MCUmodul-AC2 (update 1.03)

Túto vec som dostal nedávno mailom od Olega Kowalczuka. Ide o verziu klasického MCUmodulu-AC2 (ktorý sme s Olegom upravili z pôvodného MCUmodulu-S52), "fúziovanú" s verziou MCS BASIC–52 (v1.31), ktorý už dávnejšie upravila pre beh na jadrách AT89C51XXX Y. Anastassiová (r.2003). Používala to ako učebný – výukový MCU-kit pre študentov. Oleg Kovalczuk obidva systémy spojil do jedného a vytvoril v celku peknú vecičku. Schému neuvádzam – je identická so zapojením MCUmodulu-AC2 (manuál je vo fóre stránok www.mcontrollers.com, sekcia 8051/8052). Obidva systémy majú oddelenú pamäť pre odzálohovanie vyvíjaného programu aplikácie a navzájom sa neovplyvňujú. Auto-run programu má len MCUmodul-AC2, preto je v úvodnom menu, zobrazenom po zapnutí, ako východiskový.

PRÍLOHA: podmienkové bity

Skrz zlú čitateľnosť textového fontu (prišlo mi pár emailov ohľadom toho) v manuáli MCUmodulu-S52 a niekoľko novým funkciam pre MCUmodul-AC2 som zhrnul všetko ešte raz do jednej tabuľky. Bity, bajty a adresy useflash sú zadávané hexadecimálne, ich názvy sa zobrazujú iba po vylistovaní programu.

bit bajtu, názov (zobrazí iba list),	funkcia / činnosť
17H/bit-0 ON_time_	nastaví MCU, ak je hociktorý spínací čas a deň zhodný s hodinami
17H/bit-1 OFF_time	nastaví MCU ak je hociktorý vypínací čas a deň zhodný s hodinami
17H/bit-2 y/n_time	nastavený = sú akceptované spínacie / vypínacie časy a dni
17H/bit-3 sndALARM	nastaví MCU ak je hociktorý alarmový čas a deň zhodný s hodinami
17H/bit-4 y/nALARM	nastavený = sú akceptované navolené alarmové časy a dni
17H/bit-5 sndALERT	nastaví MCU po uplynutí času vzniku niektorého alertu
17H/bit-6 y/nALRT0	nastavený = port P1.0 spustí odpočítavanie času pre vznik alertu
17H/bit-7 y/nALRT1	nastavený = port P1.1 spustí odpočítavanie času pre vznik alertu
18H/bit-0 y/nALRT2	nastavený = port P1.2 spustí odpočítavanie času pre vznik alertu
18H/bit-1 y/nALRT3	nastavený = port P1.3 spustí odpočítavanie času pre vznik alertu
18H/bit-2 latch_CT	nastavený = neprepisuje obsah počítadla do vyrovnávacej pamäte
18H/bit-3 clear_CT	nastavený = drží obsah počítadla impulzov vynulovaný
18H/bit-4 run_TM0	nastavený = beží časovač 0
18H/bit-5 run_TM1	nastavený = beží časovač 1
18H/bit-6 run_TM2	nastavený = beží časovač 2
18H/bit-7 run_TM3	nastavený = beží časovač 3
19H/bit-0 convdisp	spôsob, akým sa zobrazí na displeji stav A/D prevodu v dekad. sústave:
19H/bit-1 convdisp	bit0 / bit1: 00=x,xx 01=xx,x 10=xxx 11=nepoužitý stav
19H/bit-2 PWMclock	0=f.OSC/6, 1=f.OSC/2 – časovanie pre PWM kanál
19H/bit-3 user_bit	pre individuálne použitie (bity 0-3 neboli v MCUmodule-S52 použité)
19H/bit-4 zero_TM0	nastaví MCU, ak má počas behu nulovú hodnotu časovač 0
19H/bit-5 zero_TM1	nastaví MCU, ak má počas behu nulovú hodnotu časovač 1
19H/bit-6 zero_TM2	nastaví MCU, ak má počas behu nulovú hodnotu časovač 2
19H/bit-7 zero_TM3	nastaví MCU, ak má počas behu nulovú hodnotu časovač 3
1AH/bit-0 P1.0ALRT	nastaví MCU, ak je alert zaznamenaný ako aktívny na vstupe P1.0
1AH/bit-1 P1.1ALRT	nastaví MCU, ak je alert zaznamenaný ako aktívny na vstupe P1.1
1AH/bit-2 P1.2ALRT	nastaví MCU, ak je alert zaznamenaný ako aktívny na vstupe P1.2
1AH/bit-3 P1.3ALRT	nastaví MCU, ak je alert zaznamenaný ako aktívny na vstupe P1.3
1AH/bit-4 i/oPORT0	0=vstupný, 1=výstupný
1AH/bit-5 i/oPORT1	0=vstupný, 1=výstupný
1AH/bit-6 i/oPORT2	0=vstupný, 1=výstupný
1AH/bit-7 FLAG_bit	nastavuje MCU použitím inkrementu, dekrementu, rotácií a porovnaní

PRÍLOHA: useflash rutiny

Sú to programové call rutiny vo FLASH MCU, vykonávajúce časti programu, ktoré by boli buď zdĺhavo realizovateľné inštrukciami MCU modulu, alebo by ich nimi nebolo možné previesť vôbec. Tým, že sa zadáva ich hexa-adresa, (nie názov) je možné vytvoriť ďalšie užívateľské podprogramy vo voľnej časti FLASH procesora (v lokácii 0x5000-0x7FFFH, pre použitie sú: RB0, A, PSW, DPTR0 / DPRT1 a 10 úrovní SP). Starý MCUmodul (r. 2001) osadený procesorom P89C668 vytvorený Y. Anastassiovou ich mal 50. Pre MCUmodul AC-2 sú k dispozícii adresy pre tieto:

adresa, názov (zobrazí list),	funkcia / činnosť
0EF5: sendCLCK	vypíše na terminal hodiny HH:MM:SS
0EF8: sendDATE	vypíše na terminal DEN/DAT.MES
0EB3: dispCLCK	vypíše hodiny HH:MM:SS na display. Pozícia zobrazenia = adr.1BH
0EB6: dispDATE	vypíše dátum DEN/DAT.MES na display. Pozícia zobrazenia = adr.1BH
0EB9: sendCRLF	vyšle CR + LF na terminál
0EBC: jump_TTY	ak je klávesnica terminálu uvolnená, tak návrat bez reakcie. Ak je prijatý serialom nejaký znak, tak data na adr.1Bh sú brané ako nižšia adresa v stránke, z ktorej useflash voláme a prevedie sa skok na túto adresu. Následne sa uloží prijatý znak z terminálu na adr.1Bh. Táto rutina reaguje pomalšie, takže treba podržať dlhšie stlačenú klávesu na počítači (cca 1 sec.), aby terminálový program poslal po sebe opakovane viac znakov.
0EBF: send_TTY	vyšle data z adresy 1Bh cez serial port.
0EC2: waiting_	časová prodleva, závisí od velkosti dát vložených pred tým na adr.1Bh (00h=minimálna, FFh=maximálna - trvá viac ako minútu).
0EC5: jumpKEYB	ak sú tlačidlá klávesničky uvolnené, tak návrat bez reakcie, inak je stav adresy 1Bh braný ako nižšia adresa v stránke z ktorej useflash voláme a prevedie sa skok na túto adresu. Zároveň uloží prijatý znak z klávesnice do adr. 1Bh.
0EC8: waitKEYB	prijme bajt z klávesničky a ulozi na adr.1Bh. Ak je klávesnica nestlačená-uvolnená, čaká.
0ECB: wait TTY	prijme znak z terminalu a uloži na adr.1Bh. Ak nieje žiadna klávesnica PC stlačená, čaká.
0ECE: editCLCK	editor hodín. Pozícia zobrazenia = adr.1BH
0ED1: editDATE	editor dátumu. Pozícia zobrazenia = adr.1BH
0ED4: editPRES	editor nastavení predvolieb časovačov T0 – T3. Pozícia zobrazenia displeja = adr.1BH
0ED7: editTMRS	editor časovačov T0 – T3. Pozícia zobrazenia displeja = adr.1BH
0EDA: edit_ON_	editor spínacích časov + dní. Pozícia zobrazenia displeja = adr.1BH
0EDD: edit_OFF	editor vypínacích časov + dní. Pozícia zobrazenia displeja = adr.1BH
0EE0: editALAR	editor alarmových časov + dní. Pozícia zobrazenia displeja = adr.1BH
0EE3: editDLAY	editor zpoždení liniek alert pre vstupy P1.0 – P1.3 . Zobrazenie na displeji = adr.1BH
0EE6: disp_TM0	zobrazí na displeji stav časovača T0. Pozícia zobrazenia displeja = adr.1BH
0EE9: disp_TM1	zobrazí na displeji stav časovača T1. Pozícia zobrazenia displeja = adr.1BH
0EEC: disp_TM2	zobrazí na displeji stav časovača T2. Pozícia zobrazenia displeja = adr.1BH
0EEF: disp_TM3	zobrazí na displeji stav časovača T3. Pozícia zobrazenia displeja = adr.1BH
0EF2: disp_CT_	zobrazí na displeji stav počítadla CT. Pozícia zobrazenia displeja = adr.1BH
0FA0: A/D_conv	obrazí na displeji A/D prevod. Pozícia zobrazenia displeja = adr.1BH
0FA3: dispconv	zobrazí stav adr.6EH na displeji, ako A/D prevod. Pos. zobr. = adr.1BH
0FA6: PWM_mode	vloží do PWM data z adr.6FH (adresy 6EH a 6FH neboli v MCUmodule-S52 použité)
0FA9: clr_P4.0	nuluje linku portu P4.0 – táto linka môže byť použitá iba ako výstupná
0FAC: set_P4.0	nastaví linku portu P4.0 – táto linka môže byť použitá iba ako výstupná
0FAF: cpl_P4.0	neguje linku portu P4.0 – táto linka môže byť použitá iba ako výstupná
0FB2: clr_P4.1	nuluje linku portu P4.1 – táto linka môže byť použitá iba ako výstupná
0FB5: set_P4.1	nastaví linku portu P4.1 – táto linka môže byť použitá iba ako výstupná
0FB8: cpl_P4.1	neguje linku portu P4.1 – táto linka môže byť použitá iba ako výstupná
0FBB: GENERmod	spustí generátor (zastaví serial port – a naopak) na linke P1.0

PRÍLOHA: popis inštrukcií

Riadia činnosť funkcií MCU a obsluhu "podmienkových" príznakov. Inštrukcie vkladáme terminálom - pod číselným kódom vojdením do sequenceru na jednotlivé adresy pamäte SEEPROM, (x)=počet bajtov, ktoré inšttrukcia spotrebuje.

kód, názov inštrukcie,	funkcia / činnosť
00(?)**disptext xx** abcd…^	zobrazí text na displayi od adr.xx, ukončený symbolom ^ (vkladá ho automaticky)
01(?)**termtext** abcd…^	zobrazí text na terminái, ukončený symbolom ^ (vkladá ho automaticky)
02(3)**dispbyte yy xx**	zobrazí bajt yy na displayi od adr.xx
03(2)**termbyte yy**	zobrazí bajt yy na terminái
04(3)**dispbits yy xx**	zobrazí 8 bitov bajtu yy na displayi od adr.xx
05(2)**termbits yy**	zobrazí 8 bitov bajtu yy na terminái
06(3)**dispchar yy xx**	zobrazí bajt yy jako ascII znak na displayi od adr.xx
07(2)**termchar yy**	zobazí bajt yy jako ascII znak na terminái
08(4)**ediDbyte yy xx zz**	edituje bajt yy na displayi od adr.xx v rozsahu 00-zz dekadicky
09(4)**ediHbyte yy xx zz**	edituje bajt yy na displayi od adr.xx v rozsahu 00-zz hexa
0A(3)**editbits yy xx**	edituje bity bajtu yy na displayi od adr.xx

0B(3)**editchar yy xx**	edituje bajt yy zápisom hexa konštanty a jej ascII znaku na displ.adr.xx
0C(3)**ediDbyte yy zz**	edituje bajt yy terminálom v rozsahu 00-zz dekadicky
0D(3)**ediHbyte yy zz**	edituje bajt yy terminálom v rozsahu 00-zz hexa
0E(2)**editbits yy**	edituje bity bajtu yy terminálom
0F(2)**editchar yy**	edituje bajt yy zápisom hexa konštanty ascII znaku terminálom
10(2)**init_sdk p**	inicializuje **D**ispay/**K**eyboard pre použitie portu Pp
11(2)**modekeyb m**	volba módu editovania užívateľskej klávesnice: klasicky(10 tl), alebo inkrement(2 tl)
12(1)**read_cfg**	načíta užívateľskú konfiguráciu do int.RAM MCU z EEPROM
13(1)**save_cfg**	uloží užívateľskú konfiguráciu z int.RAM MCU do EEPROM
14(3)**clr__bit b yy**	vymaž bit b bajtu yy
15(3)**set__bit b yy**	nastav bit b bajtu yy
16(3)**cpl__bit b yy**	neguj bit b bajtu yy
17(3)**and_byte yy xx**	log.and bajtu yy s bajtom xx
18(3)**orl_byte yy xx**	log.orl bajtu yy s bajtom xx
19(3)**xor_byte yy xx**	log.xor bajtu yy s bajtom xx
1A(2)**not_byte yy**	neguj bajt yy
1B(3)**useflash aaaa**	vykoná ľubovoľný podprogram z FLASH MCU na adr.aaaa
1C(2)**rotlbyte yy**	rotácia bajtu yy vľavo cez príznak: bit 7 -> FLAG-bit -> bitu 0
1D(2)**rotrbyte yy**	rotácia bajtu yy vpravo cez príznak: bit 0 -> FLAG-bit -> bitu 7
1E(3)**inc_byte yy n**	inkrement bajtu yy v hex/dec sústave
1F(3)**dec_byte yy n**	dekrement bajtu yy v hex/dec sústave
20(3)**movebyte yy xx**	presun obsahu bajtu xx do bajtu yy
21(3)**loaddata yy dd**	zápis dát dd do bajtu yy
22(5)**if_0=bit b yy aaaa**	ak je bit bajtu yy b=1, skok na adr.aaaa
23(5)**if_1=bit b yy aaaa**	ak je bit bajtu yy b=0, skok na adr.aaaa
24(5)**if_=byte yy xx aaaa**	ak je bajt yy zhodný s bajtom xx, skok na adr.aaaa
25(5)**if_=data yy dd aaaa**	ak je bajt yy zhodný s dátami dd, skok na adr.aaaa
26(3)**jump_adr aaaa**	skok na adr.aaaa
27(3)**call_adr aaaa**	volanie na adr.aaaa
28(1)**<return>**	návrat z volania
29(1) **----**	prázdna operácia bez významu

(x)= počet bajtov inštrukcie, **b**= bit 0-7, **p**= port 0,1,2, **m**= mód činnosti klávesnice 0-hexa /1-inkrement, **n**= pre inštr.1E a 1F: 1=desiatková / 0=hexa sústava, **dd**= datová konštanta 00-FFH, **xx,yy**= ak ide o adr.bajtu, tak rozsah: 00-FFH* pričom yy je cieľový bajt, ak ide o adr.displaya, napr. CM2016 (2x16 znakov), tak: 1.znak /1.riadok=00H, 2.znak /1.riadok=01H…0FH, a 1.znak / 2.riadok=40H, 2.znak/2.riadok=41H…4FH, **zz**= pre dec. sústavu: 00-99, pre hexa: 00-FFH, **aaaa**= adresa 0000-03FFH

(* - určité obmedzenia, týkajúce sa zákazu prístupu na niektoré RAM adresy, ktoré sú v použití programu procesora)

PRÍLOHA: BASIC-52

Interpreter Basic (modifikovaná verzia 1.31) je navrhnutý pre beh na jadre 8052 s externou RAMkou. Y. Anastassiová ho upravila pre beh v XRAM procesora a uvolnila porty P0 a P2 k použitiu. Vyžaduje minimum 1kB pamäte XRAM, pri ktorej je ale značne zlimitovaný. Beží v tzv. RAM-móde, vypustené (alebo zmenené) sú iba príkazy, ktoré pracovali s externou EPROM (jej napálenie/čítanie) a príkazy pracujúce s prerušeniami (prerušenia využíva iba MCUmodul-AC2 a v tomto použití pre basic moc veľké opodstatnenie nemajú). Použitie BASICu obsahuje kvantum príkazov, preto vážne doporučujem prečítať si original manuál k BASICU-52, kde je činnosť príkazov aj celého interpreteru výborne opísaná. Zmienim sa iba o úpravách, ktoré boli pre použitie BASICu-52 na jednočipovom mikrokontroléri urobené:

XFER: pôvodne slúžil na prenos dát medzi externými pamäťami EPROM a RAM, teraz prenesie program z internej EEPROM do XRAM; tento prenos nastáva aj automaticky – vždy po spustení interpreteru Basic.
PROG: uloží program z XRAM do internej EEPROM procesora.

Zrušené (zbytočné a nutne vypustené) príkazy sú iba tieto: ROM, RROM, PGM a FPROG (ten bol vypustený ešte pôvodnými autormi v tejto verzii interpreteru), a príkazy pracujúce s prerušeniami: CLOCK, ONTIME, ONEX1, RETI a IDLE.

Pribudli ďalšie inštrukcie, ktoré rozširujú možnosti využitia portov P0, P2 a P4 procesora (BASIC-52 ich neobsahoval):

P0: ovláda prístup (zápis, čítanie obsahu) k portu P0
P2: ovláda prístup (zápis, čítanie obsahu) k portu P2

pouzitie v programe - napr:

```
5  P2=200
10 if P0=100 then ...
15 ...
20 ...
30 end
```

Linky portu P4 – P4.0 a P4.1 je možné ovládať v BASICu-52 pomocou jeho inštrukcie „call xxxxh“.

call 0FA9h	nuluje linku portu P4.0 – táto linka môže byť použitá iba ako výstupná
call 0FACh	nastaví linku portu P4.0 – táto linka môže byť použitá iba ako výstupná
call 0FAFh	neguje linku portu P4.0 – táto linka môže byť použitá iba ako výstupná
call 0FB2h	nuluje linku portu P4.1 – táto linka môže byť použitá iba ako výstupná
call 0FB5h	nastaví linku portu P4.1 – táto linka môže byť použitá iba ako výstupná
call 0FB8h	neguje linku portu P4.1 – táto linka môže byť použitá iba ako výstupná

pouzitie v programe - napr:

```
10 call 0FA9h
20 ...
30 ...
40 end
```

Test BASICu-52:

```
[MCS BASIC-52 v1.31]
READY
>print mtop
```

1022 alokátor použiteľnej pamäte XRAM

```
>print free
```

510 voľné bajty použiteľnej XRAM

```
>print xtal
```

12000000 frekvencia kryštálu (logicky – fixne daná ☺)

PRÍLOHA – Ďalšie inštrukcie BASICu-52

BASIC-52 bol vo viacerých verziách – v1.0, v1.1, v1.2, v1.3 a finálna v1.31, preto sa môže použitie niektorých príkazov v závislosti od popisu inštrukcií odlyšovať a nie všetky sú v minimálnom móde behu MCU funkčné.

COMMAND:	FUNCTION:	USAGE EXAMPLE(S):
RUN	Execute a program	RUN
CONT	CONTinue after a STOP or control-C	CONT
LIST	LIST program to the console device LIST	LIST, LIST 10-50
LIST@	LIST program to user driver (version 1.1 only)	LIST@, LIST@ 50
NEW	erase the program stored in RAM	NEW
NULL	set NULL count after carriage return line feed	NULL, NULL 4
RAM	evoke RAM mode, current program in RAM	RAM
BAUD	set baud rate for line printer port	BAUD 1200
CALL	CALL assembly language program	CALL 9000H
CLEAR	CLEAR variables, interrupts and Strings	CLEAR
CLEARS	CLEAR Stacks	CLEARS
DATA	DATA to be read by READ statement	DATA 100
READ	READ data in DATA statement	READ A
RESTORE	RESTORE READ pointer	RESTORE

DIM	allocate memory for arrayed variables	DIM A(20)
DO	set up loop for WHILE or UNTIL	DO
UNTIL	test DO loop condition (loop if false)	UNTIL A= 10
WHILE	test DO loop condition (loop if true)	WHILE A= B
END	terminate program execution	END
FOR-TO-{STEP}	set up FOR-NEXT loop	FOR A= 1 TO 5
NEXT	test FOR-NEXT loop condition	NEXT A
GOSUB	execute subroutine	GOSUB 1000
RETURN	RETURN from subroutine	RETURN
GOTO	GOTO program line number	GOTO 500
ON GOTO	conditional GOTO	ON A GOTO 5,20
ON GOSUB	conditional GOSUB	ON A GOSUB 2,6
IF-THEN-{ELSE}	conditional test	IF A<B THEN A=0
INPUT	INPUT a string or variable	INPUT A
LET	assign a variable or string a value	LET A= 10 (LET is optional)
ONERR	ONERRor GOTO line number	ONERR 1000
PRINT	PRINT variables, strings or literals	PRINT A (P. is shorthand for PRINT)
PRINT#	PRINT to software serial port	PRINT# A
PH0.	PRINT HEX mode with zero suppression	PH0. A
PH1.	PRINT HEX mode with no zero suppression	PH1. A
PH0.	# PH0. to line printer	PH0.# A
PH1.	# PH1. to line printer	PH1.# A
PRINT@	PRINT to user defined driver (version 1.1 only)	PRINT@ 5*5
PH0.@	PH0. to user defined driver (version 1.1 only)	PH0. @ XBY(5EH)
PH1.@	PH1. to user defined driver (version 1.1 only)	PH1.@ A
PUSH	PUSH expressions on argument stack	PUSH 10, A
POP	POP argument stack to variables	POP A, B, C
PWM	PULSE WIDTH MODULATION	PWM 50, 50, 100
REM	REMark	REM DONE
STOP	break program execution	STOP
STRING	allocate memory for STRINGs	STRING 50, 10
UI1	evoke User console Input routine	UI1
UI0	evoke BASIC console Input routine	UI0
UO1	evoke User console Output routine	UO1
UO0	evoke BASIC console Output routine	UO0
ST@	store top of stack at user specified location	ST@ 1000H (version 1.1 only) ST@ A
LD@	load top of stack from user specified location	LD@ 1000H(version 1.1 only) LD@ A
+	ADDITION	1 + 1
/	DIVISION	10 / 2
**	EXPONENTATION	2 * * 4
*	MULTIPLICATION	4 * 4
-	SUBTRACTION	8 - 4
.AND.	LOGICAL AND	10.AND.5
.OR.	LOGICAL OR	2.0R.1
.XOR.	LOGICAL EXCLUSIVE OR	3.XOR.2
ABS( )	ABSOLUTE VALUE	ABS(-3)
NOT( )	ONES COMPLEMENT	NOT(0)
INT( )	INTEGER	INT(3.2)
SGN( )	SIGN	SGN(- 5)
SQR( )	SQUARE ROOT	SQR(100)
RND	RANDOM NUMBER	RND
LOG( )	NATURAL LOG	LOG(10)
EXP( )	"e" (2.7182818) TO THE X	EXP(10)
SIN( )	RETURNS THE SINE OF ARGUMENT	SIN(3.14)
COS( )	RETURNS THE COSINE OF ARGUMENT	COS(0)
TAN( )	RETURNS THE TANGENT OF ARGUMENT	TAN(.707)
ATN( )	RETURNS ARCTANGENT OF ARGUMENT	ATN(1)
CBY( )	READ PROGRAM MEMORY	P. CBY(4000)
DBY( )	READ/ASSIGN INTERNAL DATA	DBY(99)=10
XBY( )	READ/ASSIGN EXTERNAL DATA	P. XBY(10)
GET	READ CONSOLE	P. GET
PORT1	READ/ASSIGN l/O PORT 1 (P1)	PORT1=0FFH

PCON	READ/ASSIGN PCON REGISTER	PCON=0
RCAP2	READ/ASSIGN RCAP2	RCAP2=100
T2CON	READ/ASSIGN T2CON REGISTER	P. T2CON
TCON	READ/ASSIGN TCON REGISTER	TCON=10H
TMOD	READ/ASSIGN TMOD REGISTER	P. TMOD
TIMER0	READ/ASSIGN TIMER0	TIMER0=0
TIMER1	READ/ASSIGN TIMER1	P. TIMER1
TIMER2	READ/ASSIGN TIMER2	TIMER2=0FFH
PI	PI -- 3.1415926	PI