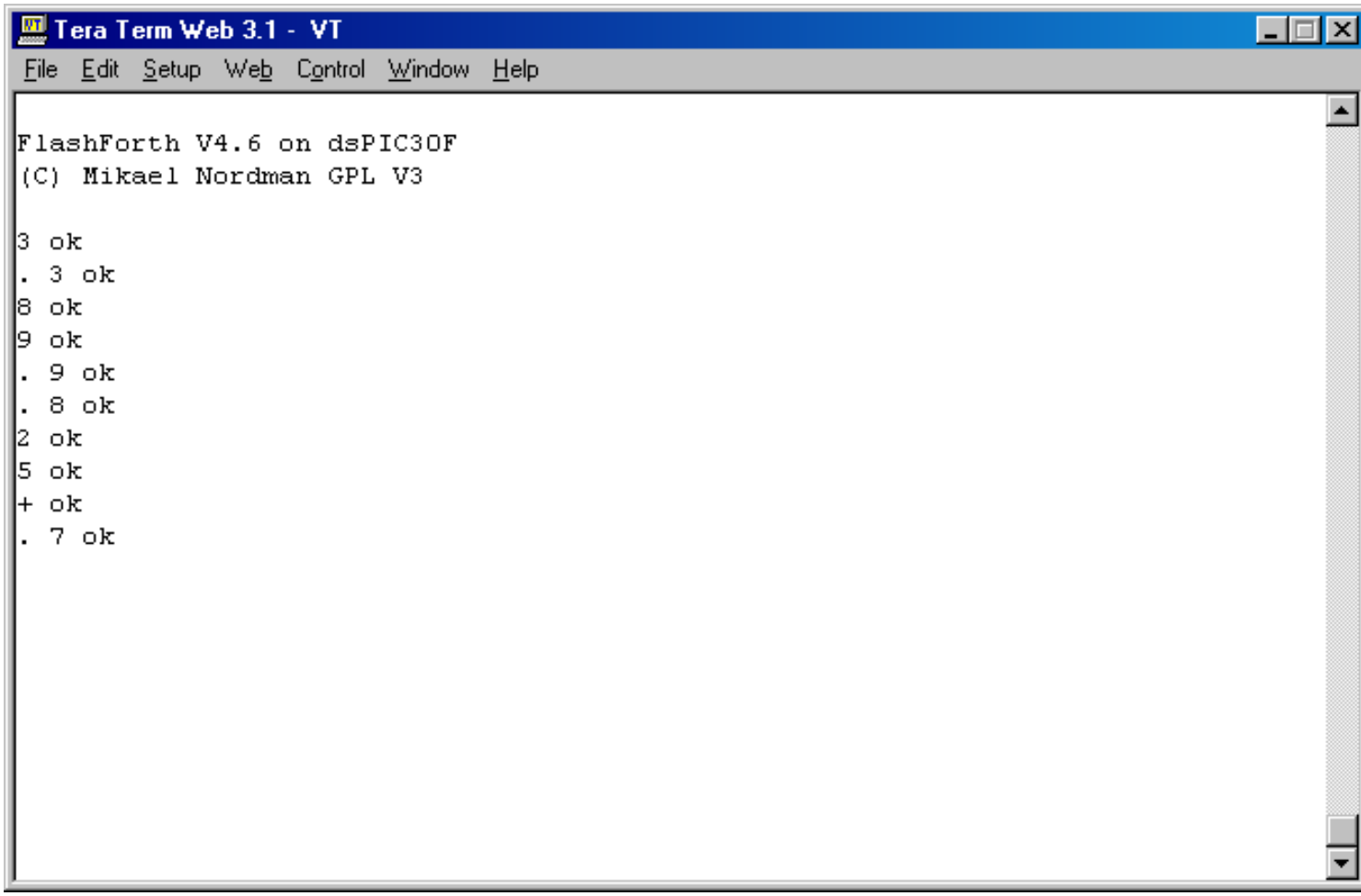
Číslo, uložené na zásobníku nevidno. Preto môžeme použit slovo, ktoré vezme číslo zo zásobníka, vypíše ho a odstráni zo zásobníka, teda slovo „.“ (bodka).



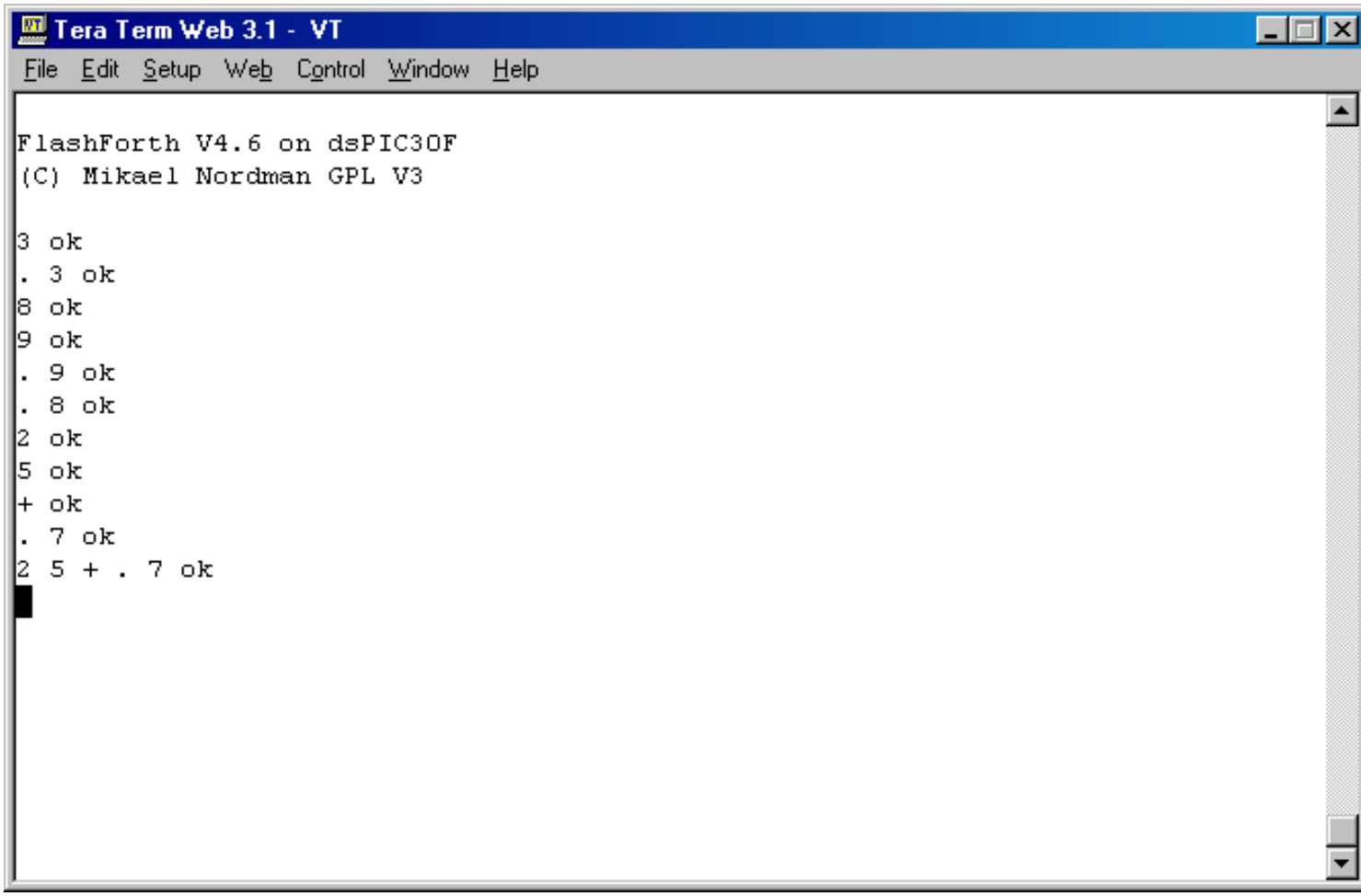
Ak na zásobník uložíme dve čísla a dvakrát dáme bodku, vypíšu sa obe čísla, najprv to druhé, potom prvé.



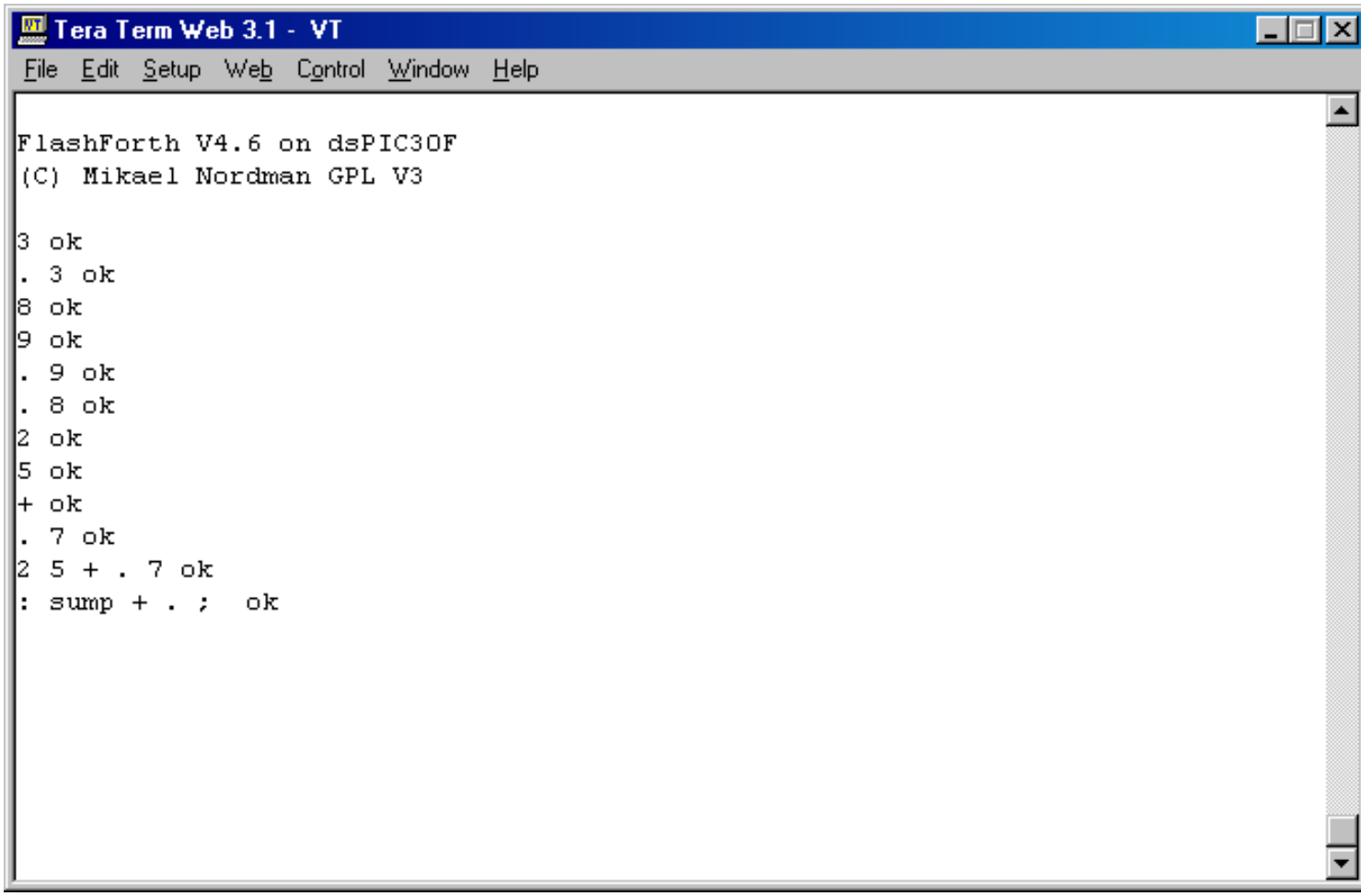
A môžeme skúsiť aj nejakú matematickú operáciu, napríklad súčet.



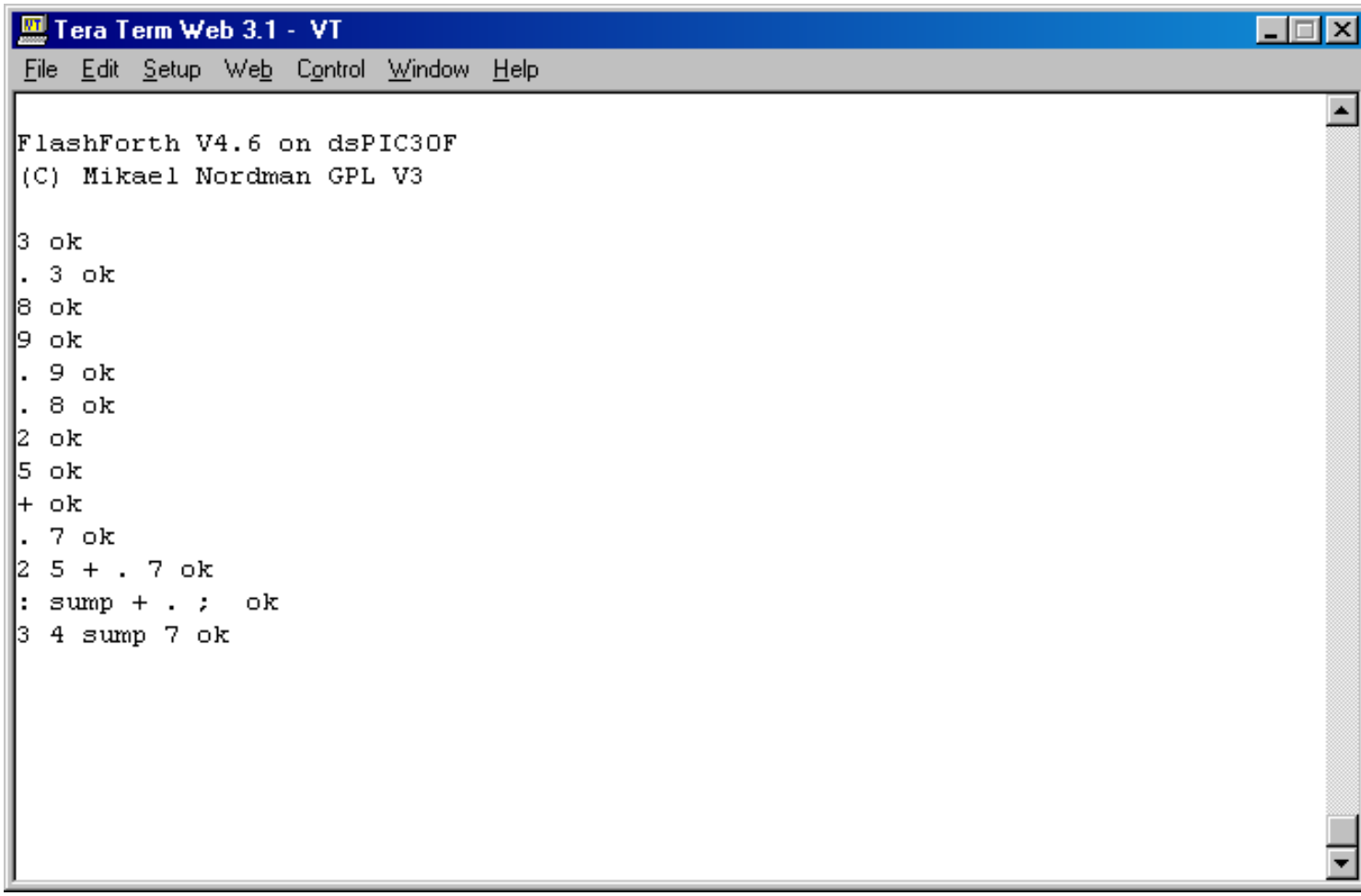
Teda prvé číslo na vrchol zásobníka, naň druhé číslo, „+“ vezme zo zásobníka obe čísla, sčíta ich a na vrchol zásobníka (ktorý je medzičasom prázdny) vloží ich súčet. Ten samozrejme nevidno až kým sa nevykoná slovo „.“.
Toto všetko sa dá vložiť na jeden riadok:



Doteraz sme používali slová, ktoré sô už definované, ale môžeme si zadefinovať aj vlastné slovo. Napríklad slovo, ktoré vykona súčet a vypíše ho na terminál – teda funkcia „+“ a .„“. Definícia začína dvojbodkou, ďalej nasleduje názov slova a potom jeho samotná definícia. Táto končí bodkočiarkou.



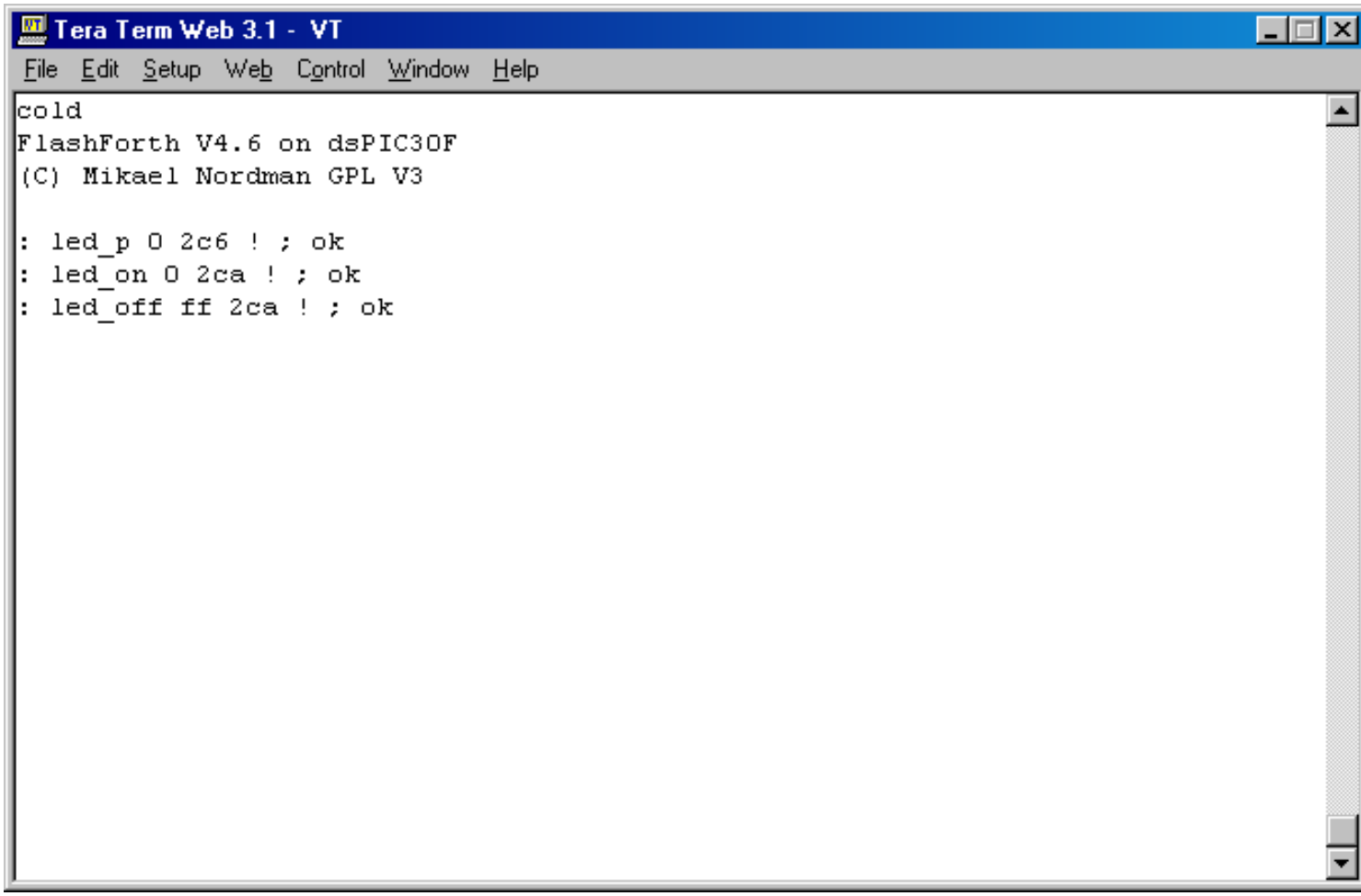
FF sa ohlási svojím klasickým „ok“, ktoré informuje o tom, že slovo bolo vytvorené. A teraz toto novodefinované slovo („sump“ – suma a plus) môžeme otestovať. Pretože suma očakáva dve čísla na zásobníku, zadáme ich a potom nasleduje slovo „sump“:



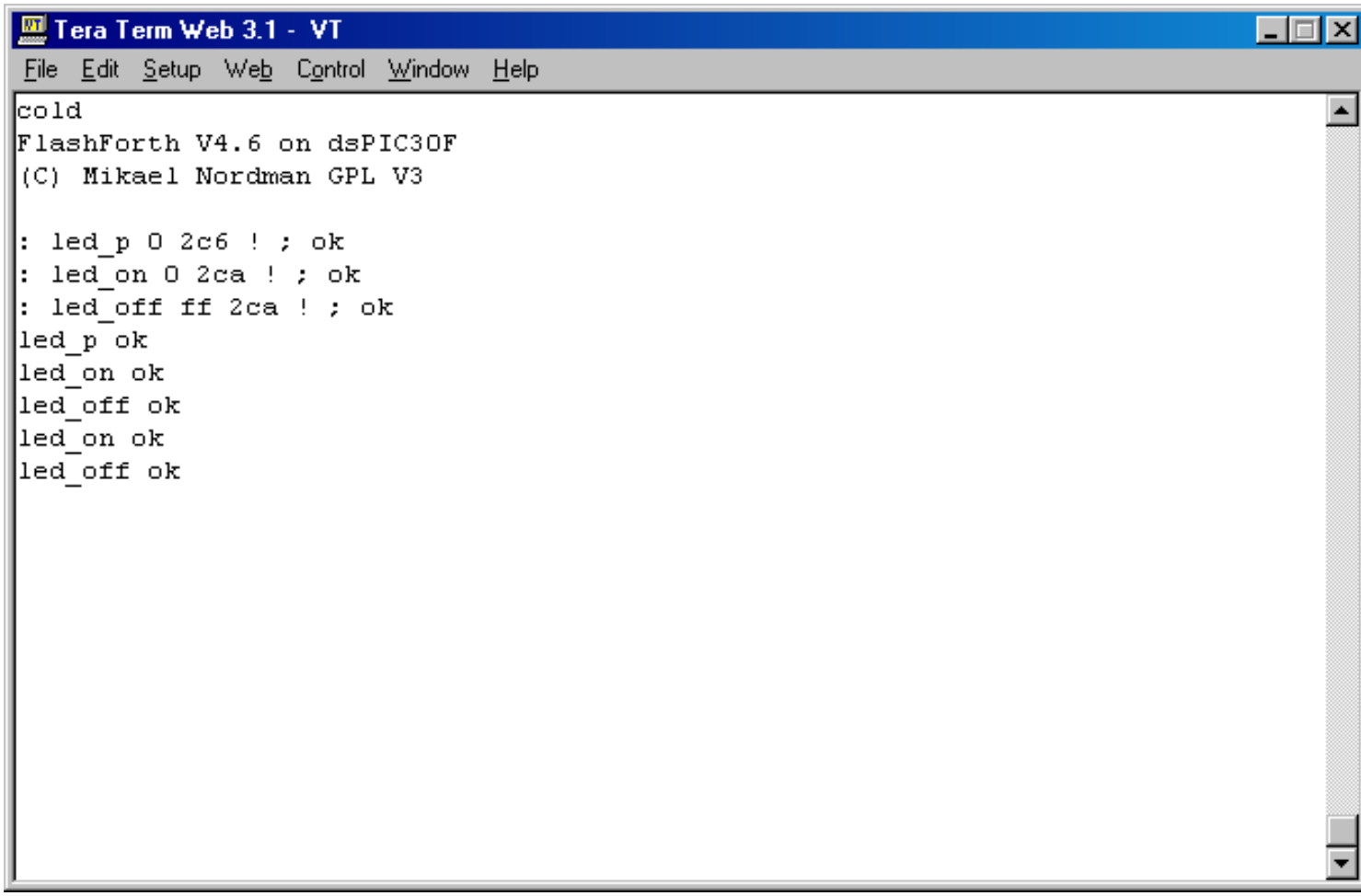
No, takže to asi aj naozaj funguje.

Blikanie LED

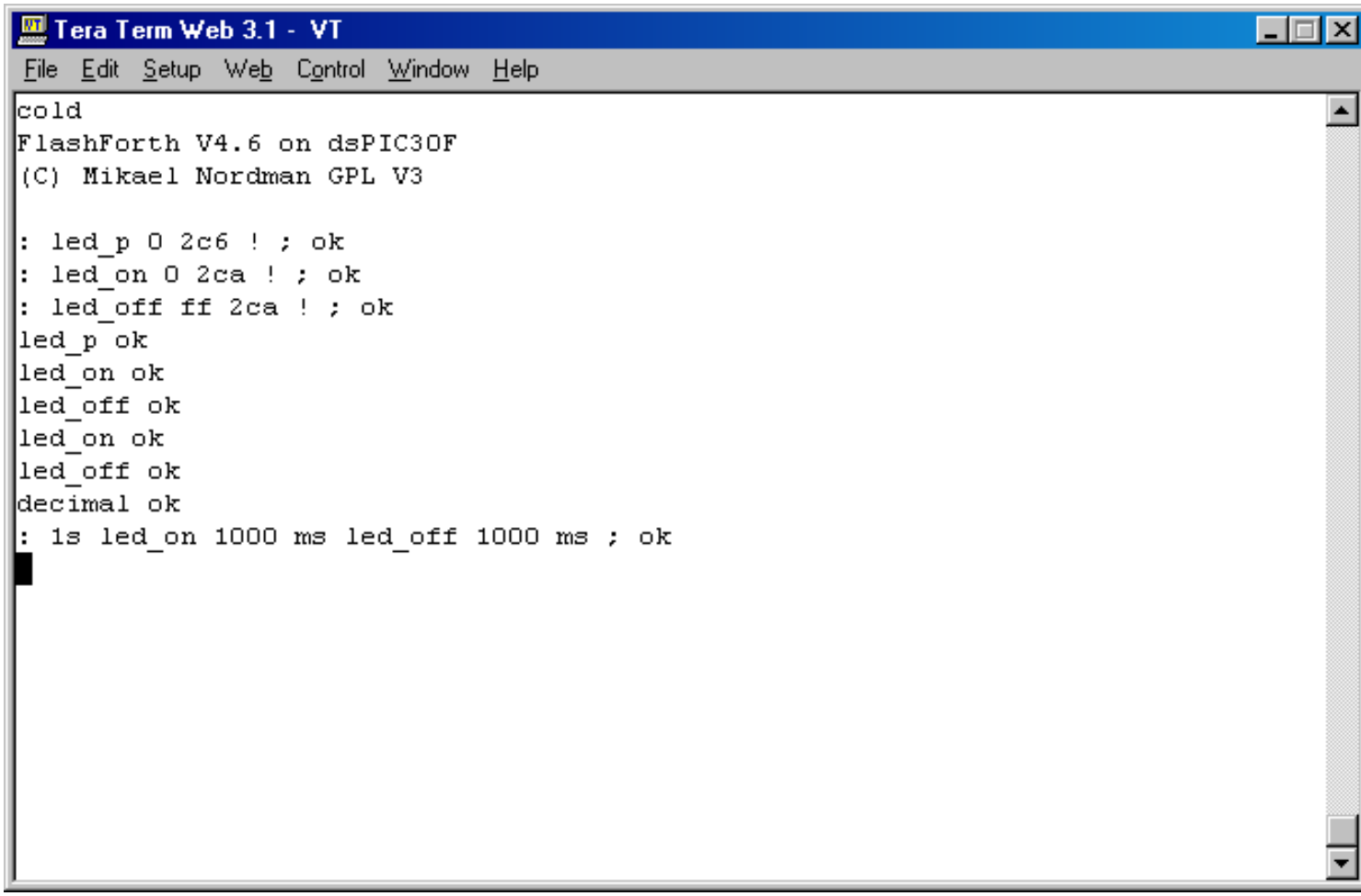
Pretože je vysoko pravdepodobné, že sôčet dvoch čísel nebude poslaním embedded systému, skúsime sa pohrať viac s hardvérom. Na porte B je LED dióda, katódou na pine MCU, anódou cez rezistor na Vdd. Nebudem už taký podrobný ako doteraz, tak len v stručnosti – zadal som takéto tri definície slov led_p, led_on a led_off.:



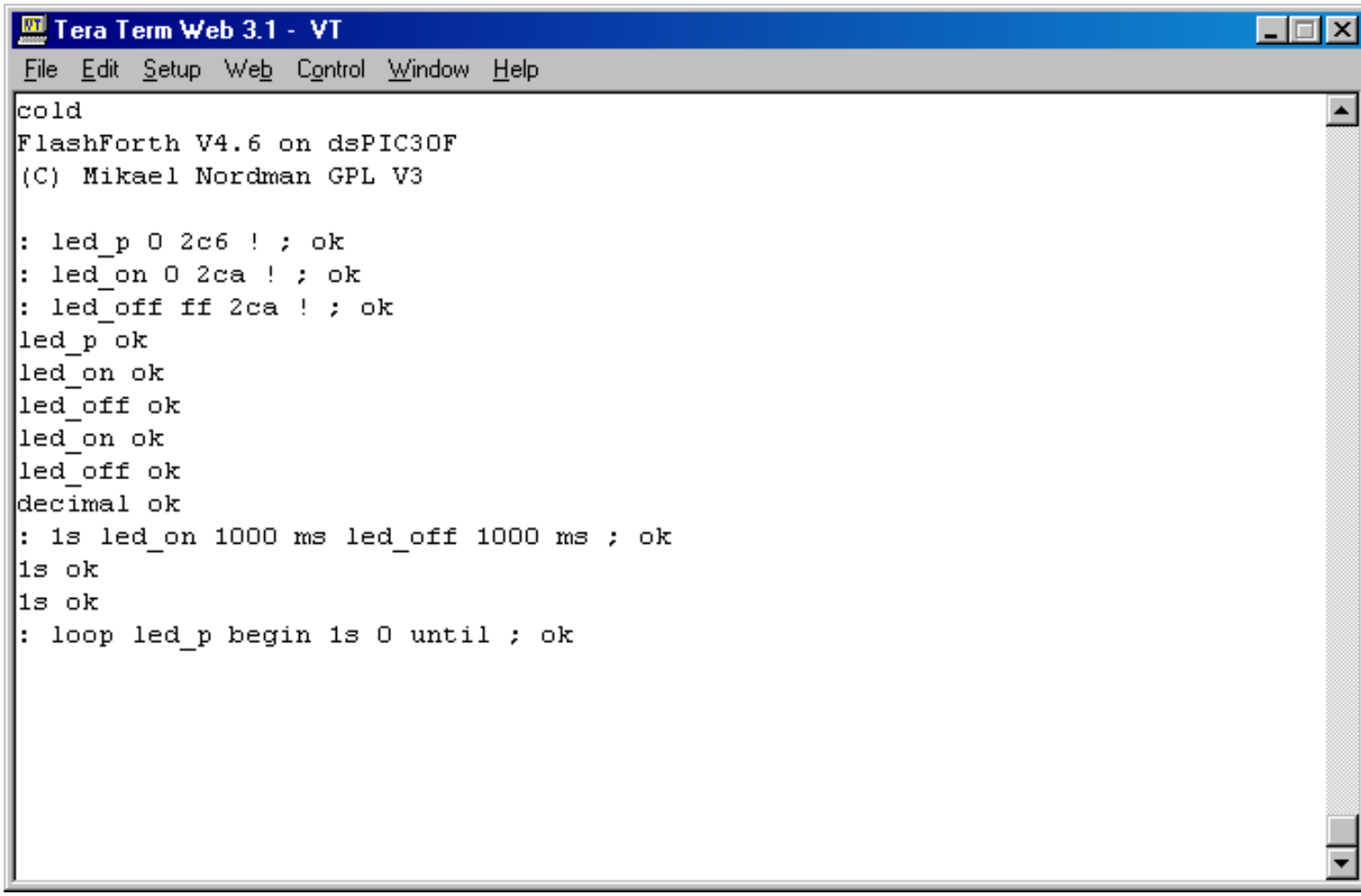
Všetky tri zapisujú do pamäte pomcou slova „!“, v prvom prípade sa vynuluje register TRISB, čím sa prepne do funkcie výstupu, dve zvyšné definície definjô slovo pre rozsvietenie (0x00 do LATB) a zhasnutie LED (0xff do LATB). Tieto môžem rovno otestovať, či robia to, čo majú:



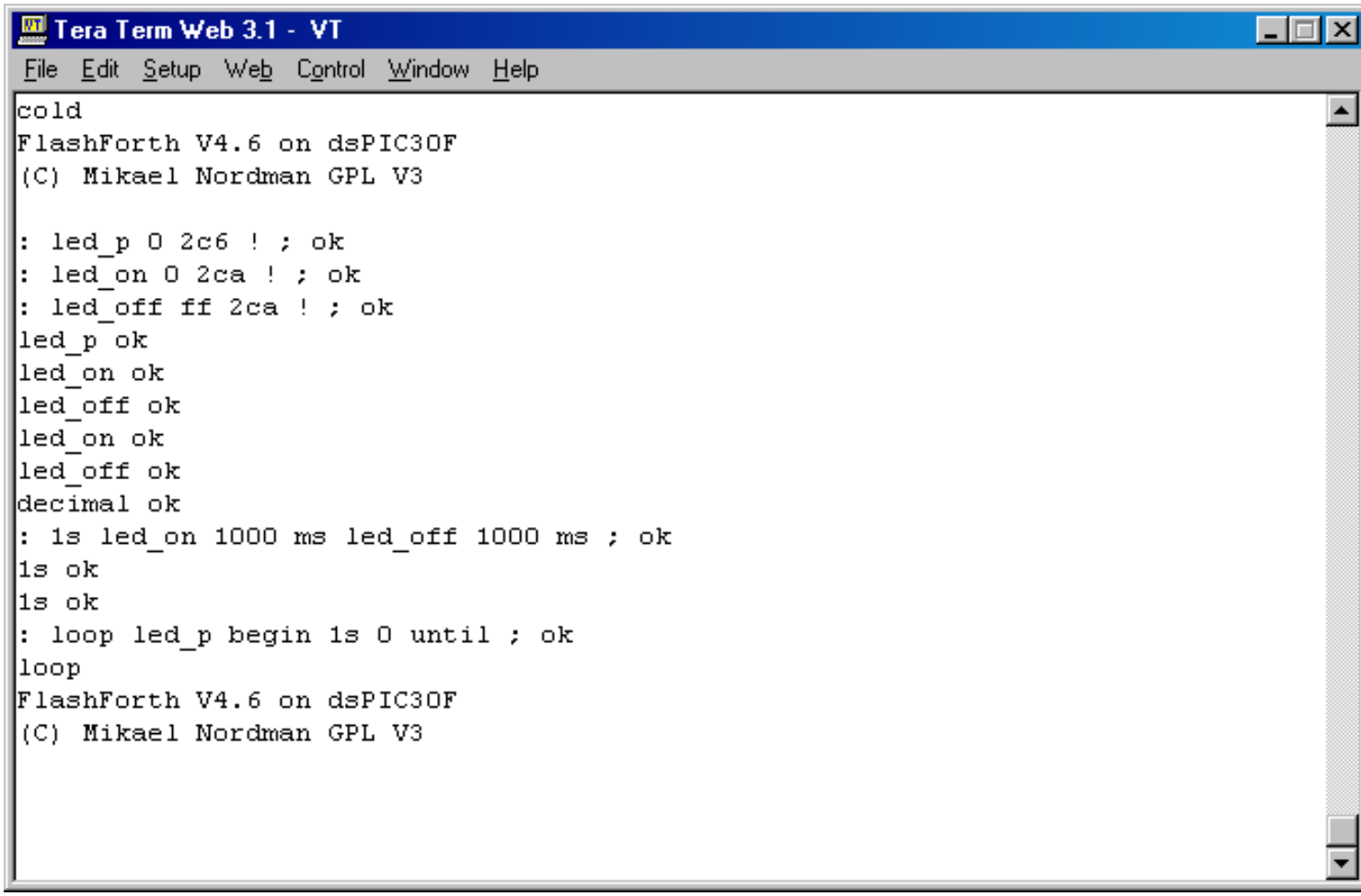
LED sa skutočne rozsvietila, resp. zhasla, podľa očakávania. Tým máme spodnú SW vrstvu zapísanú a otestovanú, môžeme prejsť k skutočnému programátorskému evergreenu – blikanie LED-kou. Najprv si vytvorme slovo, ktoré na jednu sekundu zasvieti LED, potom ju zhasne a sekundu počká (so zhasnutou LED):



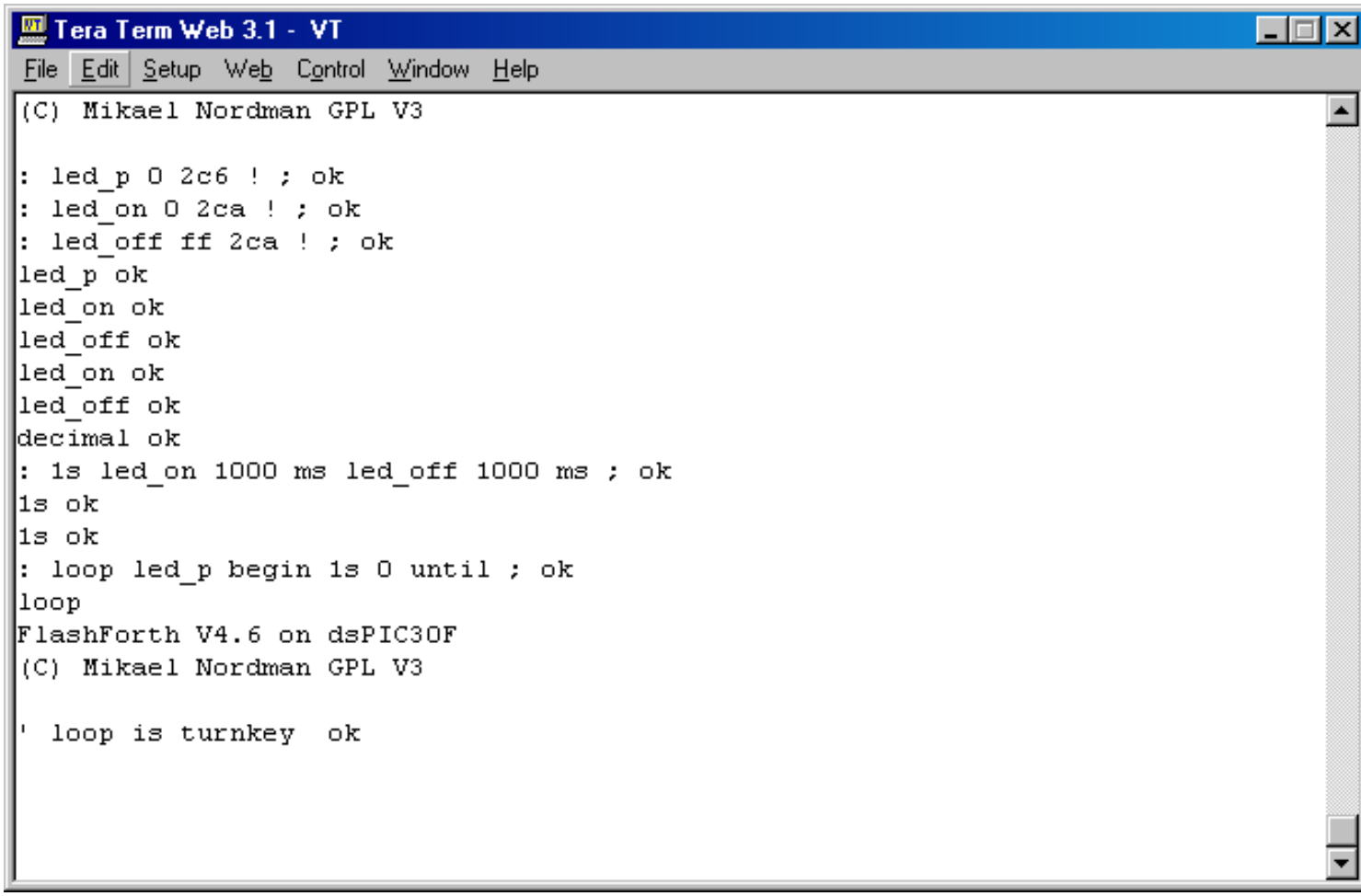
Toto slovo môžeme otestovať tak, že zadáme jednoducho „1s“ a LED na sekundu zasvieti, sekundu zostane zhasnutá a potom FF odpovie svojim „ok“ na znamenie, že vykonal všetko, čo mal. Potom zadajme definíciu slova loop takto:



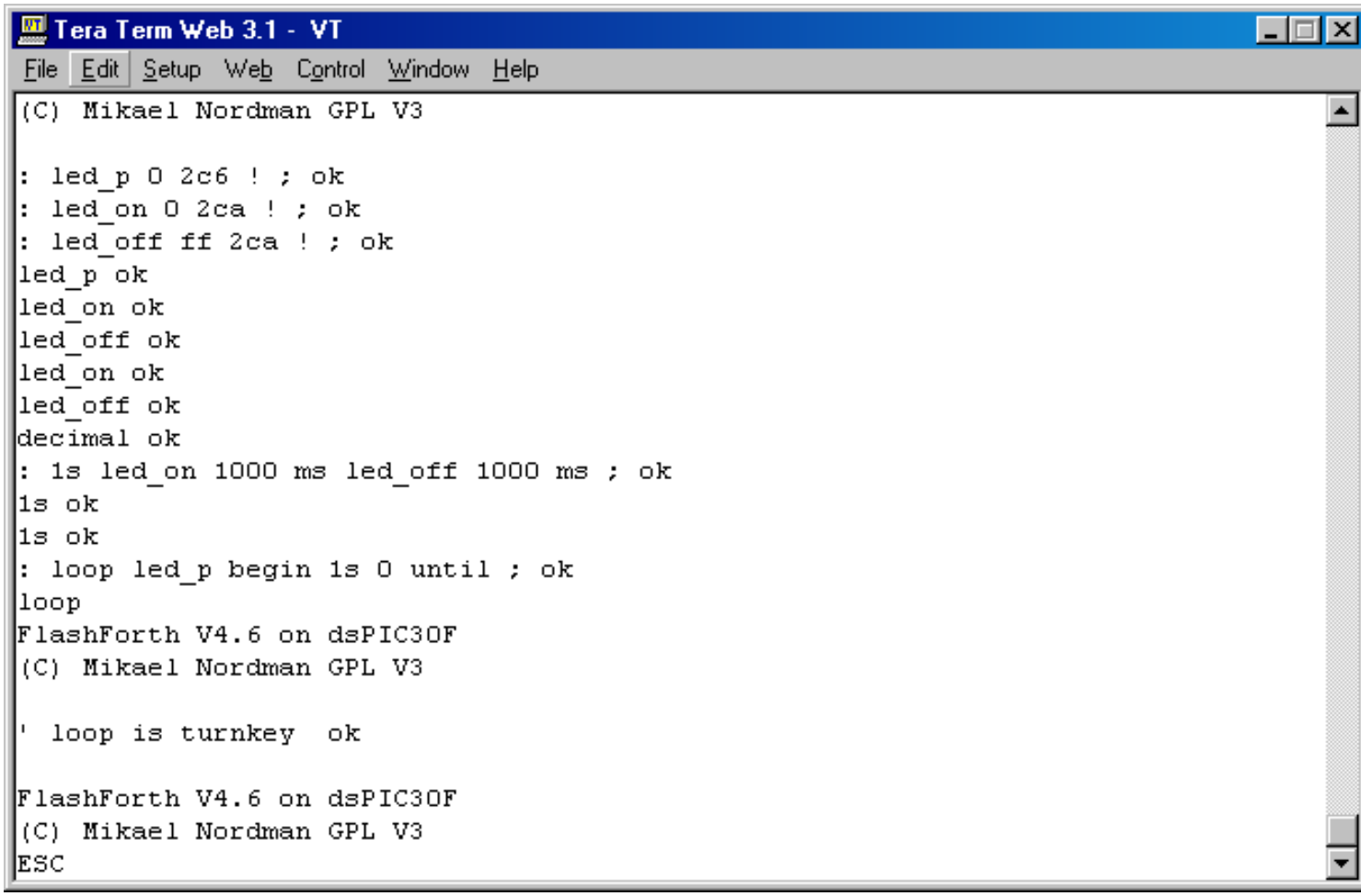
Toto slovo spustí led_p, a začne nekonečnú slučku, ktorá bude opakovať slovo „1s“, ktoré sme si pred chvíľkou otestovali, a toto pozostáva zo slov „led_on“ a „led_off“, ktoré sme skúšali ešte predtým. Po zadaní slova „loop“ začne LED blikať. FF neodpovie svojim obligátnym „ok“, pretože vykonávanie slova bude pokračovať až do skonania sveta. Alebo kým nenastane reset, napríklad Ctrl-O zo sériovej linky, ktorým sa vykoná „warm reset“.



Teda sme naspäť v hre, so svojim pekne definovaným blikajúcim programom. Ak chceme, aby sa tento program spustil vždy po resete, musíme označiť slovo, ktoré treba vykonať, ako „turnkey“. Asi takto:



FF odpovie „ok“. Po resete, s nastavenym „turnkey“, sa stane nasledovné:



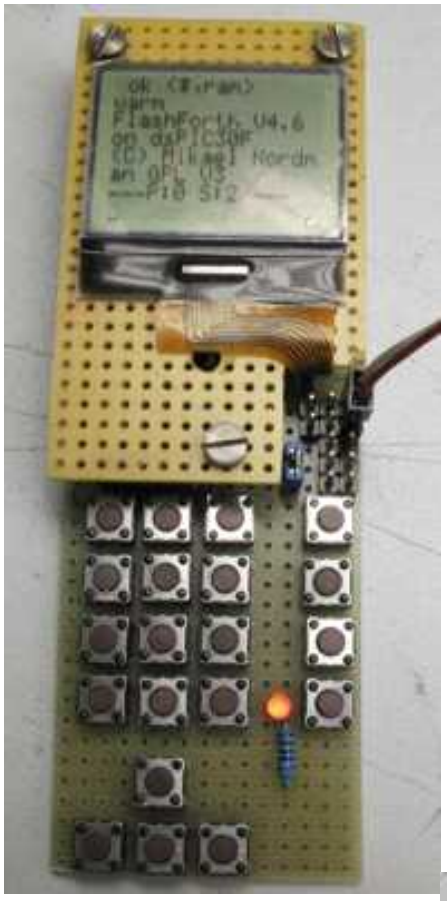
Zobrazí sa hláška, ale zakončená „ESC“. To nás informuje, že je zadaný „turnkey“ a máme 2s na to, aby sme stlačili kláves „Esc“ a zabránili tak vykonávaniu tohto slova. Ak náš čas vyprší, začne sa vykonávať slovo označené ako „turnkey“, čiže LED začne blikať. Teda ak by aj terminál nebol pripojený, po dvoch sekundách sa začne s vykonávaním napísaného programu – teda zariadenie je úplne svojbytným a autonómnym blikačom. Ak potrebujeme v programe niečo meniť, pripojíme terminál, do dvoch sekúnd po resete stlačíme „Esc“ a môžeme program meniť...

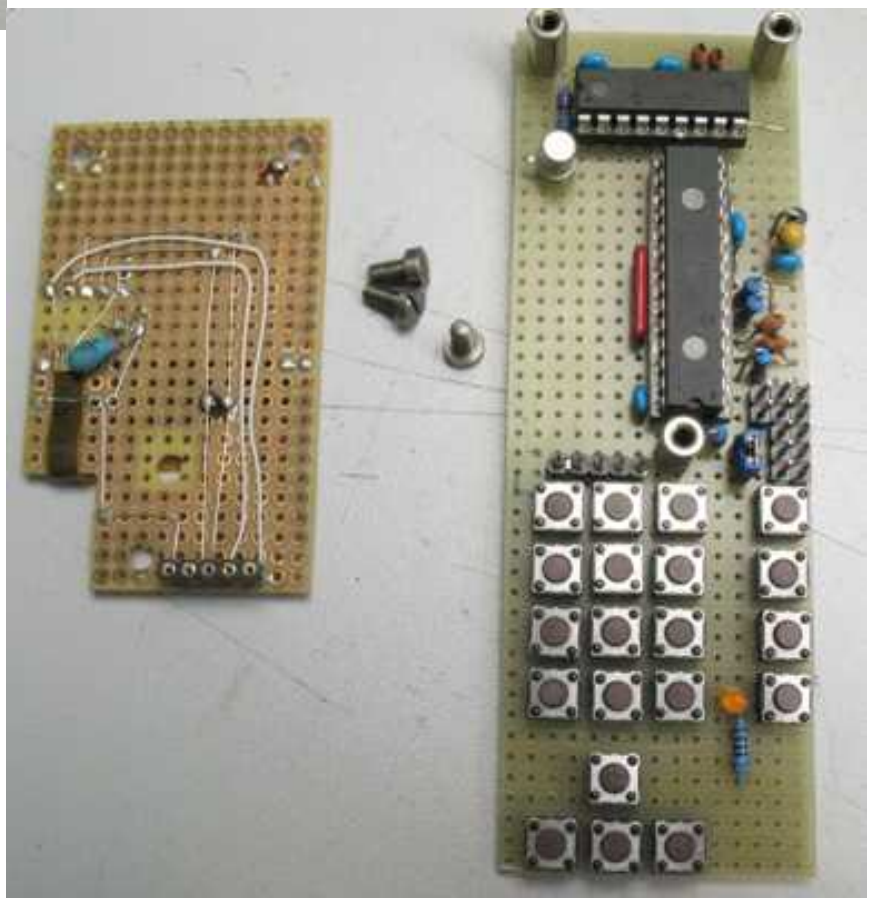
Touto postupnosťou krokov sme si vytvorili najprv najnižšiu programovú vrstvu (ovládanie portu, na ktorom je LED), potom vyššiu (zasvietenie LED na 1s) a nakoniec najvyššiu (blikanie LED), ktoré sme postupne otestovali priamo na cieľovom HW. Tým pádom nebol potrebný nijaký simulátor, resp. debugger. Samozrejme, ani žiadny kompilátor, pretože všetko programové vybavnie sa nachádza na cieľovom PIC. Jediné, čo bolo treba, bol ASCII terminál a s ním, a jedine s ním sme vytvorili klasický program – blikanie LED.

Čo s tým ďalej?

Modus operandi, ktorý som práve popísal, je vlastne plnohodnotným spôsobom písania a ladenia aplikácie, teda skôr býval. Dnes už takýto štýl práce nie je veľmi zaujímavý, hoci aj v súčasnosti si takýto postup dokážem predstaviť, zvlášť pre jednoduché aplikácie, ktoré robia ľudia nie príliš fundovaní v elektronike a programovaní.

Inou možnosťou – a tu sa dostávam k využitiu BASIC-52 alebo FF, ako som to spomínal na začiatku – je využitie takéhoto systému ako hračky pre veľké deti. Ako som spomínal, celý vývojový systém je implementovaný na jednom MCU a jediné, čo k nemu treba, je terminál. Terminál je pomerne hlúpe zariadenie a je relatívne veľkým luxusom používať PC iba na tento účel. No a tu sa vynoril moja stará túžba, ešte z dôb, kedy som s elektronikou len začínal a o mikrokontroléroch som nemal ani poňatie – teda vyrobiť si malé, prenosné programovateľné zariadenie. A z tejto platformy je k splneniu tohto sna pomerne blízko. To sa vezme MCU s FF (alebo BASIC-52) a k nemu sa pripojí iný MCU s displejom a klávesnicou, ktorý sa tvári ako terminál. A môže to vyzerať napríklad takto:





Mať vlastný počítač je nepochybne skutočný statusový symbol každého správneho „geek-a“. Ale ako som na to šiel, o tom až niekedy nabudúce.

Zdroje:

[1] http://mcu.cz/comment.php?comment.news.645
[2] http://en.wikipedia.org/wiki/Interpreted_language
[3] http://www.harbaum.org/till/nanovm/index.shtml
[4] http://wiki.python.org/moin/PyMite
[5] http://en.wikipedia.org/wiki/Forth_%28programming_language%29
[6] http://www.forth.com/starting-forth
[7] http://flashforth.sourceforge.net

Projekt pre MPLAB, ako aj vzorový príklad sú priložené k článku.