;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

;

; Author : ADI - Apps

;

; Date : January 2001

;

; File : 824CTest.asm

;

; Description : QuickStart development kit basic test routine.

;

;\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

$MOD824 ; Use predefined Symbols

FLAG BIT 00h

FIRSTGO BIT 01H

LED EQU P3.4

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; DEFINE VARIABLES IN INTERNAL RAM

DSEG

ORG 0060h

COUNT: DS 1 ; define 1 byte for the count

DACOUT: DS 1 ; define 1 byte for the DAC o/p

DPH1: DS 1 ; \ define 2 bytes for the

DPL1: DS 1 ; / temporary storage of DPTR

DATAOUT: DS 1 ; the current data put into mem

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CSEG ; (beginning of code)

ORG 0000h

JMP MAIN

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ORG 0003h ; (INT0 ISR)

CLR FLAG ; clear flag

RETI

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

ORG 0060h ; (subroutines...)

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MAIN: ; (main program)

MOV SP,#80h

; CONFIGURE UART

MOV RCAP2H,#0FFh ; config UART for 9830baud

MOV RCAP2L,#-5 ; (close enough to 9600baud)

MOV TH2,#0FFh

MOV TL2,#-5

MOV SCON,#52h

MOV T2CON,#34h

; CONFIGURE INTERRUPT 0...

SETB IT0 ; INT0 edge triggered

SETB EX0 ; enable INT0 interrupt

; ENABLE INTERRUPTS & ENTER MAIN LOOP...

SETB EA ; enable inturrupts

SETB FLAG

MOV A, #01h

SETB TI ; set TI flag to indicate that the

; buffer is ready to transmit data.

MOV DPTR, #GOMENU ; send a press INT0 message

CALL SENDSTRING

SETB FIRSTGO

BLINK: CPL LED ; blink LED until INT0 button pressed

CALL DELAY

JB FLAG, BLINK

SETB LED ; turn on LED

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

MAINMENU:

JB FIRSTGO, SKIP1

MOV DPTR, #RETMENU ; send a press INT0 message

CALL SENDSTRING

SETB FLAG

JB FLAG, $ ; wait for INT0

JMP SKIP2

SKIP1:

CLR FIRSTGO

MOV DPTR,#TITLE ; display title

CALL SENDSTRING

MOV DPTR,#LINKS1 ; display link options

CALL SENDSTRING

MOV DPTR,#LINKS2 ; display link options

CALL SENDSTRING

SKIP2:

MOV DPTR,#MENU ; display menu

CALL SENDSTRING

JNB RI, $ ; wait here for a key press

MOV A, SBUF ; read input

CLR RI ; clear RI flag for the next input

CJNE A, #'1', $+6 ; if a 1 (31H) is pressed jmp TEST1

JMP TEST1

CJNE A, #'2', $+6 ; if a 2 (32H) is pressed jmp TEST2

JMP TEST2

CJNE A, #'3', $+6 ; if a 3 (33H) is pressed jmp TEST3

JMP TEST3

CJNE A, #'4', $+6 ; if a 4 (34H) is pressed jmp TEST4

JMP TEST4

CJNE A, #'5', $+6 ; if a 5 (35H) is pressed jmp TEST5

JMP TEST5

JMP MAINMENU

;====================================================================

; TEST 1: ADC AND DAC TEST

;====================================================================

TEST1:

MOV DPTR,#PRIMADC ; display ADC title

CALL SENDSTRING

MOV DACCON, #17h ; configure DAC for

; 0-Vdd (5V), 12 bits,

MOV ADC0CON, #47h ; configure primary ADC for

; external ref 2.5V, Ain1-Ain2,

; bipolar, +-2.56V

SETB FLAG

JB FLAG, $ ; wait for INT0

MOV DACOUT, #0Ch ; start of prim ADC conv

MOV COUNT, #05h ; # of conv

CALL TESTPRIMADC

;aux channel Ain3

MOV DPTR,#AUX3ADC

CALL SENDSTRING

MOV ADC1CON,#48h ; AIN3, unipolar, ext ref

MOV DACCON,#13h ; turn DAC on, 0-2.5V

SETB FLAG

JB FLAG, $ ; wait for INT0

MOV DACOUT, #05H

CALL TESTAUXCHAN

;aux channel Ain5

MOV DPTR,#AUX5ADC

CALL SENDSTRING

MOV ADC1CON,#78h ; AIN5, unipolar, ext ref

MOV DACCON,#13h ; turn DAC on, 0-2.5V

SETB FLAG

JB FLAG, $ ; wait for INT0

MOV DACOUT, #05H

CALL TESTAUXCHAN

; END OF TEST 1

MOV ADCMODE,#00h ; turn ADCs off

MOV DACCON, #00h ; turn offthe DAC

JMP MAINMENU ; return

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TESTPRIMADC:

; output voltage on DAC

MOV DACH,DACOUT ; change hi-byte only

MOV DACL,#00h

MOV R0,#10h ; allow for settling

DJNZ R0,$

; do a single conversion on the DAC voltage. Ain2 biased to

; 2.5V. Hence the ADC conv voltage is the DAC voltage - 2.5V

MOV ADCMODE,#22h ; single conv on prim chan

JNB RDY0,$ ; wait for ADC result

CLR RDY0

; display the DAC output on the screen

MOV DPTR,#DACMSG ; "DAC="

CALL SENDSTRING

MOV R6, #00h

MOV R7, DACOUT ; display DAC value

CALL SEND12BITS

; display the ADC conv result on the screen

MOV DPTR,#ADC0MSG ; " -> ADC0 = "

CALL SENDSTRING

MOV A, ADC0H ; send the 24 bit result upo the UART

CALL SENDVAL

MOV A, ADC0M

CALL SENDVAL

MOV A, ADC0L

CALL SENDVAL

CALL CHECKADC0 ; see if results are "OK" or "ERROR!"

DEC DACOUT

DJNZ COUNT, TESTPRIMADC

RET

TESTAUXCHAN:

; output voltage on DAC

MOV DACH,DACOUT ; change hi-byte only

MOV DACL,#00h ; load DAC

MOV R0,#10h ; allow for settling

DJNZ R0, $

; do a single conversion on the DAC voltage. The DAC voltage

; appears between Ain3 and ground. Therefore the aux conv

; should equal the DAC o/p

MOV ADCMODE,#12h ; single conv on aux chan

JNB RDY1,$ ; wait for ADC result

CLR RDY1

; display the DAC output on the screen

MOV DPTR,#DACMSG ; "DAC="

CALL SENDSTRING

MOV R6, #00h

MOV R7, DACOUT ; display DAC value

CALL SEND12BITS

; display the ADC conv result on the screen

MOV DPTR,#ADC1MSG ; "ADC1="

CALL SENDSTRING

MOV A, ADC1H ; display ADC results

CALL SENDVAL

MOV A, ADC1L

CALL SENDVAL

CALL CHECKADC1 ; "ok" or "ERROR!"

DJNZ DACOUT, TESTAUXCHAN

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CHECKADC0:

MOV A, DACOUT

SWAP A ; Swap Daccout from 0Ch to C0h etc

SUBB A, #08h

MOV R2, A ; R2 holds B8h (for DACH=0C)

MOV A, ADC0H ; A holds say C1h

SUBB A, R2

CJNE A, #10h, $+3 ; C is set if A<10H i.e. OK

JC $+6

CALL PRINTERROR

RET

CALL PRINTOK

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

CHECKADC1:

; check if the ADC result is within 8LSBs of the DAC op

MOV A, DACOUT

SWAP A

SUBB A, #08h

MOV R2, A

MOV A, ADC1H

SUBB A, R2

CJNE A, #10h, $+3 ; C is set if A<10H i.e. OK

JC $+6

CALL PRINTERROR

RET

CALL PRINTOK

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

;====================================================================

; TEST 2: EXTERNAL DATA MEMORY TEST

;====================================================================

TEST2:

; save particular data into internal RAM location 50h->59h

EXTDATA:

MOV R0, #50h

MOV @R0, #0FFh ; store FFh into 50H

INC R0

MOV @R0, #00h ; store 00h into 51h

INC R0

MOV @R0, #0AAh ; store AAh into 52h

INC R0

MOV @R0, #55h ; store 55h into 53h

INC R0

MOV @R0, #0CCh ; store CCh into 54h

INC R0

MOV @R0, #33h ; store 33h into 55h

INC R0

MOV @R0, #3Fh ; store 3Fh into 56h

INC R0

MOV @R0, #66h ; store 66h into 57h

MOV DPTR,#MEMTITLE ; pick a memory mode

CALL SENDSTRING

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LK9A:

MOV DPTR,#LINK9A ; "LK9 to position A"

CALL SENDSTRING

SETB FLAG

JB FLAG, $ ; wait for INT0

MOV DPP,#00h

MOV R0, #50h

MOV DPTR, #05FFH

; save data in RAM into the following locations

XRAMIN1:

MOV DPP,#00h

MOV A, @R0

MOV DATAOUT, @R0

MOVX @DPTR, A

MOV DPH1, DPH ; DPH1 and DPL1 hold the present

MOV DPL1, DPL ; address in XRAM

MOV DPTR, #ADDRMSG

CALL SENDSTRING

MOV A, DPP

CALL SENDVAL

MOV A, DPH1

CALL SENDVAL

MOV A, DPL1

CALL SENDVAL

MOV DPTR, #WRITTENMSG

CALL SENDSTRING

MOV A, DATAOUT

CALL SENDVAL

MOV DPTR, #READMSG

CALL SENDSTRING

MOV DPP, #55H ; change the page number. DPP does

MOV DPH, DPH1 ; not hold any address lines in this mode

MOV DPL, DPL1

MOVX A, @DPTR

CALL SENDVAL

CJNE A, DATAOUT, ERROR2A

CALL PRINTOK

JMP $+6

ERROR2A:

CALL PRINTERROR

MOV DPH, DPH1

MOV DPL, DPL1

INC R0

MOV A, DPH

ADD A, #20h

MOV DPH, A

CJNE R0, #58H, XRAMIN1

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

LK9B: MOV DPTR,#LINK9B ; "LK9 to position B"

CALL SENDSTRING

SETB FLAG

JB FLAG, $ ; wait for INT0

MOV DPP,#05h

MOV DPTR, #00FFH

MOV R0, #57h ; R0 starts at 57h

; save data in RAM into the following locations

XRAMIN2:

MOV A, @R0

MOV DATAOUT, @R0

MOVX @DPTR, A

MOV DPH1, DPH ; DPH1 --> DPL1 hold the present

MOV DPL1, DPL ; address in XRAM

MOV DPTR, #ADDRMSG

CALL SENDSTRING

MOV A, DPP

CALL SENDVAL

MOV A, DPH1

CALL SENDVAL

MOV A, DPL1

CALL SENDVAL

MOV DPTR, #WRITTENMSG

CALL SENDSTRING

MOV A, DATAOUT

CALL SENDVAL

MOV DPTR, #READMSG

CALL SENDSTRING

MOV DPH, #55h ; corrupt DPH as DPP contains the

MOV DPL, DPL1 ; high address lines in this case

MOVX A, @DPTR

CALL SENDVAL

CJNE A, DATAOUT, ERROR2B

CALL PRINTOK

JMP $+6

ERROR2B:

CALL PRINTERROR

MOV DPH, DPH1

MOV DPL, DPL1

DEC R0

MOV A, DPP

ADD A, #20h

MOV DPP, A

CJNE R0, #4FH, XRAMIN2

JMP MAINMENU

;====================================================================

; TEST 3: SPI Interface TEST

;====================================================================

TEST3:

MOV DPTR,#SPITITLE ; display SPI title

CALL SENDSTRING

SETB FLAG

JB FLAG, $ ; wait for INT0

MOV SPICON,#33h ; configure SPI port Master Mode

; slowest bit rate

; Test SCLOCK

MOV SPIDAT, #00h ; transmit 00h

WAITFORLOW:

JB P3.3, SCLOCKHIGH; wait for SCLOCK to go low

WAITFORHIGH:

JNB P3.3, SCLOCKLOW ; wait for SCLOCK to go high

JNB ISPI, $

CLR ISPI

JMP TESTSDATA

SCLOCKHIGH:

JB ISPI, SPIERROR

JMP WAITFORLOW

SCLOCKLOW:

JB ISPI, SPIERROR

JMP WAITFORHIGH

; Test SDATA

TESTSDATA:

MOV DPTR, #SDATAMSG

CALL SENDSTRING

SETB FLAG

JB FLAG, $ ; wait for INT0

; ; reinitialise the SPIPORT

; MOV SPICON, #0

; MOV SPICON,#33h ; configure SPI port Master Mode

; slowest bit rate

MOV SPIDAT, #0Fh ; transmit 0Fh

WAITFORLOW1:

JB P3.3, SDATAHIGH; wait for SCLOCK to go low

WAITFORHIGH1:

JNB P3.3, SDATALOW ; wait for SCLOCK to go high

JNB ISPI, $

CLR ISPI

JMP TESTSS

SDATAHIGH:

JB ISPI, SPIERROR

JMP WAITFORLOW1

SDATALOW:

JB ISPI, SPIERROR

JMP WAITFORHIGH1

TESTSS:

; connect p3.3 to SS

MOV DPTR, #SSMSG

CALL SENDSTRING

SETB FLAG

JB FLAG, $ ; wait for INT0

MOV P3, #0FFh ; set all P3 as inputs

SETB P3.5

JNB P3.3, SPIERROR

CLR P3.5

JB P3.3, SPIERROR

SPIOK:

MOV DPTR, #SPIMSG

CALL SENDSTRING

CALL PRINTOK

JMP MAINMENU

SPIERROR:

MOV DPTR, #SPIMSG

CALL SENDSTRING

CALL PRINTERROR

JMP MAINMENU

;====================================================================

; TEST 4: RTD Demo Circuit TEST

;====================================================================

TEST4:

MOV DPTR,#RTDTITLE ; display RTD title

CALL SENDSTRING

MOV DPTR,#RTDLINKS ; display RTD links

CALL SENDSTRING

; Configure Current source

MOV ICON, #01h ; put 200uA out on pin 3

SETB FLAG

JB FLAG, $ ; wait for INT0

; Configure ADC

MOV ADC0CON, #4AH ; use external ref (R11)

; unpolar mode

; fixed 0->80mV range

MOV ADCMODE, #22H ; initiate a single prim chan conv

JNB RDY0,$ ; Wait for conversion results

MOV DPTR, #RTDSENSOR ; Send temp up UART

CALL SENDSTRING

MOV A, ADC0H

CALL SENDVAL

MOV A, ADC0M

CALL SENDVAL

; a value in AD0H of between 88h and A7h represents temps

; between -11degC and 47degC. Only these temps should

; pass the test.

MOV A, ADC0H

CLR CY

SUBB A, #08h ; A7->9Fh, 88->80h

ANL A, #0F0h ; correct data is now either 80h or 90h

CJNE A, #90h, CHECK80 ; if fails 90 check A0

CALL PRINTOK

JMP MAINMENU

CHECK80:

CJNE A, #80h, ERROR4 ; fails 80 and 90 => fail

CALL PRINTOK

JMP MAINMENU

ERROR4: CALL PRINTERROR

JMP MAINMENU

;====================================================================

; TEST 5: Automatic Checks

;====================================================================

TEST5:

MOV DPTR, #AUTO

CALL SENDSTRING

;test port 1 for shorts

MOV DPTR, #PORT1

CALL SENDSTRING

SETB P1.1

CLR P1.0

MOV A, P1

ANL A, #03h ; zero acc.2 -> acc.7

CJNE A, #02h, $+8 ; p1.1=1, p1.0=0 =>OK

CALL PRINTOK

AJMP $+5

CALL PRINTERROR

;test port 3 for shorts

MOV DPTR, #PORT3

CALL SENDSTRING

SETB P3.2 ; do not alter TXD and RXD lines

CLR P3.3

SETB P3.4

CLR P3.5

SETB P3.6

CLR P3.7

MOV A, P3

ANL A, #0FCh ; zero acc.0 and acc.1

CJNE A, #54h, $+8

CALL PRINTOK

AJMP $+5

CALL PRINTERROR

;use TIC to test if crystal is oscillating

MOV DPTR, #CRYSTAL

CALL SENDSTRING

MOV INTVAL, #0BH ; SET INTVAL FOR 12/128TH =93ms

MOV TIMECON, #03H ; MEASURE IN 1/128THS SECS

MOV A, #01H

CALL DELAY ;SOFTWARE 100ms delay => TII set

MOV A, TIMECON

JB ACC.2, $+8

CALL PRINTERROR

AJMP $+5

CALL PRINTOK

MOV TIMECON, #00h ; this clears TII and disables TIC

;any other automatic tests go here

MOV DPTR, #REMOVELINKS1

CALL SENDSTRING

MOV DPTR, #REMOVELINKS2

CALL SENDSTRING

JMP MAINMENU

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

TITLE: DB 10,10,10,13,'\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_',10,13

DB ' Analog Devices MicroConverter ADuC824',10,13

DB ' Evaluation Board Test Routine',0

Menu:

DB 10,10,13,'Main Menu:', 10, 13

DB ' 1 - Test ADCs and DAC',10,13

DB ' 2 - Test External Data Memory' ,10,13

DB ' 3 - Test SPI Interface', 10,13

DB ' 4 - Test RTD Demo Circuit',10,13

DB ' 5 - Automatic Checks',10,13

DB 'Select: ',0

LINKS1: DB 10,10,13,'Ensure that ONLY the following links are inserted'

DB 10,13,' - Slide LK1 to ON Posn'

DB 10,13,' - Slide LK5 to ON Posn'

DB 10,13,' - Slide LK6 to Posn A'

DB 10,13,' - Slide LK7 to Posn A',0

LINKS2: DB 10,13,' - Slide LK8 to Posn A'

DB 10,13,' - Slide LK9 to Posn A'

DB 10,13,' - Slide LK10 to Posn A'

DB 10,13,' - Slide LK11 to ON Posn'

DB 10,13,' - Slide LK12 to ON Posn',0

REMOVELINKS1:

DB 10,10,13,'If finished INSERT only the following Links'

DB 10,13,' - Slide LK1 to ON Posn'

DB 10,13,' - Slide LK3 to ON Posn'

DB 10,13,' - Slide LK5 to ON Posn'

DB 10,13,' - Slide LK6 to ON Posn',0

REMOVELINKS2:

DB 10,13,' - Slide LK7 to Posn A'

DB 10,13,' - Slide LK8 to Posn A'

DB 10,13,' - Slide LK9 to Posn A'

DB 10,13,'All other links should be in the OFF posn',10,13,0

PRIMADC: DB 10,10,13,'TEST 1: Testing ADCs and DAC in a Feedback Configuration'

DB 10,13,'-------------------------------------------------------',10,13

DB 'Connect the DAC (J2-13) to the Primary ADC (Ain1 J2-9).',10,13

DB 'Press the INT0 button when ready',10,13,0

AUX3ADC: DB 10,10,13,'Connect the DAC (J2-13) to the Auxilary ADC (Ain3 J2-11)',10,13

DB 'Press the INT0 button when ready',10,13,0

AUX5ADC: DB 10,10,13,'Connect the DAC (J2-13) to the Auxilary ADC (Ain5 J2-6)',10,13

DB 'Press the INT0 button when ready',10,13,0

MEMTITLE: DB 10,10,13,' TEST 2: Testing the External Data Memory'

DB 10,13,'------------------------------------------------------',10,13,0

LINK9A: DB 10,13,'Slide LK9 to position A. Press INT0 when ready.',10,13,0

LINK9B: DB 10,13,'Slide LK9 to position B. Press INT0 when ready.',10,13,0

SPITITLE: DB 10,10,13,' TEST 3: Testing the SPI/I2C Interface'

DB 10,13,'------------------------------------------------------',10,13

DB 10,13,'Connect SCLOCK (J7-1) to P3.3 (J3-4).'

DB 10,13,'Press the INT0 button when ready',10,13,0

SDATAMSG: DB 10,13,'Connect SDATA (J7-3) to P3.3 (J3-4)',10,13,0

SSMSG: DB 10,13,'Connect SS (J7-7) to P3.3 (J3-4)',10,13,0

SPIMSG: DB 10,13,'Testing SPI/I2C Interface...',0

RTDTITLE: DB 10,10,13,' TEST4: Testing the RTD Demo Circuit'

DB 10,13,'------------------------------------------------------',0

RTDLINKS:

DB 10,13,'Change the following Links:'

DB 10,13,' - Move LK7 into position B'

DB 10,13,' - Move LK8 into position B'

DB 10,13,' - REMOVE LK5'

DB 10,13,'NOTE: Remove any connector used in test 1'

DB 10,10,13,'Press INT0 when ready.',10,13,0

AUTO: DB 10,10,13,' TEST5: Automatic Checks'

DB 10,13,'------------------------------------------------------',0

PORT3: DB 10,13,'Testing Port 3 for shorts ', 0

PORT1: DB 10,13,'Testing Port1.0 and P1.1 for shorts ', 0

CRYSTAL: DB 10,13,'Testing crystal ciruit ',0

GOMENU: DB 10,10,10,13,'Press INT0 to go to the Main Menu',0

RETMENU: DB 10,13,'Press INT0 to return to the Main Menu',0

DACMSG: DB ' DAC=',0

ADC0MSG: DB ' -> Primary ADC=',0

ADC1MSG: DB ' -> Aux ADC=',0

RTDSENSOR: DB ' Temperature Conversion = ',0

OKMSG: DB ' OK',10,13,0

ERRORMSG: DB ' ERROR!!',10,13,0

ADDRMSG: DB 13,' ADDR=',0

WRITTENMSG: DB ': WRITTEN=',0

READMSG: DB ', READ=',0

;====================================================================

; FUNCTIONS

;====================================================================

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; PRINT OK AND ERROR MESSAGES

PRINTOK:

MOV DPTR,#OKMSG

CALL SENDSTRING

RET

PRINTERROR:

MOV DPTR,#ERRORMSG

CALL SENDSTRING

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; DELAY

DELAY: ; Delays by 100ms \* A

; 100mSec based on 1.573MHZ Core Clock

MOV R2,A ; Acc holds delay variable

DLY0: MOV R3,#50 ; Set up delay loop0

DLY1: MOV R4,#131 ; Set up delay loop1

DJNZ R4, $ ; Dec R4 & Jump here until R4 is 0

; wait here for 131\*15.3us=2ms

DJNZ R3, DLY1 ; Dec R3 & Jump DLY1 until R3 is 0

; Wait for 50\*2ms

DJNZ R2,DLY0 ; Dec R2 & Jump DLY0 until R2 is 0

; wait for ACC\*100ms

RET ; Return from subroutine

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SENDSTRING: ; sends ASCII string to UART starting at location DPTR

; and ending with a null (0) value

PUSH ACC

PUSH B

CLR A

MOV B,A

SENDNEXT: MOV A,B

INC B

MOVC A,@A+DPTR

JZ SENDDONE

JNB TI,$ ; wait til present char gone

CLR TI ; must clear TI

MOV SBUF,A ; transmit byte

JMP SENDNEXT

SENDDONE: POP B

POP ACC

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SENDCHAR: ; sends ASCII value contained in A to UART

JNB TI,$ ; wait til present char gone

CLR TI ; must clear TI

MOV SBUF,A

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

HEX2ASCII: ; converts A into the hex character representing the

; previous value of A's least significant nibble

ANL A,#00Fh

CJNE A,#00Ah,$+3

JC CONV

ADD A,#007h

CONV: ADD A,#'0'

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

SENDVAL: ; uses the above two subroutines to....

; 1) convert the hex value of A into two ASCII chars

; 2) spew converted chars out the UART

PUSH ACC

SWAP A

ANL A,#00Fh

CALL HEX2ASCII

CALL SENDCHAR

POP ACC

PUSH ACC

ANL A,#00Fh

CALL HEX2ASCII

CALL SENDCHAR

POP ACC

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; shifts the 12MSBs of R7,R6 to the 12LSBs of R7,R6.

; A zero is read into the 4 MSBs of R7

SHIFT:

PUSH ACC

MOV A,R6

ANL A,#0F0h

SWAP A

MOV R6,A

MOV A,R7

SWAP A

ANL A,#0F0h

ORL A,R6

MOV R6,A

MOV A,R7

SWAP A

ANL A,#0Fh

MOV R7,A

POP ACC

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

; sends the 12 LSBs of R7,R6 up the UART in ASCII form

SEND12BITS:

PUSH ACC

MOV A,R7

ANL A,#0Fh

CALL HEX2ASCII

CALL SENDCHAR

MOV A,R6

SWAP A

ANL A,#0Fh

CALL HEX2ASCII

CALL SENDCHAR

MOV A,R6

ANL A,#0Fh

CALL HEX2ASCII

CALL SENDCHAR

POP ACC

RET

;\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

END