Feature Phone based on TMS320LC203

Literature Number: BPRA050
Texas Instruments Europe
March 1997

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

List of Figures

List of Tables

Feature Phone based on TMS320LC203

ABSTRACT

This application report describes a feature phone design based on the TMS320LC203. The LC203 is a 3V member of the TMS320C2xx DSP family combining low cost and low power consumption with a computing power of 20 MIPS. Three independent onchip ports, UART, Serial Port and Parallel Port, allow designing with minimal external logic saving cost, board space and assuring fast access times.

This report discusses mainly the realization of an evaluation board for testing typical pay-phone algorithms such as signaling, speech processing and data transmission. Examples are given how to handle the different peripherals of the DSP regarding HW and SW issues. Basic initialization and test routines together with the board schematics complete this application description.

1. Introduction

In recent years, semiconductor manufacturers have succeeded in increasing the computing power of Digital Signal Processors up to 2 billion instructions per second. Such fast processing is not needed in every application.

For feature phones, price per unit and power consumption are the most important criteria.

The 3-V DSP TMS320LC203 satisfies both (low power and low cost), while its computing power of 20 million instructions per second is more than sufficient for these types of applications.

This report is based on an evaluation board developed for test and emulation of typical payphone algorithms such as signaling, speech processing and data transmission between the central office and the payphone.

This paper will mainly discuss the realization of the hardware, giving examples of how to connect and initialize each port of the DSP. The schematics are included in the appendix.

Chapter 3 gives a short description of the software that has been used for the testing and initialization of the DSP’s interfaces. The source code also is included in the appendix.

2. Hardware

2.1 System Overview

For any feature phone application, multiple interfaces to peripheral devices are needed. The TMS320C2xx family provides up to four totally independent ports. Therefore, even complex systems can be realized with a minimum of external logic - saving costs, board space and ensuring fast access times.

The TMS320LC203 used in this design provides an on-chip UART for communication with a host processor, a 16-bit wide parallel port and a single synchronous serial port.

The parallel port is used to interface external memory to the DSP. The serial port handles the data traffic between DSP and analog interfaces. In this application the serial port is multiplexed between the two CODECs necessary for the line and handset interfaces.

The JTAG port provides direct access to the DSP and optimizes test and emulation capabilities of the system. In addition a seven-segment LED display gives the designer the opportunity to monitor the status of the DSP.

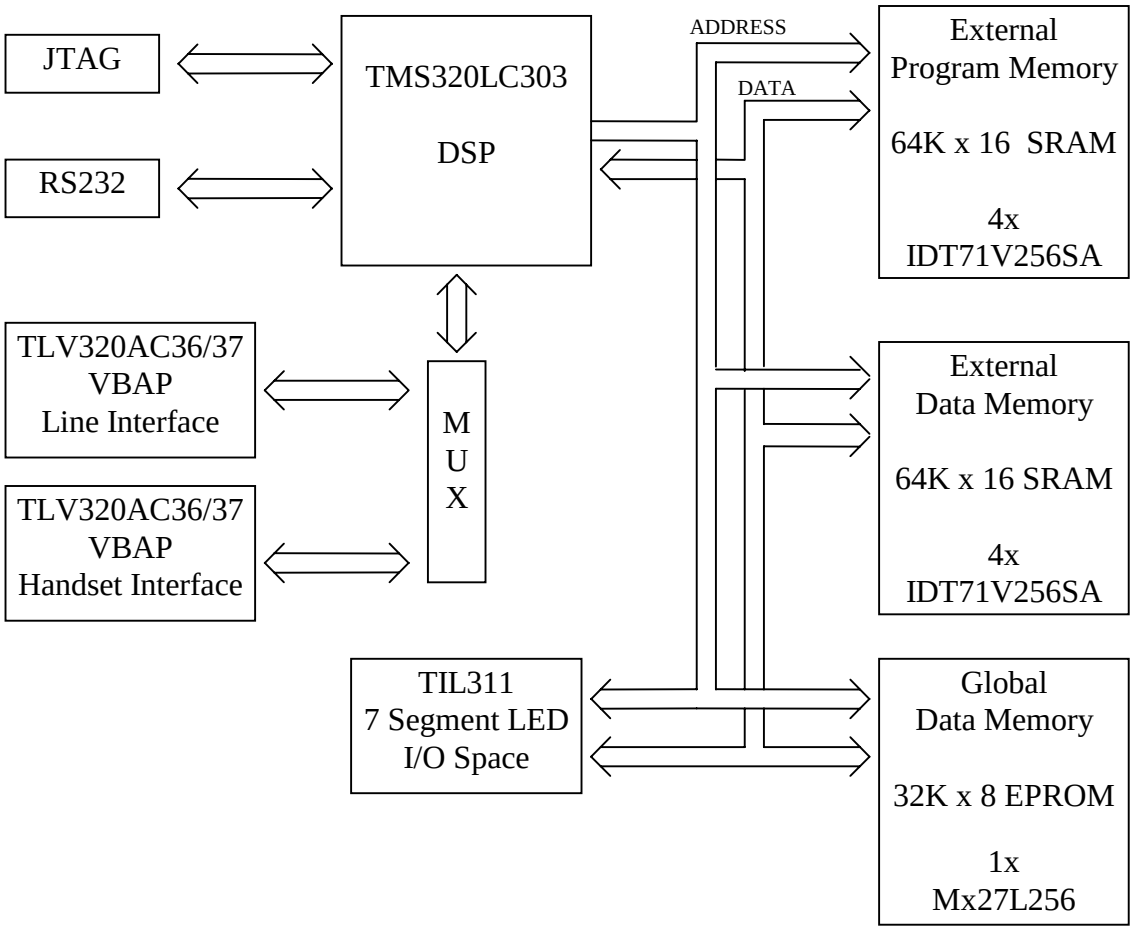


Figure 1: System Overview

2.2 Connections to the Parallel Port

The total space accessible from the DSP by the parallel port is divided in ‘Program’, ‘Data’, ‘Global Data’ and ‘I/O’. With each access to external devices the ‘LC203 selects one of these spaces by driving the appropriate memory select strobes (PS\, DS\, IS\ and BR\) low:

Table 1: Status of memory select signals

Enabled Space	PS\	DS\	IS\	BR\
Program	Low	High	High	High
Data	High	Low	High	High
Global Data	High	Low	High	Low
I/O	High	High	Low	High

The address and data lines are shared by all external spaces.

The four SRAM devices U12 to U15 (see Appendix C p.28) build the external program memory. PS\ combined with A15 gives the appropriate chip select signals for the upper (addresses 0x8000 to 0xffff) and lower (addresses 0x0000 to 0x7fff) memory banks.

WE\ is connected to the write enable input of the SRAMs.

The EPROM U16 is placed in the global data memory. It is accessed only during the boot load process. Once the program is transferred onto the faster SRAM in program space the GREG register will be changed to enable the full range of data memory. BR\ is connected to the chip enable of the EPROM.

Because DS\ is low during global data accesses and data memory accesses, it is not possible to use DS\ alone as chip enable for the SRAMs providing the external Data memory (U8, U9, U10 and U11). Bus conflicts would occur during the access to the EPROM.

Therefore, DS\ combined with BR\ and A15 constitute the chip select for data memory, A15 being the selection signal for upper and lower memory banks.

The 7-segment LED display, U17, is the only external component in the IO space of the DSP. IS\ is connected directly to the strobe input of the LED.

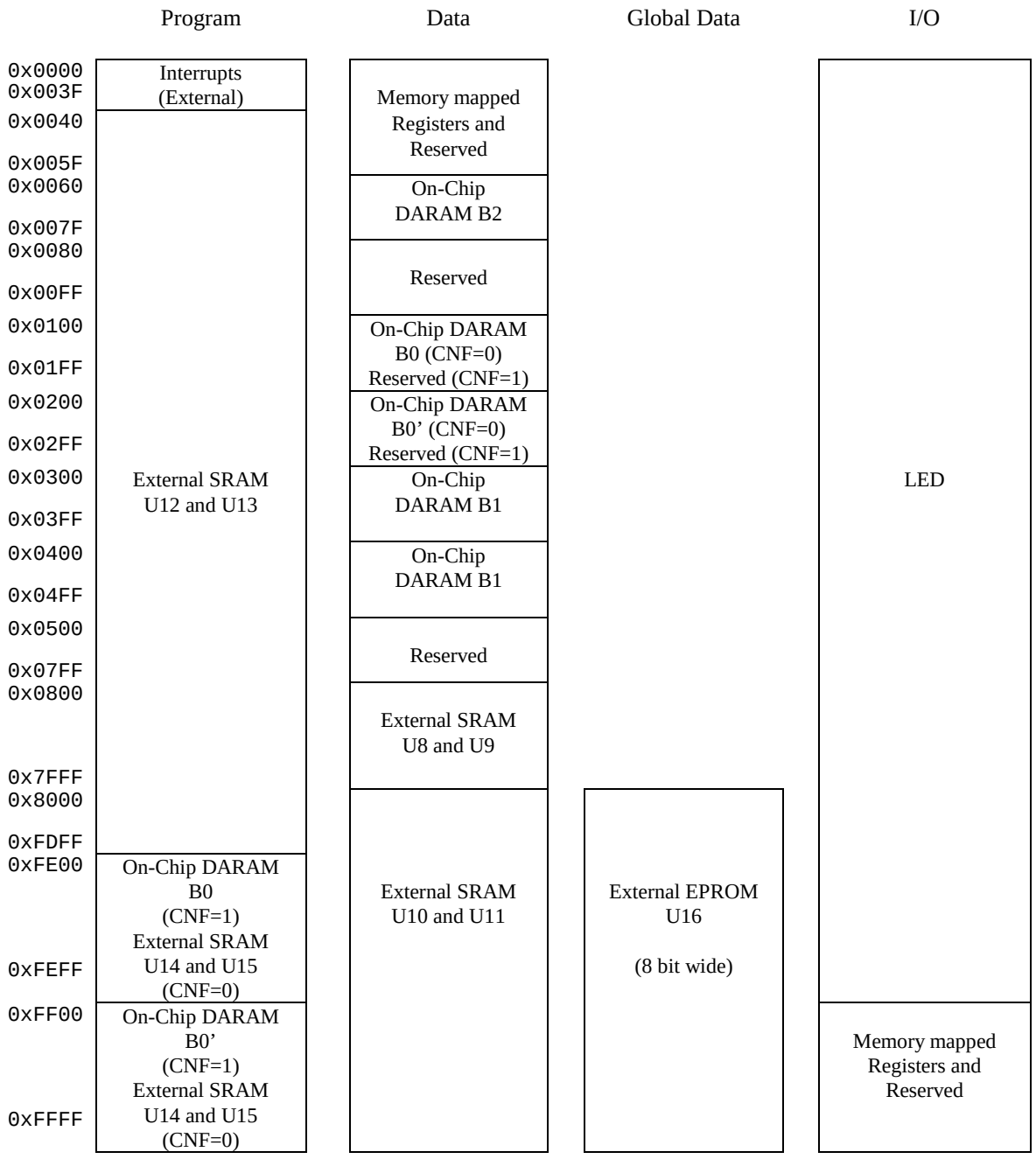


Figure 2: System Memory Map

2.3 On-Chip UART

The TMS320LC203 includes a complete UART on chip. This application uses only the TX and RX pin for a direct connection to a host via a RS232 interface. The handshake signals I/O0 to I/O4 are connected to a header for general purposes.

2.4 Multiplexing the Serial Port

Feature phone applications need two independent analog channels. One for the line interface and one for the handset. As the ‘C203 provides only one synchronous serial port it must be multiplexed between the two VBAPs (Voice Band Audio Processors).

The master clock frequency for the VBAPs is generated by the binary ripple counter SN74HC4060 (U6, Appendix C, p.29). As this frequency defines the internal filters of the TLV320AC36/37 it must be exactly at 2.048MHz. This frequency is also used to clock the data in and out of the serial port. The frame sync pulse FSX for the transmit channel is generated by the DSP. The same pulse triggers the receive channel (FSR).

The frame sync for the VBAPs is multiplexed with the XF output of the DSP. The PAL device, U18, combines FSX and XF with the following equations:

For the line interface:

FSL = FSX & XF

For the handset:

FSH = FSX & !XF

The generated frame syncs trigger the receive and transmit operations of the serial port, at the same time toggling the selection of the VBAPs.

FSX must provide the necessary 8kHz sampling frequency for the VBAPs. The QI (16kHz) of the binary counter (U6) is connected to the DSP via interrupt INT2\.

Each time the DSP receives this interrupt it writes the next value to the transmit register and triggers the FSX pulse. If XF is toggled with every INT2 each VBAP receives the frame sync pulses with a frequency of 8kHz which ensures the proper operation of the devices.

Due to the very efficient interrupt handling of the TMS320C203, the times between two frame sync pulses do not differ significantly, but it must be ensured that no interrupts with higher priority than INT2 occur while the VBAPs are active.

2.5 Variable System Clock

The clock of the DSP is generated by a programmable oscillator. The inputs A, B and C (See Appendix C, p.27) vary the output frequency on pin #2 as shown in table 2. Pin #1 always outputs the specified frequency of the device.

Table 2: Variable frequency for CLKIN

Inputs				Outputs	
C	B	A		D	F
SW4	SW3	GND		pin #2	pin #1
L	L	L	F/2	10MHz	20MHz
L	L	H	F/4	(5MHz)	20MHz
L	H	L	F/8	2.5MHz	20MHz
L	H	H	F/16	(1.25MHz)	20MHz
H	L	L	F/32	625kHz	20MHz
H	L	H	F/64	(312.5kHz)	20MHz
H	H	L	F/128	156.25kHz	20MHz
H	H	H	F/256	(78.125kHz)	20MHz

The pins DIV1 and DIV2 of the DSP specify the factor between the frequency on CLKIN and the DSP clock.

Table 3: TMS320C203 clock options

Parameter	DIV2 SW5	DIV1 SW6
Internal divided by two with external crystal	L	L
PLL multiply by one	L	H
PLL multiply by two	H	L
PLL multiply by four	H	H

3. Software: Initialization & Test

The following chapter describes the programs for initialization and test routines written during the development of the board. The source code can be found in Appendix A of this report.

3.1 Communication between DSP and PC via RS232

Windows '*Terminal*' software is used to establish communication between the DSP and the PC. The keyboard and monitor are used as input and output devices respectively . The DSP transfers its data to the PC via the RS232 interface driven by the on-chip UART.

The exchange of data between the DSP and the PC is controlled by polling the DR and THRE bits of the IOSR registers of the on-chip UART. When the DR bit is set, it indicates that a character has been received. Similarly, when the THRE bit is set, it indicates that the transmit register is empty (i.e. a character has been transmitted).

When receiving data, the function ISRXREADY() is called, waiting until a character is received before attempting to read the ASDTR register in the UART.

When transmitting data, the function ISTXREADY() is called, and waits until ASDTR register is empty before attempting to write to it.

3.2 Serial Port and TLV320AC36

The two TLV320AC36 are connected to the DSP via the serial port. The XF signal is used to switch between the two CODECS. Only the received serial port interrupt is used since the transmit and receive functions of the serial port are synchronized by tying together the FSX and FSR pins.

This paragraph describes the operation of the serial port in our test program. An external interrupt, INT2, is received every 62.5µs (16kHz). The interrupt service routine for INT2 turns on one of the VBAPs by toggling the XF pin and outputs an old sample stored in BUFFER. Since the transmit and receive blocks of the serial port are synchronized, a new sample will arrive at the serial port generating a receive interrupt. The interrupt service routine will read the new sample from the serial port and store it in BUFFER.

To perform the serial port test, an audio source may be connected to ANALOG IN. If speakers are connected to ANALOG OUT they should reproduce the same waveform as that applied to ANALOG IN.

3.3 Testing the external SRAM

In this test program the DSP reads and writes to external memory. Each stored value is checked for errors and any faults encountered increment the error counters.

The program starts by initializing a data word to 0xffffh. This data memory is copied into program memory, and immediately copied back from program to data memory. This procedure is repeated for all the external program and data SRAMS.

Next, the program starts a different test. The DSP executes 8 consecutive writes to the external data space. The data written alternates between 0x5555 and 0xaaaa with

each cycle. The DSP reads out this data and stores the sum of the eight values in the accumulator to check it. This procedure is repeated until the DSP reaches the end of the external memory space. The program executes the same routine with 0x0000 and 0xffff as values to be written.

References

1. Voice-Band -Audio -Processor Application Report, G. Davis and R. MacDonald, Texas Instruments 1994
2. TLV320AC36/37 3V Voice-Band Audio Processor, Texas Instruments 1994, SLWS006
3. TMS320C2xx data sheet, Texas Instruments 1995, SPRS025
4. TMS320C2xx User's Guide, Texas Instruments 1995, SPRU127B
5. Power Supply Circuits, Texas Instruments 1996, SLVD002
6. Digital Design Workshop, E. Haseloff and P. Forstner, Texas Instruments 1995

Appendix A. Source Code

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;     Memory Test for LC203 Board                                ;
;     File Name: memtst.asm                                      ;
;                                                                ;
;     Description:  This program is used to check the correct    ;
;     operation of the SRAMs in the board. Different values      ;
;     are written to Data memory, then transferred to Program    ;
;     memory and finally checked to see if any errors have       ;
;     occurred. The ER_COUNT variable holds the number           ;
;     of times a faulty memory cell has been found.              ;
;     The file Make.bat will assemble and link memtst.asm        ;
;     creating the memtst.out                                    ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

IMR     .set    4h                  ; Int Mask register
GREG    .set    5h                  ; Global Mask register
IFR     .set    6h                  ; Int Flag register
WSGR    .set    0fffch              ; Wait State Generator reg
Ex_dat  .set    0800h               ; External data address
Ex_prg  .set    0600h               ; External prog address
memsize .set    0f000h              ; memory size to be tested

TEMP_VAL .usect   "temp",3          ; Temporary value
ER_COUNT .set   TEMP_VAL+1          ; Error counter
TEST_VAL .set   TEMP_VAL+2          ; test value
ADDR     .set   TEMP_VAL+3          ; current program space address
tstdat  .usect  "test_d",memsize
tstprg  .usect  "test_p",memsize

        .text
        NOP
INIT    ldp     #0                   ;
        splk    #2E00h,TEMP_VAL      ; ARP=1,OVM=1,INTM=1,DP=0
        lst     #0,TEMP_VAL          ; Init ST0
        splk    #21FCh,TEMP_VAL      ; ARB=1,CNF=0,SXM=0,XF=1,PM=0
        lst     #1,TEMP_VAL          ; Init ST1
        splk    #0,IMR               ; No Interrupts  selected
        splk    #0,GREG              ; Disable Global Memory
        splk    #0E00h,TEMP_VAL      ; Init WSGR
        out     TEMP_VAL,WSGR        ; 7 WS I/0, 0 WS Data, 0 WS Prog
        splk    #0ffffh,TEST_VAL     ; Initial Test Value ffffh
        splk    #0,ER_COUNT          ; Initialised number of errors
        OUT     ER_COUNT, 0001h      ; write error counter to LED
```

```
START   lar     ar3,#memsize          ; Initialised loop counter
        larp    ar0                   ; ar0 active pointer
        lar     ar0,#Ex_dat           ; ar0=Start External Data memory
        lar     ar1,#Ex_prg           ; ar1=Start External Prog memory
LOOP    lacl    TEST_VAL              ; load Acc with test value
        sacl    *                     ; store value in Data mem
        sar     ar1,TEMP_VAL          ; load ar1 into TEMP_VAL
        lacl    TEMP_VAL              ; store TEMP_VAL into acc to perform
                 ; transfers
        tblw    *                     ; transfer from data to prog
        tblr    *                     ; tranfer from prog to data
        lacl    TEST_VAL              ; Load Acc with test value
        sub     *+,ar1                ; Test the prog memory
        mar     *+,ar0
        bcnd    COUNT,NEQ             ; If error occur update error count

CONT
        mar     *,ar3                 ; make counter active
        banz    LOOP,ar0              ; check if counter is zero
        NOP
        NOP
;Start test for 8 consecutive reads and 8 consecutive writes

        LACC    #memsize, 13          ; load memorisize devided by 8
        SACH    TEMP_VAL              ; in AR3 as loop counter
        LAR     AR3, TEMP_VAL
        LAR     AR4, #Ex_dat          ; load start address of data memory
        MAR     *,AR4                 ; activate AR4
CONS1:  LACC    #05555h               ; load 0x5555aaaa in accumulator
        SACL    TEMP_VAL
        LACC    TEMP_VAL, 16
        OR      #0aaaah
        SACL    *+                    ; write 0xaaaa
        SACH    *+                    ; write 0x5555
        SACL    *+                    ; write 0xaaaa
        SACH    *+                    ; write 0x5555
        SACL    *+                    ; write 0xaaaa
        SACH    *+                    ; write 0x5555
        SACL    *+                    ; write 0xaaaa
        SACH    *+                    ; write 0x5555
        SAR     AR4,ADDR              ; reset AR4 to the address of first
        LACL    ADDR                  ; value written in this loop
        SUB     #8
        SACL    ADDR
        LAR     AR4, ADDR
        LACC    *+                    ; read and accumulate all values
        ADD     *+                    ; written before
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     #4                    ; the sum + 4 should give 0x40000
        AND     #0ffffh               ; check is lower 16bit are zero
        BCND    ERR_CONS1, NEQ        ; if not increment error counter
C1:     MAR     *, AR3                ; decrement loop counter and repeat
        BANZ    CONS1,*-,AR4          ; the test for the whole data memory

        LACC    #memsize, 13          ; same as above with 0xffff and
        SACH    TEMP_VAL              ; 0x0000 as values to be written
        LAR     AR3, TEMP_VAL
        LAR     AR4, #Ex_dat
        MAR     *,AR4
CONS2:  LACC    #0ffffh
        SACL    TEMP_VAL
        LACC    TEMP_VAL, 16
```

```
        SACL    *+
        SACH    *+
        SACL    *+
        SACH    *+
        SACL    *+
        SACH    *+
        SACL    *+
        SACH    *+
        SAR     AR4,ADDR
        LACL    ADDR
        SUB     #8
        SACL    ADDR
        LAR     AR4, ADDR
        LACC    *+
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     *+
        ADD     #4
        AND     #0ffffh
        BCND    ERR_CONS2, NEQ
C2:     MAR     *, AR3
        BANZ    CONS2,*-,AR4
        B       START                   ; repeat the whole test


ERR_CONS1 lacl    ER_COUNT              ; Load counter in acc
        add     #1                      ; add 1 to acc
        sacl    ER_COUNT                ; store back in counter
        OUT     ER_COUNT, 0001h         ; write error counter to LED
        B         C1
ERR_CONS2 lacl    ER_COUNT              ; Load counter in acc
        add     #1                      ; add 1 to acc
        sacl    ER_COUNT                ; store back in counter
        OUT     ER_COUNT, 0001h         ; write error counter to LED
        B         C2


COUNT   lacl    ER_COUNT                ; Load counter in acc
        add     #1                      ; add 1 to acc
        sacl    ER_COUNT                ; store back in counter
        OUT     ER_COUNT, 0001h         ; write error counter to LED
        b       CONT

        .sect   "vectors"
        b       INIT
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                                ;
;       Serial Port Test for LC203 Board                         ;
;       File: sp.asm                                             ;
;                                                                ;
;       Description: This file is used to check the correct      ;
;       operation between the on-chip serial port and the two    ;
;       VBAPS. An Audio source is connected to the ANALOG IN ;
;       and the speakers should be connected to ANALOG OUT       ;
;       The serial port will take samples from the ANALOG IN     ;
;       and output the same sample to the ANALOG OUT.       ;
;       The speaker output should be the same as that applied to ;
;       the input.                                               ;
;       The multiplexing between the two VBAPS is achieved as    ;
;       follows:                                                 ;
;       INT2 provides a 16kHz interrupt which toggles            ;
;       the XF flag. The XF flag selects one of the VBAPS        ;
;       thereby giving an 8KhkHz sampling rate to each VBAP      ;
;       A file called MAKE.BAT will assemble, and link SP.ASM    ;
;       producing SP.OUT                                         ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

IMR      .set    4h                        ; Int Mask register
GREG     .set    5h                        ; Global Mask register
IFR      .set    6h                        ; Int Flag register
ICR     .set    0ffech                    ;ICR reg
SDTR    .set    0fff0h                    ; Synch Serial Port
                                           ; Transmit/Receive Reg
SSPCR   .set    0fff1h                    ; Synch Serial Port Control Reg
WSGR     .set    0fffch                    ; Wait State Generator reg

TEMP_VAL .usect   "temp",3                 ; Temporary value
RX_READY .set   TEMP_VAL+1                 ; Receive Flag
BUFFER   .set   TEMP_VAL+1

         .text
         NOP
INIT     ldp     #0
         splk    #2E00h,TEMP_VAL           ; ARP=1,OVM=1,INTM=1,DP=0
         lst     #0,TEMP_VAL               ; Init ST0
         splk    #21FCh,TEMP_VAL           ; ARB=1,CNF=0,SXM=0,XF=1,PM=0
         lst     #1,TEMP_VAL               ; Init ST1
         splk    #000Ah ,IMR               ; INT2/INT3 and Serial Port Receive
                                           ; Interrupt selected
        splk    #0001h,TEMP_VAL            ; INT2 selected
        out     TEMP_VAL,ICR               ; Init ICR reg
         splk    #0,GREG                   ; Disable Global Memory
         splk    #0E00h,TEMP_VAL           ; Init WSGR
         out     TEMP_VAL,WSGR             ; 7 WS I/0, 0 WS Data, 0 WS Prog
         splk    #000ah,TEMP_VAL           ; Loop Back Mode Selected
        out     TEMP_VAL,SSPCR             ; Init SSPCR
        splk    #003ah,TEMP_VAL            ; Boot up SSPCR
        out    TEMP_VAL,SSPCR              ; Init SSPCR
        lacl    #0
        sacl    RX_READY                   ; Init RX_READY flag
        setc    XF                         ; Selects the VBAP
        splk    #0000h,TEMP_VAL            ; TEMP_VAL = 0
        clrc    INTM                       ; Enable interrupts
```

```
START   b START

INT2    sst     #1,TEMP_VAL             ;Load ST1 in TEMP_VAL
        lacl    TEMP_VAL                ;Load ACC with ST1
        xor     #0010h                  ;Toggle XF bit from ST1
        sacl    TEMP_VAL
        lst     #1,TEMP_VAL             ;Restore ST1
        ;lacl   TEMP_VAL                ;Load int2 interrupt counter
        ;sub    #1                      ;Check if two int2 have occurred
        ;bcnd   OUTPUT,EQ
        ;lacl   TEMP_VAL
        ;add    #1                      ; add one to counter
        ;sacl   TEMP_VAL                ; store back
        ;clrc   INTM                    ;enable interrupts
        ;ret
OUTPUT  out     BUFFER,SDTR             ; Output old data value
        splk    #0000h,TEMP_VAL         ; zero the counter
        clrc    INTM                    ; Enable interrupts
        ret                             ; return

RECVINT in      BUFFER,SDTR             ; Input new data value
        clrc    INTM                    ; Enable interrupts
        ret                             ; return

        .sect   "vectors"
rs      b       INIT                    ; reset
int1    b       int1                    ; int1/hold
int2_3  b       INT2                    ; int2
tint    b       tint                    ; timer
rint    b       RECVINT                 ; Synch receive
xint    b       xint                    ; Synch transmit
txrxint b       txrxint                 ; Asynch Tran/Rec
        .space  2*16                    ; reserved interrupt for emulator
int8    b       int8
int9    b       int9
int10   b       int10
int11   b       int11
int12   b       int12
int13   b       int13
int14   b       int14
int15   b       int15
int16   b       int16
trap    b       trap
nmi     b       nmi                     ;  None maskable interrupt
        .space 2*16                     ;  reserved interrupt for emulator
int20   b       int20
int21   b       int21
int22   b       int22
int23   b       int23
int24   b       int24
int25   b       int25
int26   b       int26
int27   b       int27
int28   b       int28
int29   b       int29
int30   b       int30
int31   b       int31
```

```
/*****************************************************************/
/*    LC203 UART TEST                                            */
/*    File Names: uart.c,uart1.h,initp.asm                       */
/*                                                               */
/*    Description:  This program is used to check the correct    */
/*    operation of the on-chip UART. A Windows based        */
/*    application is needed to establish communication           */
/*    between the PC and the UART. The executable           */
/*    file UART.TRM is used in this case. From the               */
/*    communications menu choose, 9600 Baud Rate, 1 Stop bit,    */
/*    ,no parity check, 8 bit data, no flow control.             */
/*    Connect an RS-232 cable from the PC to the Board.          */
/*    Start the emulator with UART.OUT. Once UART.OUT is         */
/*    running on the DSP, click on the Windows application       */
/*    UART.TRM.                                                  */
/*    The following message should appear on the screen:         */
/*                                                               */
/*    *** LC203 Test Platform ***                                */
/*    Type a character                                           */
/*      The char should be displayed in the screen               */
/*    To stop the test press (s)                                 */
/*                                                               */
/*    A file called MAKE.BAT will compile, assemble, and link    */
/*    the necessary files creating the UART.OUT.                 */
/*****************************************************************/

#include "uart1.h"
void    main(void)
{
        #define max 100
        char C, COMMAND;
        char s[max];
        int i,n;
        int DONE = 0;
      void INIT(void);        /* Initialization routine                */

      INIT();
      sendinteger (0xa);      /* Set cursor begining of current line   */
      sendinteger (0xd);      /* Set cursor to next line               */
      sendinteger (0xa);      /* Set cursor begining of current line   */
      sendinteger (0xd);      /* Set cursor to next line               */
      sendstring(" *** LC203 Test Platform ***",27);
      sendinteger (0xa);      /* Set cursor begining of current line   */
      sendinteger (0xd);      /* Set cursor to next line               */
      sendinteger (0xa);      /* Set cursor begining of current line   */
      sendinteger (0xd);      /* Set cursor to next line               */
      sendstring(" Type a character ",18);
      sendinteger (0xa);      /* Set cursor begining of current line   */
      sendinteger (0xd);      /* Set cursor to next line               */
      sendstring(" The char should be display in the screen",41);
      sendinteger (0xa);      /* Set cursor begining of current line   */
      sendinteger (0xd);      /* Set cursor to next line               */
      sendstring(" To stop the test press (s) ",28);
      sendinteger (0xa);      /* Set cursor begining of current line   */
      sendinteger (0xd);      /* Set cursor to next line               */
      while (!DONE)
            {
            COMMAND = receivechar();
            switch (COMMAND)
                  {
                  case 's': case 'S':     DONE =1;
                                    break;
                  }
            sendchar (COMMAND);
            }
```

```
/*****************************************************************/
/*          Filename:  UART1.h                                   */
/*****************************************************************/


#include "c:\dsp\c2xx\lib\string.h"
#include "c:\dsp\c2xx\lib\ioports.h"
ioport int portffff4;
void ISTXREADY(void);
void ISRXREADY(void);
unsigned char reply;


void sendinteger (int i)
      {
      ISTXREADY();
      portffff4 = i;
      }

void sendchar ( char c)
      {
      ISTXREADY();
      portffff4 = c;
      }

void sendstring ( char s[],int length)
      {
      int c,i;
      i = 0;
      while ( --length >= 0)
            sendchar(s[i++]);
      }
char receivechar(void)
      {
      ISRXREADY();
      reply = portffff4;
      return (reply);
      }
```

```
*****************************************************************
*           UART TEST PROGRAM LC203 BOARD                       *
*           File Name intp.asm                                  *
*****************************************************************

IMR   .set  4h                      ; IMR reg
GREG  .set  5h                      ; GREG reg
IFR   .set  6h                      ; IFR reg
ADTR  .set  0fff4h                  ; ADTR reg
ASPCR .set  0fff5h                  ; ASPCR reg
IOSR  .set  0fff6h                  ; IOSR reg
BRD   .set  0fff7h                  ; BRD reg
WSGR  .set  0fffch                  ; WSGR reg
TEMP_VAL    .usect  "temp",2        ; Temp value
STAT1       .set  TEMP_VAL+1        ; ST1 storage

      .def  _INIT
      .def  _ISTXREADY
      .def  _ISRXREADY
      .ref  _c_int0

_INIT   ldp #0                      ;Set dp=0
      setc  INTM                    ;turn off interrupts
      clrc  CNF                     ;Map B0 in data space
      clrc  SXM                     ;No sign extension
      setc  OVM                     ;No overflow allowed
      SPM   0                       ;No product shift
      splk  #0000h,IMR              ;Turn off interrupts
      splk  #0ffffh,IFR             ;Clear all pending interrupt
      splk  #0,GREG                 ;Disable all global mem
      splk  #0e00h,TEMP_VAL         ;7 WS I/O, 0 WS DAT, 0 WS PRG
      out   TEMP_VAL,WSGR           ;Init WSGR
      splk  #0082h,TEMP_VAL         ;9600 Baud Rate
      out   TEMP_VAL,BRD            ;Init BDR
      splk  #0000h,TEMP_VAL         ;Reset TEMP_VAL
      out   TEMP_VAL,IOSR           ;Init IOSR
      splk  #0000h,TEMP_VAL         ;Reset UART
      out   TEMP_VAL,ASPCR          ;Init ASPCR
      splk  #2000h,TEMP_VAL         ;Boot the UART
      out   TEMP_VAL,ASPCR          ;UART is now turned on
      ret

_ISRXREADY  SST   #1,STAT1          ; Store ST1 in TEMP_VAL
READ_RX     IN    TEMP_VAL,IOSR     ; Read the IOSR register
            BIT   TEMP_VAL,0111b    ; Test DR bit from IOSR
            BCND  READ_RX,NTC       ; wait until TC=1
            LACL  STAT1             ; Load accumulator with ST1
            AND   #0f7ffh           ; Set TC=0
            SACL  STAT1             ; Store ST1 in TEMP_VAL
            LST   #1,STAT1          ; Restore ST1
            RET                     ; Return

_ISTXREADY  SST   #1,STAT1          ; Store ST1 in TEMP_VAL
READ_TX     IN    TEMP_VAL,IOSR     ; Read the IOSR register
            BIT   TEMP_VAL,0100b    ; Test THRE bit from IOSR
            BCND  READ_TX,NTC       ; wait until TC=1
```

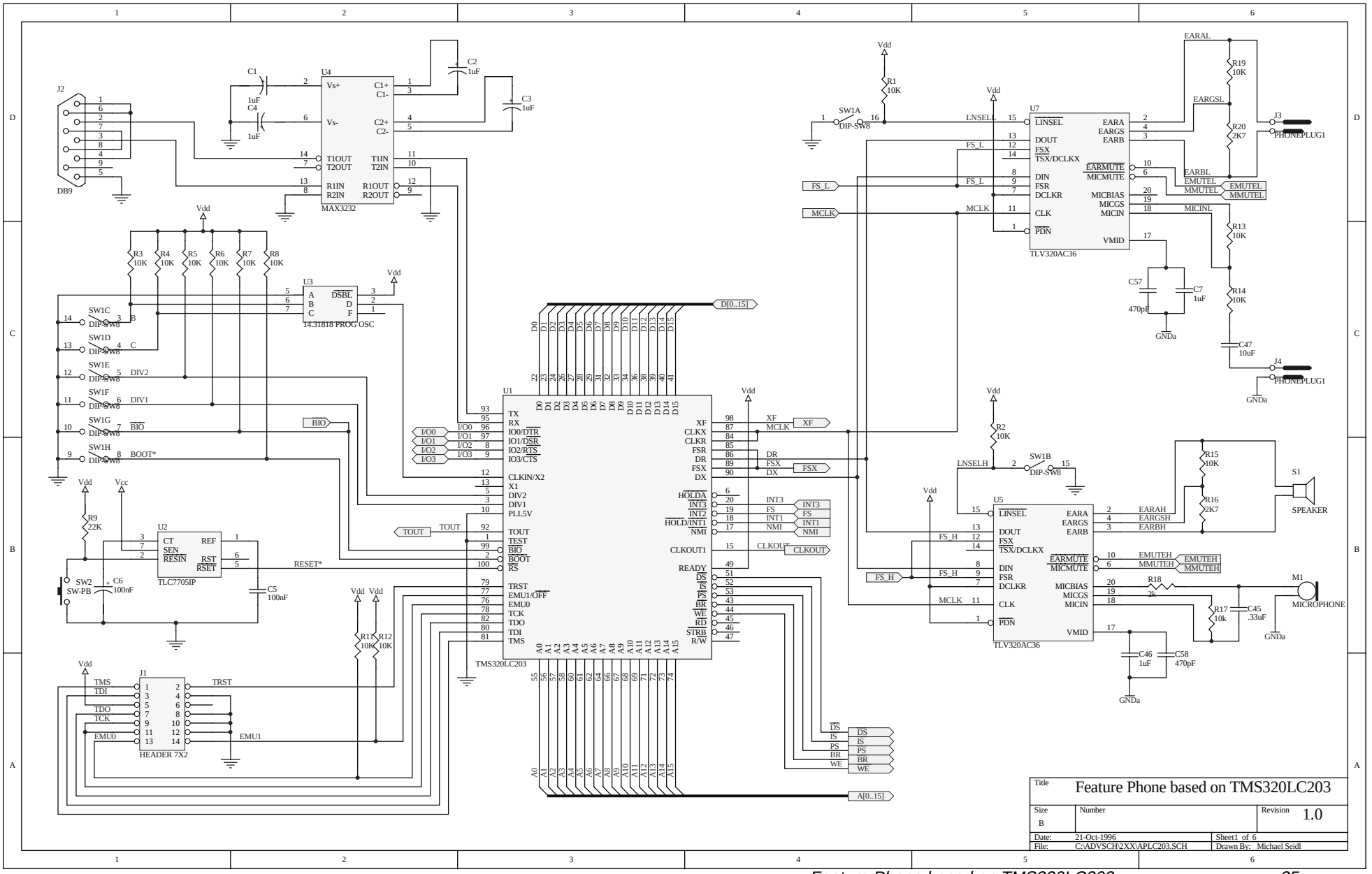
```
            LACL  STAT1       ; Load accumulator with ST1
            AND   #0f7ffh     ; Set TC=0
            SACL  STAT1       ; Store ST1 in TEMP_VAL
            LST   #1,STAT1    ; Restore ST1
            RET               ; Return


      .sect "vectors"
rs          b     _c_int0     ; reset
int1        b     int1        ; int1/hold
int2_3      b     int2_3      ; int2/int3
tint        b     tint        ; timer
rint        b     rint        ; Synch receive
xint        b     xint        ; Synch transmit
txrxint     b     txrxint     ; Asynch Tran/Rec
      .space      2*16        ; reserved interrupt for emulator
int8        b     int8
int9        b     int9
int10       b     int10
int11       b     int11
int12       b     int12
int13       b     int13
int14       b     int14
int15       b     int15
int16       b     int16
trap        b     trap
nmi         b     nmi         ;  None maskable interrupt
      .space 2*16               ;  reserved interrupt for emulator
int20       b     int20
int21       b     int21
int22       b     int22
int23       b     int23
int24       b     int24
int25       b     int25
int26       b     int26
int27       b     int27
int28       b     int28
int29       b     int29
int30       b     int30
int31       b     int31
```

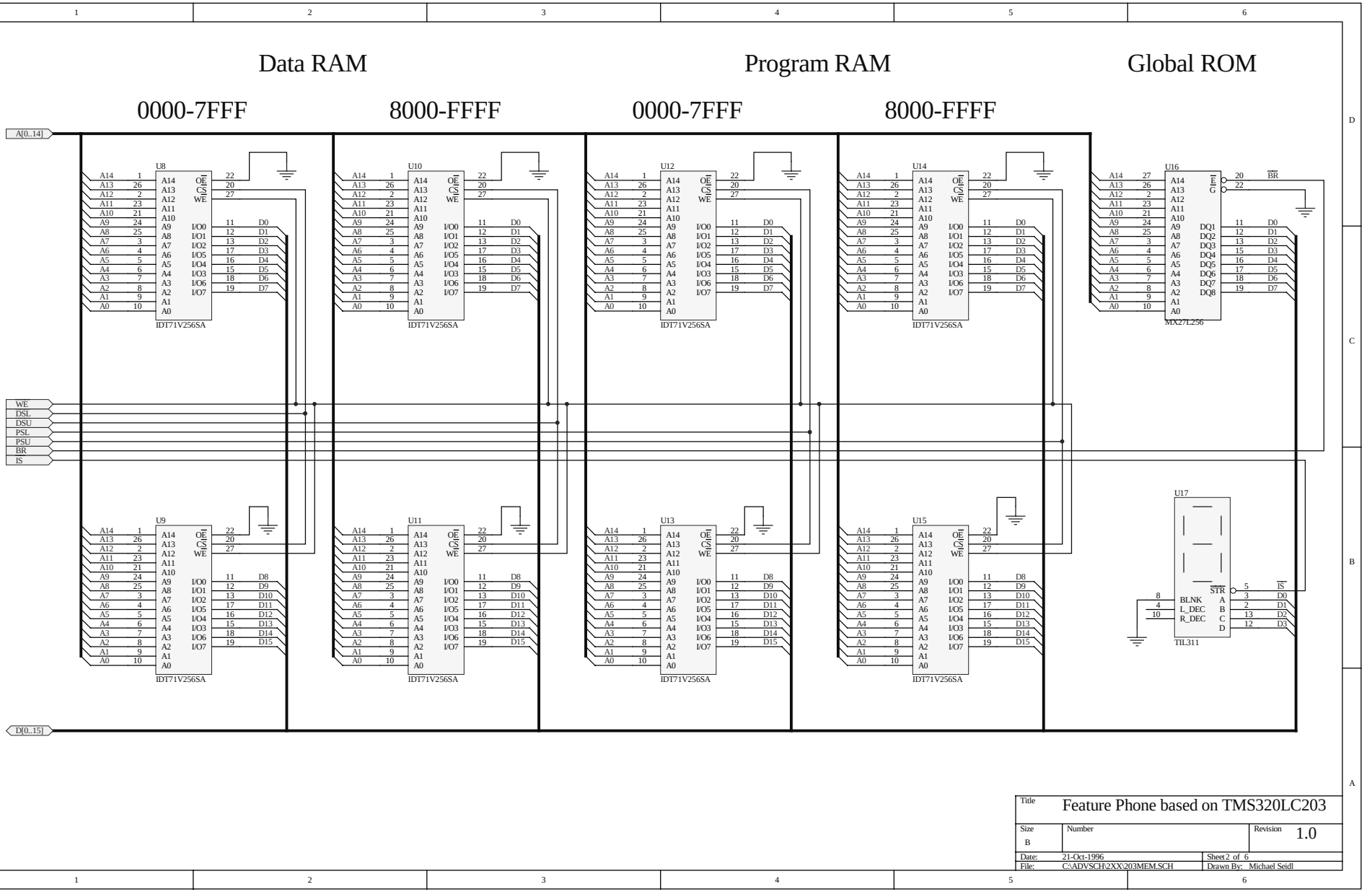
```
/*        TMS320LC203  UART LINK COMMAND FILE                          */
uart.obj
intp.obj
-m uart.map
-o uart.out
-stack 512
-cr
-i c:\dsp\c2xx\lib
-l c:\dsp\c2xx\lib\rts2xx.lib


MEMORY   {
  PAGE 0 :                                          /*  Program space */
         VECS      :  o = 0x0000  ,  l = 0x0040   /*  Vector area   */
         CODE      :  o = 0x0040  ,  l = 0x05C0   /*  1.4K CODE     */
         EXT_PRG   :  o = 0x0600  ,  l = 0xf7ff   /*  External Prog */
         BLK_B0    :  o = 0xFE00  ,  l = 0x0100   /*  BLK B0        */
         BLK_B0M   :  o = 0xFF00  ,  l = 0x0100   /*  BLK B0'       */

  PAGE 1 :                                          /*  Data space    */
         MEM_REG   :  o = 0x0000  ,  l = 0x0060   /*  Mem Map Reg   */
         BLK_B2    :  o = 0x0060  ,  l = 0x0020   /*  BLK B2        */
         BLK_B0    :  o = 0x0100  ,  l = 0x0100   /*  BLK B0        */
         BLK_B0M   :  o = 0x0200  ,  l = 0x0100   /*  BLK B0'       */
         BLK_B1    :  o = 0x0300  ,  l = 0x0100   /*  BLK B1        */
         BLK_B1M   :  o = 0x0400  ,  l = 0x0100   /*  BLK B1'       */
         EXT_DAT   :  o = 0x0800  ,  l = 0xf7ff   /*  Ext Data      */

               }
SECTIONS {
         vectors    :  {}   >   VECS    PAGE 0    /* int vectors     */
         .text      :  {}   >   CODE    PAGE 0    /* code            */
         .data      :  {}   >   BLK_B0  PAGE 0    /* Init data       */
         .bss       :  {}   >   BLK_B1  PAGE 1    /* Uninit data     */
         temp       :  {}   >   BLK_B2  PAGE 1    /* Temp Storage    */
       .cinit       :  {}   >   EXT_PRG PAGE 0
       .stack       :  {}   >   EXT_DAT PAGE 1
       .const       :  {}   >   EXT_DAT PAGE 1

         }
```

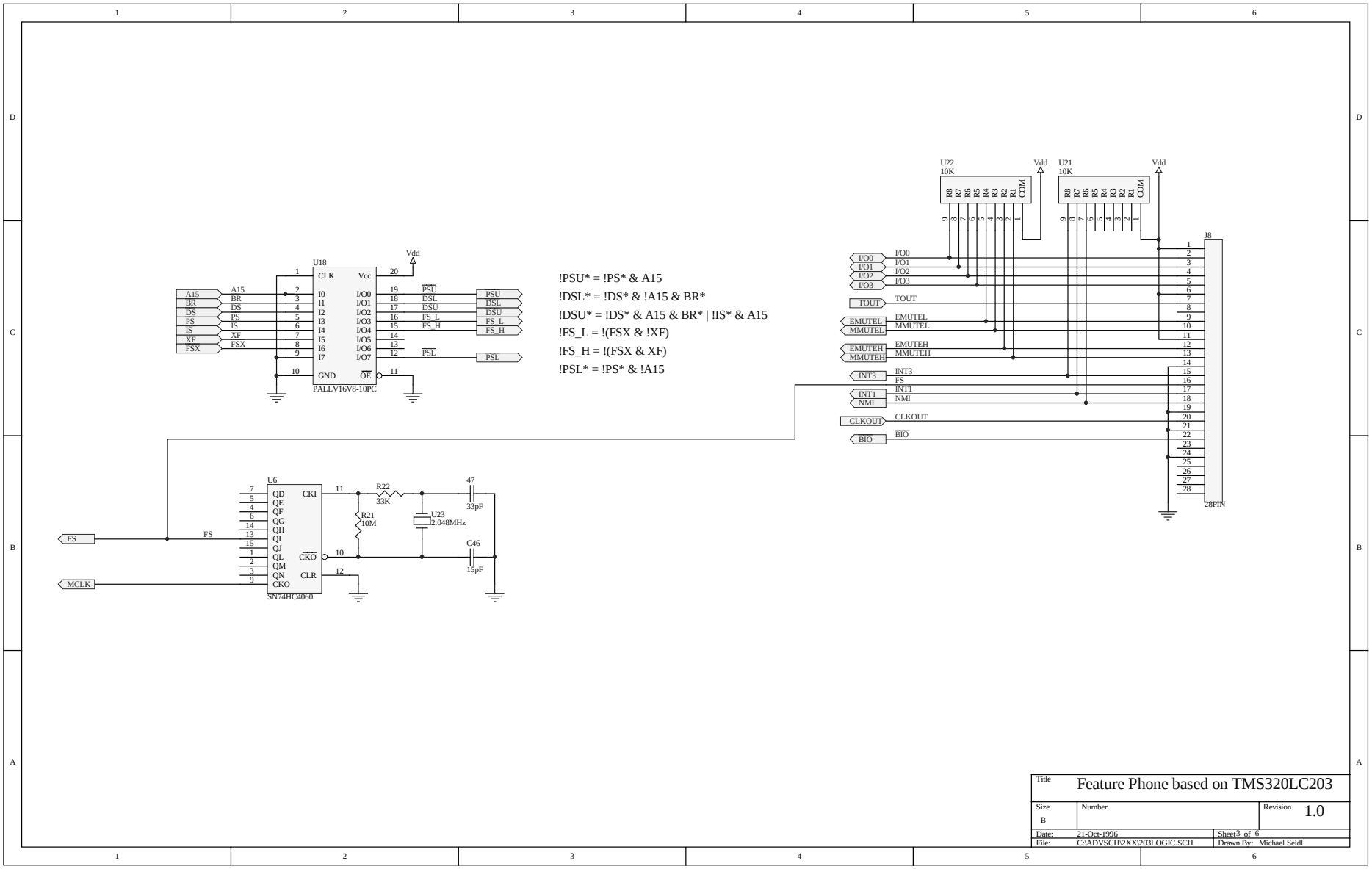
Appendix B. PAL Programming

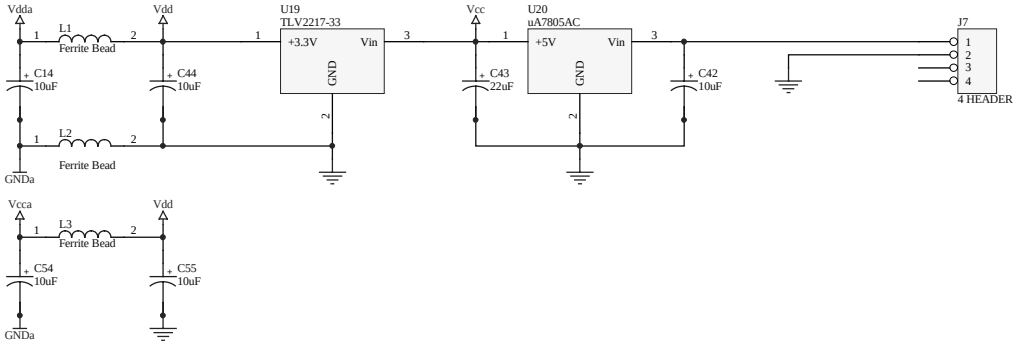
```
MODULE module_name
    gpt3 DEVICE 'p16v8as';

"INPUTS
    A15,BR,DS,PS                PIN 2,3,4,5;
    IS,XF,FSX                   PIN 6,7,8;



"OUTPUTS
    PSL,PSU,DSL,DSU,FSL,FSH  PIN 12,19,18,17,16,15;

    X,C,Z=.X.,.C.,.Z.;


EQUATIONS


!PSL    = !PS & !A15;
!PSU    = !PS & A15;

!DSL    = !DS & !A15 & BR;
!DSU    = !DS & A15 & BR # !IS & A15;

!FSL    = !(FSX & !XF);
!FSH    = !(FSX & XF);



TEST_VECTORS ([A15,BR,DS,PS,IS,XF,FSX] -> [PSL,PSU,DSL,DSU,FSL,FSH])

              [ 0, 0,  0, 0, 0, 0, 0]  -> [ 0 , 1 , 1 , 1 , 0 , 0 ];
              [ 1, 1,  0, 0, 1, 1, 1]  -> [ 1 , 0 , 1 , 0 , 0 , 1 ];
              [ 1, 1,  1, 1, 0, 0, 1]  -> [ 1,  1 , 1 , 0 , 1 , 0 ];
              [ 1, 0,  0, 1, 1, 1, 0]  -> [ 1,  1 , 1 , 1 , 0 , 0 ];


END module_name
```

Appendix C. Schematics

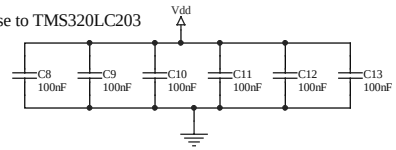

Feature Phone based on TMS320LC203
Size B
Number
Revision 1.0
Date: 21-Oct-1996
Sheet1 of 6
File: C:\ADVSCH\2XX\APLC203.SCH
Drawn By: Michael Seidl
U1
TMS320LC203
U2
TLC7705IP
U3
14.31818 PROG OSC
U4
MAX3232
U5
TLV320AC36
U7
TLV320AC36
J1
HEADER 7X2
J2
DB9
J3
PHONEPLUG1
J4
PHONEPLUG1
S1
SPEAKER
M1
MICROPHONE
SW1A
SW1B
SW1C
SW1D
SW1E
SW1F
SW1G
SW1H
SW2
SW-PB
Vdd
Vcc
GNDa
D[0..15]
A[0..15]
RESET*
BOOT*
BIO
TOUT
TRST
EMU1
EARAL
EARGSL
EARBL
MICINL
EARAH
EARGSH
EARBH
EMUTEL
MMUTEL
EMUTEH
MMUTEH


Data RAM
0000-7FFF
8000-FFFF
Program RAM
0000-7FFF
8000-FFFF
Global ROM
A[0..14]
WE
DSL
DSU
PSL
PSU
BR
IS
D[0..15]
U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
IDT71V256SA
MX27L256
TIL311
Title
Feature Phone based on TMS320LC203
Size
B
Number
Revision
1.0
Date:
21-Oct-1996
Sheet 2 of 6
File:
C:\ADVSCH\2XX\203MEM.SCH
Drawn By:
Michael Seidl

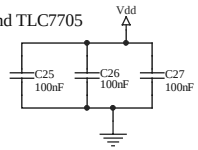

!PSU* = !PS* & A15
!DSL* = !DS* & !A15 & BR*
!DSU* = !DS* & A15 & BR* | !IS* & A15
!FS_L = !(FSX & !XF)
!FS_H = !(FSX & XF)
!PSL* = !PS* & !A15
U18
PALLV16V8-10PC
U6
SN74HC4060
U22
U21
10K
J8
28PIN
R22
33K
R21
10M
U23
12.048MHz
47
33pF
C46
15pF
Vdd
Title
Feature Phone based on TMS320LC203
Size
B
Number
Revision
1.0
Date:
21-Oct-1996
File:
C:\ADVSCH\2XX\203LOGIC.SCH
Sheet 3 of 6
Drawn By:
Michael Seidl



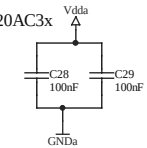
Close to TMS320LC203



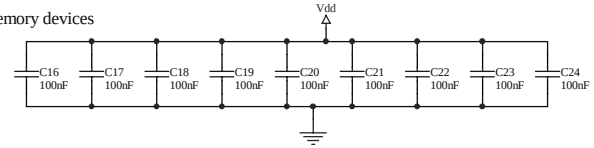
Close to Osc. and TLC7705



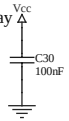
Close to TLV320AC3x



Close to memory devices

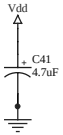


Close to LED display

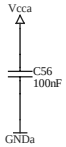


Close to PAL

Close to MAX3232



Close to TL061



Title	Feature Phone based on TMS320LC203		
Size: B	Number	Revision: 1.0	
Date: 21-Oct-1996		Sheet4 of 6	
File: C:\ADVSCH\2XX\203POWER.SCH		Drawn By: Michael Seidl	