KL05Z UART0 Polled Serial I/O

The Rx (receive) and Tx (transmit) pins of UART0 (UART zero) on the KL05 can be connected for a serial port connection through the OpenSDA mini-B USB connector (J2) on the KL05Z Freedom board to use with a USB serial port driver. The base address of UART0 in the KL05 memory map is 0x4006A000. Its registers appear to the Cortex-M0+ CPU as being at the following addresses.

Register Symbol	Register Name	Memory Map Address
UART0_BDH	Baud Rate High	0x4006A000
UART0_BDL	Baud Rate Low	0x4006A001
UART0_C1	Control Register 1	0x4006A002
UART0_C2	Control Register 2	0x4006A003
UART0_S1	Status Register 1	0x4006A004
UART0_S2	Status Register 2	0x4006A005
UART0_C3	Control Register 3	0x4006A006
UART0_D	Receive Data Register (Read)/ Transmit Data Register (Write)	0x4006A007
UART0_MA1	Match Address Register 1	0x4006A008
UART0_MA2	Match Address Register 2	0x4006A009
UART0_C4	Control Register 4	0x4006A00A
UART0_C5	Control Register 5	0x4006A00B

Note that both the transmit data register (TDR, write only) and receive data register (RDR, read only), map to the same UART0_D address. The MKL05Z4.s include file provided for this course has EQUates for the following UART0 register values.

Quantity	EQUate Name	Example EQUate
UART0 base address	UART0_BASE	UART0_BASE
Register address	Register symbol	UART0_S1
Register address offset from UART0 base address	Register symbol with _OFFSET suffix	UART0_S1_OFFSET
Register field bit mask	Register symbol with _field name and _MASK suffix	UART0_S1_TDRE_MASK
Register field bit offset from lsb	Register symbol with _field name and _SHIFT suffix	UART0_S1_TDRE_SHIFT

Using UART0 in a program for polled serial I/O involves several initialization steps. KL05 system configuration options related to UART0 affect various UART0 initialization settings, so the system configuration options must be set first. Next the UART0 can be initialized, which consists of disabling UART0, setting the serial communication speed,

configuring the serial communication protocol, and then enabling the UART0 transmitter and receiver. This initialization is required only once in the program—typically near its start with other initialization code. Afterward, UART0 may be used for sending and receiving as many characters as desired.

Hardware configuration

In lab, UART0 is used to send and receive characters to and from a terminal using a character format of eight data bits, no parity, and one stop bit (8N1) at 9600 baud. The OpenSDA application on the KL05Z provides a USB virtual seral port through the KL05Z OpenSDA mini-B USB connector (J2). On the KL05, UART0_RX and UART0_TX connect to the virtual serial port through port B pins 2 and 1 (respectively). No additional external wiring is required to use UART0 for this configuration.

Program initialization

The following initializations are required for sending and receiving characters via the USB virtual serial port using an 8N1 format at 9600 baud.

- UART0 module clock source selected as MCG FLL clock (MCGFLLCLK) in SIM_SOPT2: UART0SRC = 01
- UART0 receive and transmit through UART0_RX and UART0_TX pins with open drain disabled in SIM_SOPT5:
  UART0ODE = 0; UART0RXSRC = 0; UART0TXSRC = 0
- UART0 module clock enabled in SIM_SCGC4: UART0 = 1
- Port B module clock enabled in SIM_SCGC5: PORTB = 1
- UART0_TX connected to port B pin 1 in PORTB_PCR1: MUX = 010_2
- UART0_RX connected to port B pin 2 in PORTB_PCR2: MUX = 010_2
- UART0 one stop bit using polling at 9600 baud selected in UART0_BDH:
  LBKDIE = 0; RXEDGIE = 0; SBNS = 0; SBR[12:8] = 00001_2
- UART0 at 9600 baud selected in UART0_BDL: SBR[7:0] = 0x38
- UART0 normal operation with eight data bits with no parity selected in UART0_C1:
  LOOPS = 0; DOZEEN = 0; M = 0; WAKE = 0; ILT = 0; PE = 0
- UART0 receiver and transmitter enabled for polling in UART0_C2:
  TIE = 0; TCIE = 0; RIE = 0; ILIE = 0; TE = 1 RE = 1; RWU = 0; SBK = 0
- UART0 polling with normal transmitter operation in UART0_C3:
  TXDIR = 0; TXINV = 0; ORIE = 0; NEIE = 0; FEIE = 0 PEIE = 0
- UART0 eight data bits with normal receiver operation using default sampling rate in UART0_C4:
  MAEN1 = 0; MAEN2 = 0; M10 = 0; OSR[4:0] = 01111_2
- UART0 polling with normal receiver operation in UART0_C5:
  TDMAE = 0; RDMAE = 0; BOTHEDGE = 0; RESYNCDIS = 0
- UART0 normal/default operation in UART0_S5:
  MSBF = 0; RXINV = 0; BRK13 = 0; LBKDE = 0

Analysis. How the UART0 is initialized for baud rate depends on the KL05 clock configuration and the desired baud rate. The KL05 assembly language program template provided for this course masks all maskable interrupts and calls the Startup subroutine, (in the Start module provided for this course), before any other program code gets executed. Startup calls the SetClock48MHz subroutine that configures the multipurpose clock generator (MCG) module to use a 32.768-kHz clock from the crystal oscillator, generate a 48-MHz FLL clock (MCGFLLCLK), and establish a 48-MHz core clock for the KL05.

Before UART0 can be initialized, its clock must be selected and enabled. Since it is to be used with an external device through the OpenSDA connector of the Freedom Board, UART0 must be configured for external connection, two KL05 pins must be configured for a serial port connection through the OpenSDA connector on the board, and the KL05 port(s) to which these pins belong must have their clock(s) enabled. Since the KL05Z provides an OpenSDA connection with KL05 port B pins 1 and 2, the port B clock must be enabled. Also, port B pin 2 (PTB2) must be connected to the Rx pin of UART0 (UART0_RX), and port B pin 1 (PTB1) must be connected to the Tx pin of UART0 (UART0_TX). (Port B pins 1 and 2 are also connected to pins 2 and 1, respectively, of the KL05Z J8 connector.) The EQUates and program code that follow are an example of the initialization required for the UART0 external connection, UART0 module clock (MCGFLLCLK, which gives 48MHz), port B module clock, and port B pins.

In addition to the initialization to make the UART0 Rx and Tx signals available through Port B to the OpenSDA connector, UART0 needs to be configured for the proper serial timing and protocol to communicate with the connected PC through PuTTY (or another serial terminal program). SBR (13-bit value in the baud rate registers) and OSR (5-bit value in control register 4) control the timing, M (control register 1 bit 4) and M10 (control register 4 bit 5) determine the number of data bits per character, PE (control register 1 bit 1) sets the parity, SBNS (baud rate register high bit 5) sets the number of stop bits, RE (control register 2 bit 2) enables the receiver, and TE (control register 2 bit 3) enables the transmitter. In this example, UART0 is configured for PuTTY's default serial connection settings: 9600 baud, 8 data bits, no parity bit, and 1 stop bit.

The baud rate is determined by the UART0 module clock, SBR, and OSR, as expressed by the following equation.

$$BaudRate = \frac{UART0ModuleClock}{(OSR+1)\times SBR}$$

As part of the previous initialization steps, the MCG 48-MHz FLL clock was selected as the UART0 module clock. The default value of OSR is 15. Given these two values, the equation above can be solved for the 13-bit integer value of SBR to produce a baud rate of 9600.

$$SBR = \left\lfloor \frac{UART0ModuleClock}{(OSR+1)\times BaudRate} \right\rfloor = \left\lfloor \frac{48\times 10^6}{16\times 9600} \right\rfloor = \lfloor 312.5 \rfloor = 312 = 0\text{x}138$$

Thus, for a 9600-baud serial connection with 8 data bits, no parity, and 1 stop bit (8N1), the baud rate register high should be set to 0x01 (0 for SBNS and 1 for SBR[12:8]), and the baud rate register low should be set to 0x38 (SBR[7:0]). Control register 1 needs to be set to 0x00, (0 for both M and PE), and control register 4 needs to be set to 0x0F, (0 for M10 and 15 for OSR). Control registers 3 and 5 should also be set to 0x00. After the serial baud rate and protocol are set in these control registers, UART0's receiver and transmitter need to be enabled by setting control register 2 to 0x0C, (1 for TE and 1 for RE).

Assembly language code. Using the EQUates below, the program code that follows initializes UART0 for polled serial I/O through the KL05Z USB virtual serial port with eight data bits, no parity, and one stop bit at 9600 baud. All symbols not defined in the EQUates here are from the RIT CMPE-250 include file MKL05Z4.s

```
;-------------------------------------------------------------
;PORTx_PCRn (Port x pin control register n [for pin n])
;___->10-08:Pin mux control (select 0 to 8)
;Use provided PORT_PCR_MUX_SELECT_2_MASK
;-------------------------------------------------------------
;Port B
PORT_PCR_SET_PTB2_UART0_RX  EQU  (PORT_PCR_ISF_MASK :OR: \
                                  PORT_PCR_MUX_SELECT_2_MASK)
PORT_PCR_SET_PTB1_UART0_TX  EQU  (PORT_PCR_ISF_MASK :OR: \
                                  PORT_PCR_MUX_SELECT_2_MASK)
;-------------------------------------------------------------
;SIM_SCGC4
;1->10:UART0 clock gate control (enabled)
;Use provided SIM_SCGC4_UART0_MASK
;-------------------------------------------------------------
;SIM_SCGC5
;1->10:Port B clock gate control (enabled)
;Use provided SIM_SCGC5_PORTB_MASK
;-------------------------------------------------------------
;SIM_SOPT2
;01=27-26:UART0SRC=UART0 clock source select (MCGFLLCLK)
;-------------------------------------------------------------
SIM_SOPT2_UART0SRC_MCGFLLCLK  EQU  \
                                (1 << SIM_SOPT2_UART0SRC_SHIFT)
;-------------------------------------------------------------
;SIM_SOPT5
; 0->   16:UART0 open drain enable (disabled)
; 0->   02:UART0 receive data select (UART0_RX)
;00->01-00:UART0 transmit data select source (UART0_TX)
SIM_SOPT5_UART0_EXTERN_MASK_CLEAR  EQU  \
                               (SIM_SOPT5_UART0ODE_MASK :OR: \
                                SIM_SOPT5_UART0RXSRC_MASK :OR: \
                                SIM_SOPT5_UART0TXSRC_MASK)
```

```
;--------------------------------------------------------------
;UART0_BDH
;    0->  7:LIN break detect IE (disabled)
;    0->  6:RxD input active edge IE (disabled)
;    0->  5:Stop bit number select (1)
;00001->4-0:SBR[12:0] (UART0CLK / [9600 * (OSR + 1)])
;UART0CLK is MCGFLLCLK
;MCGPLLCLK is 47972352 Hz ~=~ 48 MHz
;SBR ~=~ 48 MHz / (9600 * 16) = 312.5 --> 312 = 0x138
;SBR = 47972352 / (9600 * 16) = 312.32 --> 312 = 0x138
UART0_BDH_9600  EQU  0x01
;--------------------------------------------------------------
;UART0_BDL
;26->7-0:SBR[7:0] (UART0CLK / [9600 * (OSR + 1)])
;UART0CLK is MCGFLLCLK
;MCGPLLCLK is 47972352 Hz ~=~ 48 MHz
;SBR ~=~ 48 MHz / (9600 * 16) = 312.5 --> 312 = 0x138
;SBR = 47972352 / (9600 * 16) = 312.32 --> 312 = 0x138
UART0_BDL_9600  EQU  0x38
;--------------------------------------------------------------
;UART0_C1
;0-->7:LOOPS=loops select (normal)
;0-->6:DOZEEN=doze enable (disabled)
;0-->5:RSRC=receiver source select (internal--no effect LOOPS=0)
;0-->4:M=9- or 8-bit mode select
;        (1 start, 8 data [lsb first], 1 stop)
;0-->3:WAKE=receiver wakeup method select (idle)
;0-->2:IDLE=idle line type select (idle begins after start bit)
;0-->1:PE=parity enable (disabled)
;0-->0:PT=parity type (even parity--no effect PE=0)
UART0_C1_8N1  EQU  0x00
;--------------------------------------------------------------
;UART0_C2
;0-->7:TIE=transmit IE for TDRE (disabled)
;0-->6:TCIE=transmission complete IE for TC (disabled)
;0-->5:RIE=receiver IE for RDRF (disabled)
;0-->4:ILIE=idle line IE for IDLE (disabled)
;1-->3:TE=transmitter enable (enabled)
;1-->2:RE=receiver enable (enabled)
;0-->1:RWU=receiver wakeup control (normal)
;0-->0:SBK=send break (disabled, normal)
UART0_C2_T_R  EQU  (UART0_C2_TE_MASK :OR: UART0_C2_RE_MASK)
;--------------------------------------------------------------
;UART0_C3
;0-->7:R8T9=9th data bit for receiver (not used M=0)
;          10th data bit for transmitter (not used M10=0)
;0-->6:R9T8=9th data bit for transmitter (not used M=0)
;          10th data bit for receiver (not used M10=0)
;0-->5:TXDIR=UART_TX pin direction in single-wire mode
;           (no effect LOOPS=0)
;0-->4:TXINV=transmit data inversion (not inverted)
;0-->3:ORIE=overrun IE for OR (disabled)
;0-->2:NEIE=noise error IE for NF (disabled)
;0-->1:FEIE=framing error IE for FE (disabled)
;0-->0:PEIE=parity error IE for PF (disabled)
UART0_C3_NO_TXINV  EQU  0x00
```

```
;--------------------------------------------------------------
;UART0_C4
;    0-->  7:MAEN1=match address mode enable 1 (disabled)
;    0-->  6:MAEN2=match address mode enable 2 (disabled)
;    0-->  5:M10=10-bit mode select (not selected)
;01111-->4-0:OSR=over sampling ratio (16)
;               = 1 + OSR for 3 <= OSR <= 31
;               = 16 for 0 <= OSR <= 2 (invalid values)
UART0_C4_OSR_16           EQU  0x0F
UART0_C4_NO_MATCH_OSR_16  EQU  UART0_C4_OSR_16
;--------------------------------------------------------------
;UART0_C5
;  0-->  7:TDMAE=transmitter DMA enable (disabled)
;  0-->  6:Reserved; read-only; always 0
;  0-->  5:RDMAE=receiver full DMA enable (disabled)
;000-->4-2:Reserved; read-only; always 0
;  0-->  1:BOTHEDGE=both edge sampling (rising edge only)
;  0-->  0:RESYNCDIS=resynchronization disable (enabled)
UART0_C5_NO_DMA_SSR_SYNC  EQU  0x00
;--------------------------------------------------------------
;UART0_S1
;0-->7:TDRE=transmit data register empty flag; read-only
;0-->6:TC=transmission complete flag; read-only
;0-->5:RDRF=receive data register full flag; read-only
;1-->4:IDLE=idle line flag; write 1 to clear (clear)
;1-->3:OR=receiver overrun flag; write 1 to clear (clear)
;1-->2:NF=noise flag; write 1 to clear (clear)
;1-->1:FE=framing error flag; write 1 to clear (clear)
;1-->0:PF=parity error flag; write 1 to clear (clear)
UART0_S1_CLEAR_FLAGS  EQU  (UART0_S1_IDLE_MASK :OR: \
                            UART0_S1_OR_MASK :OR: \
                            UART0_S1_NF_MASK :OR: \
                            UART0_S1_FE_MASK :OR: \
                            UART0_S1_PF_MASK)
;--------------------------------------------------------------
;UART0_S2
;1-->7:LBKDIF=LIN break detect interrupt flag (clear)
;             write 1 to clear
;1-->6:RXEDGIF=RxD pin active edge interrupt flag (clear)
;              write 1 to clear
;0-->5:(reserved); read-only; always 0
;0-->4:RXINV=receive data inversion (disabled)
;0-->3:RWUID=receive wake-up idle detect
;0-->2:BRK13=break character generation length (10)
;0-->1:LBKDE=LIN break detect enable (disabled)
;0-->0:RAF=receiver active flag; read-only
UART0_S2_NO_RXINV_BRK10_NO_LBKDETECT_CLEAR_FLAGS  EQU  \
        (UART0_S2_LBKDIF_MASK :OR: UART0_S2_RXEDGIF_MASK)
;--------------------------------------------------------------
```

```
;Select MCGFLLCLK as UART0 clock source
    LDR   Ri,=SIM_SOPT2
    LDR   Rj,=SIM_SOPT2_UART0SRC_MASK
    LDR   Rk,[Ri,#0]
    BICS  Rk,Rk,Rj
    LDR   Rj,=SIM_SOPT2_UART0SRC_MCGFLLCLK
    ORRS  Rk,Rk,Rj
    STR   Rk,[Ri,#0]
;Set UART0 for external connection
    LDR   Ri,=SIM_SOPT5
    LDR   Rj,=SIM_SOPT5_UART0_EXTERN_MASK_CLEAR
    LDR   Rk,[Ri,#0]
    BICS  Rk,Rk,Rj
    STR   Rk,[Ri,#0]
;Enable UART0 module clock
    LDR   Ri,=SIM_SCGC4
    LDR   Rj,=SIM_SCGC4_UART0_MASK
    LDR   Rk,[Ri,#0]
    ORRS  Rk,Rk,Rj
    STR   Rk,[Ri,#0]
;Enable PORT B module clock
    LDR   Ri,=SIM_SCGC5
    LDR   Rj,=SIM_SCGC5_PORTB_MASK
    LDR   Rk,[Ri,#0]
    ORRS  Rk,Rk,Rj
    STR   Rk,[Ri,#0]
;Select PORT B Pin 2 (D0) for UART0 RX (J8 Pin 01)
    LDR     Ri,=PORTB_PCR2
    LDR     Rj,=PORT_PCR_SET_PTB2_UART0_RX
    STR     Rj,[Ri,#0]
; Select PORT B Pin 1 (D1) for UART0 TX (J8 Pin 02)
    LDR     Ri,=PORTB_PCR1
    LDR     Rj,=PORT_PCR_SET_PTB1_UART0_TX
    STR     Rj,[Ri,#0]
;Disable UART0 receiver and transmitter
    LDR   Ri,=UART0_BASE
    MOVS  Rj,#UART0_C2_T_R
    LDRB  Rk,[Ri,#UART0_C2_OFFSET]
    BICS  Rk,Rk,Rj
    STRB  Rk,[Ri,#UART0_C2_OFFSET]
;Set UART0 for 9600 baud, 8N1 protocol
    MOVS  Rj,#UART0_BDH_9600
    STRB  Rj,[Ri,#UART0_BDH_OFFSET]
    MOVS  Rj,#UART0_BDL_9600
    STRB  Rj,[Ri,#UART0_BDL_OFFSET]
    MOVS  Rj,#UART0_C1_8N1
    STRB  Rj,[Ri,#UART0_C1_OFFSET]
    MOVS  Rj,#UART0_C3_NO_TXINV
    STRB  Rj,[Ri,#UART0_C3_OFFSET]
    MOVS  Rj,#UART0_C4_NO_MATCH_OSR_16
    STRB  Rj,[Ri,#UART0_C4_OFFSET]
    MOVS  Rj,#UART0_C5_NO_DMA_SSR_SYNC
    STRB  Rj,[Ri,#UART0_C5_OFFSET]
    MOVS  Rj,#UART0_S1_CLEAR_FLAGS
    STRB  Rj,[Ri,#UART0_S1_OFFSET]
    MOVS  Rj, \
          #UART0_S2_NO_RXINV_BRK10_NO_LBKDETECT_CLEAR_FLAGS
    STRB  Rj,[Ri,#UART0_S2_OFFSET]
```

```
    ;Enable UART0 receiver and transmitter
        MOVS  Rj,#UART0_C2_T_R
        STRB  Rj,[Ri,#UART0_C2_OFFSET]
```

C code. Using the #defines below, the program code that follows initializes UART0 for polled serial I/O through the KL05Z USB virtual serial port with eight data bits, no parity, and one stop bit at 9600 baud. All symbols not defined in the #defines here are from the Keil NXP (Freescale) include file MKL05Z4.h

```
/*--------------------------------------------------------------*/
/* PORTx_PCRn (Port x pin control register n [for pin n])       */
/* 10-08:MUX=Pin mux control (select 0 to 8)                    */
/*--------------------------------------------------------------*/
#define PORT_PCR_MUX_SELECT_2_MASK (2u << PORT_PCR_MUX_SHIFT)
/*--------------------------------------------------------------*/
/* Port B                                                       */
/*--------------------------------------------------------------*/
#define PORT_PCR_PTB2_MUX_UART0_RX (PORT_PCR_MUX_SELECT_2_MASK)
#define PORT_PCR_PTB1_MUX_UART0_TX (PORT_PCR_MUX_SELECT_2_MASK)
#define PORT_PCR_SET_PTB2_UART0_RX (PORT_PCR_ISF_MASK | \
                                    PORT_PCR_MUX_SELECT_2_MASK)
#define PORT_PCR_SET_PTB1_UART0_TX (PORT_PCR_ISF_MASK | \
                                    PORT_PCR_MUX_SELECT_2_MASK)
/*--------------------------------------------------------------*/
/* SIM_SCGC4                                                    */
/* 1->10:UART0 clock gate control (enabled)                     */
/*--------------------------------------------------------------*/
/* Use provided SIM_SCGC4_UART0_MASK                            */
/*--------------------------------------------------------------*/
/* SIM_SCGC5                                                    */
/* 1->09:Port A clock gate control (enabled)                    */
/*--------------------------------------------------------------*/
/* Use provided SIM_SCGC5_PORTA_MASK                            */
/*--------------------------------------------------------------*/
/* SIM_SOPT2                                                    */
/* 01=27-26:UART0SRC=UART0 clock source select                  */
/*          PLLFLLSEL determines MCGFLLCLK' or MCGPLLCLK/2      */
/*--------------------------------------------------------------*/
#define SIM_SOPT2_UART0SRC_MCGFLLCLK \
                                (1u << SIM_SOPT2_UART0SRC_SHIFT)
/*--------------------------------------------------------------*/
/* SIM_SOPT5                                                    */
/*  0->   16:UART0 open drain enable (disabled)                 */
/*  0->   02:UART0 receive data select (UART0_RX)               */
/* 00->01-00:UART0 transmit data select source (UART0_TX)       */
/*--------------------------------------------------------------*/
#define SIM_SOPT5_UART0_EXTERN_MASK_CLEAR  \
                                  (SIM_SOPT5_UART0ODE_MASK | \
                                   SIM_SOPT5_UART0RXSRC_MASK | \
                                   SIM_SOPT5_UART0TXSRC_MASK)
```

```
/*-------------------------------------------------------------*/
/* UART0_BDH                                                   */
/*     0->  7:LIN break detect IE (disabled)                   */
/*     0->  6:RxD input active edge IE (disabled)              */
/*     0->  5:Stop bit number select (1)                       */
/* 00001->4-0:SBR[12:0] (MCGFLLCLK / [9600 * (OSR + 1)])       */
/* MCGFLLCLK is 47972352 Hz ~=~ 48 MHz                         */
/* SBR ~=~ 48 MHz / (9600 * 16) = 312.5 --> 312                */
/* SBR = 47972352 / (9600 * 16) = 312.32 --> 312 = 0x138       */
/*-------------------------------------------------------------*/
#define UART0_BDH_9600  (0x01u)
/*-------------------------------------------------------------*/
/* UART0_BDL                                                   */
/* 26->7-0:SBR[7:0] (MCGFLLCLK / [9600 * (OSR + 1)])           */
/* MCGFLLCLK is 47972352 Hz ~=~ 48 MHz                         */
/* SBR ~=~ 48 MHz / (9600 * 16) = 312.5 --> 312                */
/* SBR = 47972352 / (9600 * 16) = 312.32 --> 312 = 0x138       */
/*-------------------------------------------------------------*/
#define UART0_BDL_9600  (0x38u)
/*-------------------------------------------------------------*/
/* UART0_C1                                                    */
/* 0-->7:LOOPS=loops select (normal)                           */
/* 0-->6:DOZEEN=doeze enable (disabled)                        */
/* 0-->5:RSRC=receiver source select                           */
/*            (internal--no effect LOOPS=0)                    */
/* 0-->4:M=9- or 8-bit mode select                             */
/*        (1 start, 8 data [lsb first], 1 stop)                */
/* 0-->3:WAKE=receiver wakeup method select (idle)             */
/* 0-->2:IDLE=idle line type select                            */
/*           (idle begins after start bit)                     */
/* 0-->1:PE=parity enable (disabled)                           */
/* 0-->0:PT=parity type (even parity--no effect PE=0)          */
/*-------------------------------------------------------------*/
#define UART0_C1_8N1  (0x00)
/*-------------------------------------------------------------*/
/*UART0_C2                                                     */
/*0-->7:TIE=transmit IE for TDRE (disabled)                    */
/*0-->6:TCIE=transmission complete IE for TC (disabled)        */
/*0-->5:RIE=receiver IE for RDRF (disabled)                    */
/*0-->4:ILIE=idle line IE for IDLE (disabled)                  */
/*1-->3:TE=transmitter enable (enabled)                        */
/*1-->2:RE=receiver enable (enabled)                           */
/*0-->1:RWU=receiver wakeup control (normal)                   */
/*0-->0:SBK=send break (disabled, normal)                      */
/*-------------------------------------------------------------*/
#define UART0_C2_T_R  (UART0_C2_TE_MASK | UART0_C2_RE_MASK)
```

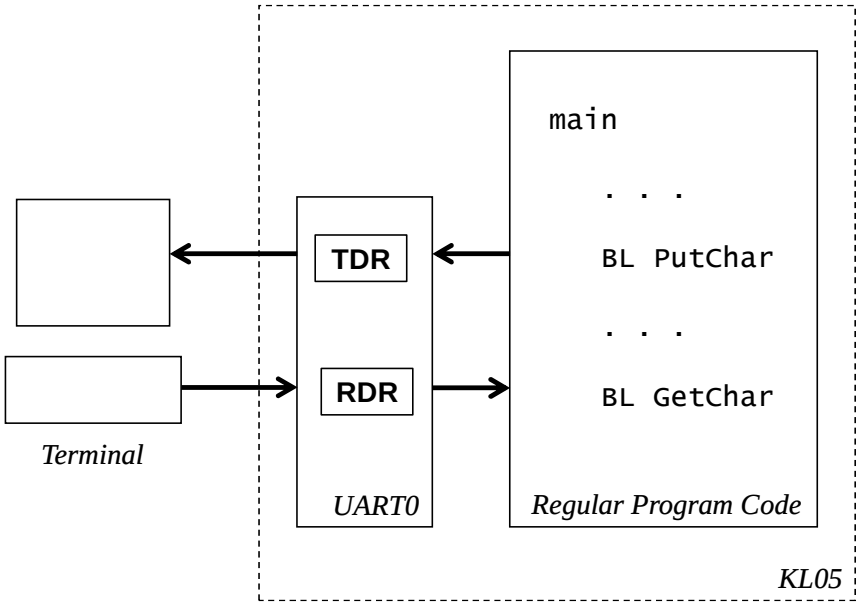
```
/*-------------------------------------------------------------*/
/* UART0_C3                                                    */
/* 0-->7:R8T9=9th data bit for receiver (not used M=0)         */
/*           10th data bit for transmitter (not used M10=0)    */
/* 0-->6:R9T8=9th data bit for transmitter (not used M=0)      */
/*           10th data bit for receiver (not used M10=0)       */
/* 0-->5:TXDIR=UART_TX pin direction in single-wire mode       */
/*            (no effect LOOPS=0)                              */
/* 0-->4:TXINV=transmit data inversion (not inverted)          */
/* 0-->3:ORIE=overrun IE for OR (disabled)                     */
/* 0-->2:NEIE=noise error IE for NF (disabled)                 */
/* 0-->1:FEIE=framing error IE for FE (disabled)               */
/* 0-->0:PEIE=parity error IE for PF (disabled)                */
/*-------------------------------------------------------------*/
#define UART0_C3_NO_TXINV  (0x00)
/*-------------------------------------------------------------*/
/* UART0_C4                                                    */
/*     0-->  7:MAEN1=match address mode enable 1 (disabled)    */
/*     0-->  6:MAEN2=match address mode enable 2 (disabled)    */
/*     0-->  5:M10=10-bit mode select (not selected)           */
/* 01111-->4-0:OSR=over sampling ratio (16)                    */
/*                 = 1 + OSR for 3 <= OSR <= 31                */
/*                 = 16 for 0 <= OSR <= 2 (invalid values)     */
/*-------------------------------------------------------------*/
#define UART0_C4_OSR_16  (0x0Fu)
#define UART0_C4_NO_MATCH_OSR_16  (UART0_C4_OSR_16)
/*-------------------------------------------------------------*/
/* UART0_C5                                                    */
/*   0-->  7:TDMAE=transmitter DMA enable (disabled)           */
/*   0-->  6:Reserved; read-only; always 0                     */
/*   0-->  5:RDMAE=receiver full DMA enable (disabled)         */
/* 000-->4-2:Reserved; read-only; always 0                     */
/*   0-->  1:BOTHEDGE=both edge sampling (rising edge only)    */
/*   0-->  0:RESYNCDIS=resynchronization disable (enabled)     */
/*-------------------------------------------------------------*/
#define UART0_C5_NO_DMA_SSR_SYNC  (0x00)
/*-------------------------------------------------------------*/
/* UART0_S1                                                    */
/* 0-->7:TDRE=transmit data register empty flag; read-only     */
/* 0-->6:TC=transmission complete flag; read-only              */
/* 0-->5:RDRF=receive data register full flag; read-only       */
/* 1-->4:IDLE=idle line flag; write 1 to clear (clear)         */
/* 1-->3:OR=receiver overrun flag; write 1 to clear (clear)    */
/* 1-->2:NF=noise flag; write 1 to clear (clear)               */
/* 1-->1:FE=framing error flag; write 1 to clear (clear)       */
/* 1-->0:PF=parity error flag; write 1 to clear (clear)        */
/*-------------------------------------------------------------*/
#define UART0_S1_CLEAR_FLAGS  (UART0_S1_IDLE_MASK | \
                               UART0_S1_OR_MASK | \
                               UART0_S1_NF_MASK | \
                               UART0_S1_FE_MASK | \
                               UART0_S1_PF_MASK)
```

```
/*-------------------------------------------------------------*/
/* UART0_S2                                                    */
/* 1-->7:LBKDIF=LIN break detect interrupt flag (clear)        */
/*              write 1 to clear                               */
/* 1-->6:RXEDGIF=RxD pin active edge interrupt flag (clear)    */
/*               write 1 to clear                              */
/* 0-->5:(reserved); read-only; always 0                       */
/* 0-->4:RXINV=receive data inversion (disabled)               */
/* 0-->3:RWUID=receive wake-up idle detect                     */
/* 0-->2:BRK13=break character generation length (10)          */
/* 0-->1:LBKDE=LIN break detect enable (disabled)              */
/* 0-->0:RAF=receiver active flag; read-only                   */
/*-------------------------------------------------------------*/
#define UART0_S2_NO_RXINV_BRK10_NO_LBKDETECT_CLEAR_FLAGS  ( \
               UART0_S2_LBKDIF_MASK | UART0_S2_RXEDGIF_MASK)

  /* Select MCGFLLCLK as UART0 clock source */
  SIM->SOPT2 &= ~SIM_SOPT2_UART0SRC_MASK;
  SIM->SOPT2 |= SIM_SOPT2_UART0SRC_MCGFLLCLK;
  /* Set UART0 for external connection */
  SIM->SOPT5 &= ~SIM_SOPT5_UART0_EXTERN_MASK_CLEAR;
  /* Enable UART0 module clock */
  SIM->SCGC4 |= SIM_SCGC4_UART0_MASK;
  /* Some OpenSDA applications provide a virtual serial port */
  /* through the OpenSDA USB connection using PTA1 and PTA2  */
  /* Enable PORT B module clock */
  SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK;
  /* Select PORT B Pin 2 (D0) for UART0 RX */
  PORTB->PCR[2] = PORT_PCR_SET_PTB2_UART0_RX;
  /* Select PORT B Pin 1 (D1) for UART0 TX */
  PORTB->PCR[1] = PORT_PCR_SET_PTB1_UART0_TX;
  /* Set for 9600 baud from 96MHz PLL clock */
  UART0->C2 &= ~UART0_C2_T_R;  /* disable UART0 */
  UART0->BDH = UART0_BDH_9600;
  UART0->BDL = UART0_BDL_9600;
  UART0->C1 = UART0_C1_8N1;
  UART0->C3 = UART0_C3_NO_TXINV;
  UART0->C4 = UART0_C4_NO_MATCH_OSR_16;
  UART0->C5 = UART0_C5_NO_DMA_SSR_SYNC;
  UART0->S1 = UART0_S1_CLEAR_FLAGS;
  UART0->S2 = UART0_S2_NO_RXINV_BRK10_NO_LBKDETECT_CLEAR_FLAGS;
  UART0->C2 = UART0_C2_T_R;  /* enable UART0 */
```

UART0 polled transmit and receive

Once a program has initialized UART0 for the desired serial baud rate and serial protocol and has enabled it to transmit and receive, the program may use UART0 to send and receive data. The figure below shows the overall system configuration for polled serial I/O. The examples that follow use R*i* to hold the character for character I/O.



KL05 System Configuration for Terminal I/O through UART0 with Polling

Transmit. A polled transmit, (i.e., send character), polls UART0 status register 1 for the transmit data register empty (TDRE) condition in bit 7. If TDRE = 1, a byte may be transmitted (i.e., sent); otherwise, UART0 is not yet ready to transmit a byte, and the status register must be polled until TDRE = 1.

Assembly language code: The following code implements a polled transmit for a single character—the basis for a "PutChar" operation.

```
;Poll TDRE until UART0 ready to transmit
         LDR   Rj,=UART0_BASE
         MOVS  Rk,#UART0_S1_TDRE_MASK
PollTx   LDRB  Rm,[Rj,#UART0_S1_OFFSET]
         ANDS  Rm,Rm,Rk
         BEQ   PollTx
;Transmit character stored in Ri
         STRB  Ri,[Rj,#UART0_D_OFFSET]
```

C code: The following PutChar function implements a polled transmit for a single character.

```
void PutChar (char Character) {
/*****************************************************************/
/* Puts Character to UART0 using polling.                        */
/* Waits until UART0 TDRE and then puts Character into           */
/* UART0 TDR.                                                    */
/*****************************************************************/
  while (!(UART0->S1 & UART0_S1_TDRE_MASK));
  UART0->D = Character;
}
```

Receive. Similarly, a polled receive, (i.e., get character), polls UART0 status register 1 for the receive data register full (RDRF) condition in bit 5. If RDRF = 1, a byte may be read by the KL05, (i.e., received); otherwise, UART0 has not yet received a byte, and the status register must be polled until RDRF = 1.

Assembly language code: The following code implements a polled receive for a single character—the basis for a "GetChar" operation.

```
     ;Poll RDRF until UART0 ready to receive
              LDR   Rj,=UART0_BASE
              MOVS  Rk,#UART0_S1_RDRF_MASK
     PollRx   LDRB  Rm,[Rj,#UART0_S1_OFFSET]
              ANDS  Rm,Rm,Rk
              BEQ   PollRx
     ;Receive character and store in Ri
              LDRB  Ri,[Rj,#UART0_D_OFFSET]
```

C code: The following GetChar function implements a polled receive for a single character.

```
char GetChar (void) {
/*****************************************************************/
/* Gets a character from UART0 using polling.                    */
/* Waits until UART0 RDRF and then gets a character from         */
/* UART0 RDR.                                                    */
/*****************************************************************/
  while (!(UART0->S1 & UART0_S1_RDRF_MASK));
  return (UART0->D);
}
```

Receive buffer overrun: Since UART0 has only a one-character receive buffer, (the receive data register), receive buffer overrun occurs whenever two characters are received before one is read from UART0. When this happens, the receiver overrun flag OR in UART0_S1 gets set, and afterward any new UART0 characters received are lost, (i.e., polled receive will no longer work). If it is anticipated that a long time will elapse between polled receive operations, OR should be cleared prior to attempting a polled receive operation to ensure input characters can continue to be received. (Note that characters received during the overrun condition will have been lost.) OR can be cleared by writing a 1 to it.

- Assembly language code:

```
LDR   Rj,=UART0_BASE
MOVS  Rk,#UART0_S1_OR_MASK       ;OR mask
STRB  Rk,[Rj,#UART0_S1_OFFSET]   ;Clear OR
```

- C code:

```
UART0->S1 = UART0_S1_OR_MASK;
```