TITLE	ASTE TEST: SERIAL PORTS AND INTERNAL TIMER
DOCUMENT TYPE	REPORT
DOC No.	AMST/AST2/1/A
ISSUE No.	1
DATE	19 April 2006

Prepared by: Fabio Turchet **(CARSO)** .....

Approved by: Paolo Trampus **(CARSO)** .....

CHANGE RECORD

Issue	*Date*	*Affected Pages*	*Description of Change*	*Change Autority*

DISTRIBUTION LIST

Company	***Department***	***Name***	***Quantity***	***Remarks***

TABLE OF CONTENTS

1. HARDWARE USED 5
2. Test 6
3. SOFTWARE 14
   3.1. Flash.ach 14
   3.2. Flash.asm 15
   3.3. Serial.ach 19
   3.4. Serial.asm 20

1. HARDWARE USED

- ASTE BOARD: mounted only DSP, FPGA U31, memories and power supply for DSP and memories, a led with a 5,6k series resistor is connected between FLAG3 output and GND.
- RS232 level converter interface based on MAX233 chip
- Laboratory power supplies for 5V
- 21020 JTAG interface
- PC with ISA JTAG ADAPTER and 21020 software development tools

2. Test

TITLE:	**SERIAL PORTS AND INTERNAL TIMER TEST**				
STEP	**TEST SEQUENCE**	**EXPECTED VALUE**	**MEASURED VALUE**	**REMARKS**	
1	**Installing the software in ASTE board**				
1.1	Connect ASTE to power supply				
1.2	Turn on power supply				
1.3	Connect JTAG interface				
1.4	Assembly serial.asm with the command: C:\ADI_DSP\BIN\ASM_21K.EXE serial				
1.5	Link serial.asm with the command: C:\ADI_DSP\BIN\LD_21K.EXE serial -a serial -m				
1.6	Assembly flash.asm with the command: C:\ADI_DSP\BIN\ASM_21K.EXE flash				
1.7	Link flash.asm with the command: C:\ADI_DSP\BIN\LD_21K.EXE flash -a flash -m				
1.8	Start the simulator wsim020.exe				
1.9	Load the program serial.exe in the simulator				
DATE :		**TEST CONDUCTOR:**			

TITLE:	**SERIAL PORTS AND INTERNAL TIMER TEST**				
STEP	**TEST SEQUENCE**	**EXPECTED VALUE**	**MEASURED VALUE**	**REMARKS**	
1.10	Make a memory dump of program memory from 0x000000 to 0x000200 and save it as 21k.dmp				
1.11	Close the simulator				
1.12	Start the emulator wice020.exe				
1.13	Using the emulator load the program flash.exe in ASTE program memory ram				
1.14	Load the 21k.dmp in program memory starting at 0x800100				
1.15	Load in the Program Counter the value 0x800000 and start the execution				
1.16	The led turns on and the program stops (sector 0 of the flash is erased)	OK			
1.17	Start again the execution				
1.18	The led turns off during programming of sector 0 and then starts blinking	OK			
DATE :		**TEST CONDUCTOR:**			

TITLE:	**SERIAL PORTS AND INTERNAL TIMER TEST**				
STEP	**TEST SEQUENCE**	**EXPECTED VALUE**	**MEASURED VALUE**	**REMARKS**	
1.19	Close the emulator wice020.exe				
1.20	Disconnect JTAG interface				
1.21	Turn off power supply				
1.22	Turn on power supply				
1.23	See if the led is blinking	OK			
DATE :		**TEST CONDUCTOR:**			

TITLE:	**SERIAL PORTS AND INTERNAL TIMER TEST**				
STEP	**TEST SEQUENCE**	**EXPECTED VALUE**	**MEASURED VALUE**	**REMARKS**	
2	**Serial ports test**				
2.1	In order to do this test the test "1 Installing the software in ASTE board" must be already successfully completed				
2.2	Connect the RS232 level converter to serial port of the computer				
2.3	Connect the RS232 level converter to serial port 0 of ASTE BOARD				
2.4	In the computer load the program Hyperteminal (start>program>accessories>HyperTerminal and load Hypertrm.exe)				
2.5	Write the name ASTE and press OK				
2.6	Choose "connect directly to COM1" (or COM2) and press OK				
2.7	Set Bit per second: 19200 Data bits: 8 Parity: none Stopbits: 1 Protocol: none and press OK				
DATE :		**TEST CONDUCTOR:**			

TITLE:	**SERIAL PORTS AND INTERNAL TIMER TEST**				
STEP	**TEST SEQUENCE**	**EXPECTED VALUE**	**MEASURED VALUE**	**REMARKS**	
2.8	Turn on the power supply unit of ASTE board				
2.9	See if the led is blinking	OK			
2.10	Type a random character (from “A” to ”Z” or from “b” to “z”) on the keyboard of the computer				
2.11	See if the typed character appears in hyperterminal	OK			
2.12	Repeat many times step 2.9 and 2.10, the character typed must always appear in hyperterminal	OK			
2.13	Connect the RS232 level converter to serial port 1 of ASTE BOARD				
2.14	Type a random character (from “A” to ”Z” or from “b” to “z”) on the keyboard of the computer				
2.15	See if the typed character appears in hyperterminal	OK			
2.16	Repeat many times step 2.13 and 2.14, the character typed must always appear in hyperterminal	OK			
DATE :		**TEST CONDUCTOR:**			

TITLE:	**SERIAL PORTS AND INTERNAL TIMER TEST**				
STEP	**TEST SEQUENCE**	**EXPECTED VALUE**	**MEASURED VALUE**	**REMARKS**	
3	**ASTE timer test**				
3.1	In order to do this test the test “1 Installing the software in ASTE board” and test “2 serial ports test” must be already successfully completed				
3.2	Connect the RS232 level converter to serial port of the computer				
3.3	Connect the RS232 level converter to serial port 0 of ASTE BOARD				
3.4	In the computer load the program Hyperteminal (start>program>accessories>HyperTerminal and load ASTE.ht)				
3.5	Turn on the power supply unit of ASTE board				
3.6	See if the led is blinking	OK			
3.7	Type the “a” character on the keyboard of the computer				
3.8	See if the output in hyperterminal is the timer value in the format “Txxxxxxxx”, where x is a hex digit	Txxxxxxxx			
3.9	Close the timer reset switch on the ASTE board connector				
DATE :		**TEST CONDUCTOR:**			

TITLE:	**SERIAL PORTS AND INTERNAL TIMER TEST**				
STEP	**TEST SEQUENCE**	**EXPECTED VALUE**	**MEASURED VALUE**	**REMARKS**	
3.10	Type the “a” character on the keyboard of the computer				
3.11	See if the timer value is “T00000000”	T00000000			
3.12	In the computer double click on the clock in the system tray	The DATE/TIME properties window opens, showing the time in hh:mm:ss			
3.13	Open the timer reset switch on the ASTE board connector taking note of the clock time	Time:	9:27:00		
3.14	Wait about 48 hours				
3.15	Type the “a” character on the keyboard of the computer, taking note of the clock time	Time:	10:47:00		
3.16	Read the value of the ASTE timer in hyperterminal	Txxxxxxxx	T0A95F15F		
3.17	Convert the value of the ASTE timer from hex to decimal (the result is in milliseconds)	ASTE timer (ms):	177598815		
3.18	Calculate the time difference between times in 3.13 and 3.15	Time difference:hh:mm:ss	49:20:00		
3.19	Convert the time difference in milliseconds	Time difference (ms):	177600000		
3.20	Calculate the time error as the difference between 3.17 and 3.19	Time error (ms):	1185		
3.21	Check if the error is less than 0,01% (Time error / Time difference)*100	<0.01%	0.000667%		
DATE :		**TEST CONDUCTOR:**			

3. SOFTWARE

3.1. Flash.ach

```
.system scrittura_flash
.processor=ADSP21020;

!******************** Program memory ************************/
!vector table in flash (ora vista come ram per scriverci)
.SEGMENT /RAM  /BEGIN=0x000008 /END=0x00000f /PM rst_svc;
!program code in ram alta
.SEGMENT /RAM  /BEGIN=0x800000 /END=0x800100 /PM pm_code;

!********************   Data memory  ************************/
!data ram a 40 bit
.SEGMENT /RAM  /BEGIN=0x00000000 /END=0x0003ffff /DM dm_bank0;
!data ram a 16 bit
.SEGMENT /RAM  /BEGIN=0x00040000 /END=0x0007ffff /DM dm_bank1;
!ram immagini
.SEGMENT /RAM /BEGIN=0x00080000 /END=0x000bffff /DM dm_bank2;
!porte i/o
.SEGMENT /port /BEGIN=0x000c0000 /END=0x000c00ff /DM dm_bank3;
.ENDSYS;
```

3.2. Flash.asm

```
#include <def21020.h>

#define		source_addr	0x00800100 /*indirizzo di origine in PM ram*/
#define		dest_addr	0x00000000 /*indirizzo di destinazione in PM flash*/
#define		n		32727 /*numero di parole da scrivere 32511*/
/*
.SEGMENT /PM rst_svc;
	jump begin;
.ENDSEG;*/
/* programma principale */
.SEGMENT /PM pm_code;
begin:	nop;
	PMWAIT = b#00000010110001;/* imposta i wait states: 1ws banco 1, 4ws banco0 */
	DMWAIT = b#000011101111010010100101;/* DM 7ws b3, 7ws b2, 1ws b1, 1ws b0 */
	bit set MODE2		FLG3O;	/* imposta flag3 come output */
	bit set ASTAT	FLG3;	/* mette a 1 flag3, accende il led */


/* cancella flash settore 0*/
	r0=0x00aa00aa;		/*scrive aa in 555*/
	px1=r0;
	px2=r0;
	pm(0x555)=px;
	r0=0x00550055;		/*scrive 55 in 2aa*/
	px1=r0;
	px2=r0;
	pm(0x2aa)=px;
	r0=0x00800080;		/*scrive 80 in 555*/
	px1=r0;
	px2=r0;
	pm(0x555)=px;
	r0=0x00aa00aa;		/*scrive aa in 555*/
	px1=r0;
	px2=r0;
	pm(0x555)=px;
	r0=0x00550055;		/*scrive 55 in 2aa*/
	px1=r0;
	px2=r0;
	pm(0x2aa)=px;
	r0=0x00300030;		/*scrive 30 in indirizzo dentro il settore da cancellare*/
	px1=r0;
	px2=r0;
	pm(0x000000)=px;
	idle;	/*ferma il processore per evitare latchup*/
/* controllo fine cancellazione*/
cDQ7_0:		px=pm(0x000000);	/*controllo DQ7_0*/
```

```
        r0=px1;
        btst r0 by 7;
        if sz jump cDQ5_0;
        jump   cDQ7_1; /*DQ7_0 vale 1*/
cDQ5_0:     btst r0 by 5;
        if sz jump cDQ7_0;
        px=pm(0x000000);   /*DQ5_0 vale 1, ricontrollo DQ7_0*/
        r0=px1;
        btst r0 by 7;
        if not sz jump cDQ7_1;      /*DQ7_0 vale 1*/
        idle; /*errore!!!!!!!!!!!!!!!!*/
cDQ7_1:     px=pm(0x000000);   /*controllo DQ7_1*/
        r1=px2;
        btst r1 by 7;
        if sz jump cDQ5_1;
        jump   cDQ7_2; /*DQ7_1 vale 1*/
cDQ5_1:     btst r1 by 5;
        if sz jump cDQ7_1;
        px=pm(0x000000);   /*DQ5_1 vale 1, ricontrollo DQ7_1*/
        r1=px2;
        btst r1 by 7;
        if not sz jump cDQ7_2;      /*DQ7_1 vale 1*/
        idle; /*errore!!!!!!!!!!!!!!!!*/
cDQ7_2:     px=pm(0x000000);   /*controllo DQ7_2*/
        r1=px2;
        btst r1 by 23;
        if sz jump cDQ5_2;
        jump   c_fine; /*DQ7_2 vale 1*/
cDQ5_2:     btst r1 by 21;
        if sz jump cDQ7_2;
        px=pm(0x000000);   /*DQ5_2 vale 1, ricontrollo DQ7_2*/
        r1=px2;
        btst r1 by 23;
        idle;
        if not sz jump c_fine; /*DQ7_2 vale 1*/
        idle; /*errore!!!!!!!!!!!!!!!!*/

c_fine: bit clr ASTAT   FLG3;     /* mette a 0 flag3, spegne il led */

/* scrittura dati */
        b8=source_addr;
        b9=dest_addr;
        l8=0;   l9=0;
        m8=1; m9=-1;

        r0=0x00aa00aa;      /*scrive aa in 555, unlock bypass*/
        px1=r0;
        px2=r0;
        pm(0x555)=px;
```

```
        r0=0x00550055;      /*scrive 55 in 2aa*/
        px1=r0;
        px2=r0;
        pm(0x2aa)=px;
        r0=0x00200020;      /*scrive 20 in 555*/
        px1=r0;
        px2=r0;
        pm(0x555)=px;

        lcntr=n, do scrivi until lce;
                r0=0x00A000A0;      /*scrive A0 in 000000, unlock bypass program*/
                px1=r0;
                px2=r0;
                pm(0x000000)=px;
                px=pm(i8,m8); /*legge dato dalla ram*/
                r0=px1;
                r1=px2;
                pm(i9,m8)=px;/*scrive dato in indirizzo*/
fcontr:         px=pm(m9,i9); /*legge dato dalla flash*/
                r2=px1;
                r3=px2;
                comp(r0,r2);
                if ne jump fcontr;
                comp(r1,r3);
                if ne jump fcontr;
                nop;
                nop;
                nop;
scrivi:         nop;

        r0=0x00900090;      /*scrive 90 in 000000, unlock bypass reset*/
        px1=r0;
        px2=r0;
        pm(0x000000)=px;
        r0=0x00000000;      /*scrive 00 in 000000*/
        px1=r0;
        px2=r0;
        pm(0x000000)=px;

/*fine scrittura, lampeggio led*/
led:    bit set ASTAT   FLG3;      /* mette a 1 flag3, led acceso  */
        lcntr=4000000; do rit1 until lce;
rit1:   nop;
        bit clr ASTAT   FLG3;      /* mette a 0 flag3, led spento  */
        lcntr=4000000; do rit2 until lce;
rit2:   nop;
        jump   led;

done:   idle;
```

.ENDSEG;

3.3. Serial.ach

```
.system scrittura_flash
.processor=ADSP21020;

!******************** Program memory ************************/
!vector table in flash
.SEGMENT /ROM  /BEGIN=0x000008 /END=0x00000f /PM rst_svc;

!program code in flash
.SEGMENT /ROM  /BEGIN=0x000100 /END=0x000200 /PM pm_code;

!********************   Data memory  ************************/
!data ram a 40 bit
.SEGMENT /RAM  /BEGIN=0x00000000 /END=0x0003ffff /DM dm_bank0;
!data ram a 16 bit
.SEGMENT /RAM  /BEGIN=0x00040000 /END=0x0007ffff /DM dm_bank1;
!ram immagini
.SEGMENT /RAM /BEGIN=0x00080000 /END=0x000bffff /DM dm_bank2;
!porte i/o
.SEGMENT /port /BEGIN=0x000c0000 /END=0x000c00ff /DM dm_bank3;

.ENDSYS;
```

3.4. Serial.asm

```
#include <def21020.h>

#define		ser0_data	0x80000000 /*indirizzo dati seriale 0 in DM*/
#define		ser0_reg	0x80000005 /*indirizzo reg seriale 0 in DM*/
#define		ser1_data	0x80000010 /*indirizzo dati seriale 1 in DM*/
#define		ser1_reg	0x80000015 /*indirizzo reg seriale 1 in DM*/
#define		timerlow	0x80000020 /*indirizzo timerlow in DM*/
#define		timerhigh	0x80000021 /*indirizzo timerhigh in DM*/

.SEGMENT /PM rst_svc;
	jump begin;
.ENDSEG;

/* programma principale */
.SEGMENT /PM pm_code;
begin:	PMWAIT = b#00000010101101;/* imposta i wait states: 1ws banco 1, 3ws banco0 */
	DMWAIT = b#000000101111010010100101;/* DM 1ws b3, 7ws b2, 1ws b1, 1ws b0 */
	bit set MODE2		FLG2O;	/* imposta flag2 come output */
	bit clr ASTAT	FLG2;	/* mette a 0 flag2 per il multiplexer */
	bit set MODE2		FLG3O;	/* imposta flag3 come output */
	bit clr ASTAT	FLG3;	/* mette a 0 flag3, spegne il led */

/*lampeggio led*/
led:	bit set ASTAT	FLG3;	/* mette a 1 flag3, led acceso  */
	nop;
	nop;
	nop;
	lcntr=4, do ria until lce;
ria:	nop;
	nop;
	nop;
	r0=dm(ser0_reg); /*leggo registro di stato*/
	btst r0 by 0;
	if sz jump ser1;
	nop;
	nop;
	nop;
	lcntr=200, do ri0 until lce;
ri0:	nop;
	nop;
	nop;
	r0=dm(ser0_data); /*leggo dato*/
	r1=0x61;
	comp(r0,r1);
	if ne jump dato; /*se ="a" scrive il tempo, altrimenti il dato*/
	r0=0x54;
	dm(ser0_data)=r0;
```

```
        r3=dm(timerhigh); /*legge il timer*/
        r4=dm(timerlow);
car1:   nop;
        nop;
        nop;
        lcntr=200, do ri1 until lce;
ri1:    nop;
        r0=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump car1;
        r0=FEXT r3 BY 12:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser0_data)=r0;
car2:   nop;
        nop;
        nop;
        lcntr=200, do ri2 until lce;
ri2:    nop;
        r0=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump car2;
        r0=FEXT r3 BY 8:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser0_data)=r0;
car3:   nop;
        nop;
        nop;
        lcntr=200, do ri3 until lce;
ri3:    nop;
        r0=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump car3;
        r0=FEXT r3 BY 4:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
```

```
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser0_data)=r0;
car4:   nop;
        nop;
        nop;
        lcntr=200, do ri4 until lce;
ri4:    nop;
        r0=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump car4;
        r0=FEXT r3 BY 0:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser0_data)=r0;
car5:   nop;
        nop;
        nop;
        lcntr=200, do ri5 until lce;
ri5:    nop;
        r0=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump car5;
        r0=FEXT r4 BY 12:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser0_data)=r0;
car6:   nop;
        nop;
        nop;
        lcntr=200, do ri6 until lce;
ri6:    nop;
        r0=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump car6;
        r0=FEXT r4 BY 8:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
```

```
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser0_data)=r0;
car7:   nop;
        nop;
        nop;
        lcntr=200, do ri7 until lce;
ri7:    nop;
        r0=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump car7;
        r0=FEXT r4 BY 4:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser0_data)=r0;
car8:   nop;
        nop;
        nop;
        lcntr=200, do ri8 until lce;
ri8:    nop;
        r0=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump car8;
        r0=FEXT r4 BY 0:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser0_data)=r0;
        nop;
        jump fines;
        nop;
        nop;
        nop;
dato:   r1=dm(ser0_reg); /*leggo registro di stato*/
        nop;
        btst r1 by 5;
        if sz jump dato;
```

```
        dm(ser0_data)=r0; /*trasmetto dato*/
        nop;
        nop;
        jump fines;
        nop;
        nop;

ser1:   nop;
        r0=dm(ser1_reg); /*leggo registro di stato*/
        btst r0 by 0;
        if sz jump fine;
        nop;
        nop;
        lcntr=200, do ari0 until lce;
ari0:   nop;
        nop;
        nop;
        r0=dm(ser1_data); /*leggo dato*/
        r1=0x61;
        comp(r0,r1);
        if ne jump adato; /*se ="a" scrive il tempo, altrimenti il dato*/
        r0=0x54;
        dm(ser1_data)=r0;
        r3=dm(timerhigh); /*legge il timer*/
        r4=dm(timerlow);
acar1:  nop;
        nop;
        nop;
        lcntr=200, do ari1 until lce;
ari1:   nop;
        r0=dm(ser1_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump acar1;
        r0=FEXT r3 BY 12:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser1_data)=r0;
acar2:  nop;
        nop;
        nop;
        lcntr=200, do ari2 until lce;
ari2:   nop;
        r0=dm(ser1_reg); /*leggo registro di stato*/
        nop;
```

```
        btst r0 by 5;
        if sz jump acar2;
        r0=FEXT r3 BY 8:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser1_data)=r0;
acar3:  nop;
        nop;
        nop;
        lcntr=200, do ari3 until lce;
ari3:   nop;
        r0=dm(ser1_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump acar3;
        r0=FEXT r3 BY 4:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser1_data)=r0;
acar4:  nop;
        nop;
        nop;
        lcntr=200, do ari4 until lce;
ari4:   nop;
        r0=dm(ser1_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump acar4;
        r0=FEXT r3 BY 0:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser1_data)=r0;
acar5:  nop;
        nop;
        nop;
        lcntr=200, do ari5 until lce;
ari5:   nop;
```

```
        r0=dm(ser1_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump acar5;
        r0=FEXT r4 BY 12:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser1_data)=r0;
acar6:  nop;
        nop;
        nop;
        lcntr=200, do ari6 until lce;
ari6:   nop;
        r0=dm(ser1_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump acar6;
        r0=FEXT r4 BY 8:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser1_data)=r0;
acar7:  nop;
        nop;
        nop;
        lcntr=200, do ari7 until lce;
ari7:   nop;
        r0=dm(ser1_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump acar7;
        r0=FEXT r4 BY 4:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser1_data)=r0;
acar8:  nop;
        nop;
        nop;
```

```
        lcntr=200, do ari8 until lce;
ari8:   nop;
        r0=dm(ser1_reg); /*leggo registro di stato*/
        nop;
        btst r0 by 5;
        if sz jump acar8;
        r0=FEXT r4 BY 0:4; /*estrae il nibble*/
        r1=0x30; /*carattere "0"*/
        r0=r1+r0;
        r1=0x39;
        r2=0x7;
        comp(r1,r0);
        if lt r0=r0+r2;
        dm(ser1_data)=r0;
        nop;
        jump fines;
        nop;
        nop;
        nop;
adato:  nop;
        nop;
        r1=dm(ser1_reg); /*leggo registro di stato*/
        nop;
        btst r1 by 5;
        if sz jump adato;
        dm(ser1_data)=r0; /*trasmetto dato*/

fines:  nop;
        nop;
        nop;
        lcntr=200, do ridf until lce;
ridf:   nop;
        nop;
        nop;
        nop;
fine:   nop;
        nop;
        nop;
        jump    led;

done:   idle;
.ENDSEG;
```