NEC

Application Note

78K0R/Kx3-L/Ix3

16-Bit Single-Chip Microcontrollers

STOP mode release by frame reception of the Serial Array Unit in UART mode

---

78K0R/KC3-L
78K0R/KD3-L
78K0R/KE3-L
78K0R/IB3
78K0R/IC3
78K0R/ID3
78K0R/IE3

Document No. U19742EE1V0IF00
Date Published April 2009

Table of Contents

(A) Introduction

This document contains additional information on the usage of the Serial Array Unit SAU in UART mode during STOP mode without switching off the Serial Array Unit before entering the STOP mode. The Serial Array Unit and the CPU are supplied by the 8MHz internal high-speed oscillator

In a lot of low power applications it is necessary to release the microcontroller from STOP mode on frame reception of the Serial Array Unit UART mode.

Scope of this document is the description of how to use the Serial Array Unit for STOP mode release without missing frames.

(B) Related Products

78K0R/KC3-L:
µPD78F1000, µPD78F1001, µPD78F1002, µPD78F1003

78K0R/KD3-L:
µPD78F1004, µPD78F1005, µPD78F1006

78K0R/KE3-L:
µPD78F1007, µPD78F1008, µPD78F1009

78K0R/IB3:
µPD78F1201, µPD78F1203

78K0R/IC3:
µPD78F1211, µPD78F1213, µPD78F1214, µPD78F1215

78K0R/ID3:
µPD78F1223, µPD78F1224, µPD78F1225

78K0R/IE3:
µPD78F1233, µPD78F1234, µPD78F1235

(C) Introduction

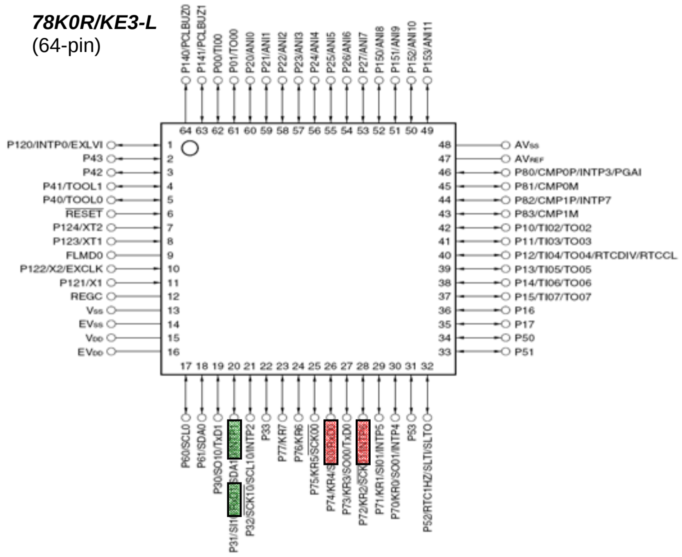
(C.1.) STOP mode release

On STOP mode release by frame reception, special care has to be taken with the Serial Array Unit macro.

A STOP mode release by frame reception is not directly possible, because the Serial Array Unit can only be operated on the main system clock, if feasible baud rates for UART communication shall be achieved. In STOP mode, the main system clock is switched off and therefore the Serial Array Unit operation is stopped.

Due to this, a second interrupt source is necessary for STOP mode release, which is independent from the clock source supply. Here the external interrupt would be a suitable source, either due to the fact, that one external interrupt pin might be shared with a Serial Array Unit UART receive pin or that an external interrupt pin is available closed to a Serial Array Unit UART receive pin (Figure 1).

Figure 1 Example Pin Configuration



INTPx and RxD shared on one pin

INTPx and RxD on two separate pins

(C.2) External Interrupts

External maskable interrupt requests are generated on rising edge, falling edge or both edges on the according port pin.

The valid edges will be specified by the External Interrupt Rising Edge Enable Register (EGP0) and/or the External Interrupt Falling Edge Enable Register (EGN0).

Figure 2 Format of External Interrupt Rising Edge Enable Register (EGP0) and Format of External Interrupt Rising Edge Enable Register (EGN0)

Address: FFF38H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
EGP0	EGP7	EGP6	EGP5	EGP4	EGP3	EGP2	EGP1	EGP0

Address: FFF39H After reset: 00H R/W

Symbol	7	6	5	4	3	2	1	0
EGN0	EGN7	EGN6	EGN5	EGN4	EGN3	EGN2	EGN1	EGN0

EGPn	EGNn	INTPn pin valid edge selection (n = 0 to 7)
0	0	Edge detection disabled
0	1	Falling edge
1	0	Rising edge
1	1	Both rising and falling edges

(D) Software Application Example

As mentioned in Chapter (C.1) STOP mode release, a STOP mode release by UART frame reception is only possible by using a second interrupt source, which is independent from the clock supply. In this example the external interrupt INTP1 is used for a STOP mode release by frame reception of the Serial Array Unit UART1.

(D.1) Initialization of the Clock Generator for 8MHz internal high-speed osc. operation

On the 78K0R/Kx3-L Series, the clock supply after reset release must be specified by the User Option Byte setting of byte 000C1H/010C1H

Figure 3 Format of User Option Byte (000C1H/010C1H) on 78K0R/Kx3-L Series

Address: 000C1H/010C1H Note 1

7	6	5	4	3	2	1	0
1	1	1	1	1	FRQSEL2	FRQSEL1	LVIOFF

FRQSEL2	FRQSEL1	Internal high-speed oscillator frequency
0	1	8 MHz/20 MHz Note 2
1	0	1 MHz Note 3
Other than the above		Setting prohibited

LVIOFF	Setting of LVI on power application
0	LVI is ON by default (LVI default start function enabled) upon reset release (upon power application)
1	LVI is OFF by default (LVI default start function stopped) upon reset release (upon power application)

Notes 1. Set the same value as 000C1H to 010C1H when the boot swap operation is used because 000C1H is replaced by 010C1H.

2. When 8 MHz or 20 MHz has been selected, the 8 MHz internal high-speed oscillator automatically starts oscillating after reset release. To use the 20 MHz internal high-speed oscillator to operate the microcontroller, oscillation is started by setting bit 0 (DSCON) of the 20 MHz internal high-speed oscillation control register (DSCCTL) to 1 with $V_{DD} \geq 2.7$ V. The circuit cannot be changed to a 1 MHz internal high-speed oscillator while the microcontroller operates.

3. When 1 MHz has been selected, the microcontroller operates on the 1 MHz internal high-speed oscillator after reset release. The circuit cannot be changed to an 8 MHz or 20 MHz internal high-speed oscillator while the microcontroller operates.

When the FRQSEL2,FRQSEL1 bits are set to 0, 1, the 8MHz internal high-speed oscillator automatically starts after reset release on the 78K0R/Kx3-L Series.

```
/* ==========================================================================
** option byte definitions
** ==========================================================================
*/
#pragma constseg = OPTBYTE

    __root const char option[4] =
        {
            0x00,   // 000C0H: 00000000 = 0x00
            0xFB,   // 000C1H: 11111101 = 0xFB
            0xFF,   // 000C2H: !!!!! ALWAYS SET TO 0xFF !!!!!
            0x85    // 000C3H: 10000101 = 0x85
        };

#pragma constseg = default
```

For the 78K0R/Ix3 Series is no clock supply related setting necessary for the User Option Byte setting of byte 000C1/010C1H, pls. take care, its setting is according to the User's Manual.

Figure 4 Format of User Option Byte (000C1H/010C1H) on 78K0R/Ix3 Series

Address: 000C1H/010C1H[Note]

7	6	5	4	3	2	1	0
1	1	1	1	1	1	1	LVIOFF

LVIOFF	Setting of LVI on power application
0	LVI is ON by default (LVI default start function enabled) upon reset release (upon power application)
1	LVI is OFF by default (LVI default start function stopped) upon reset release (upon power application)

Note Set the same value as 000C1H to 010C1H when the boot swap operation is used because 000C1H is replaced by 010C1H.

```
/* ==========================================================================
** option byte definitions
** ==========================================================================
*/
#pragma constseg = OPTBYTE

    __root const char option[4] =
        {
            0x00,   // 000C0H: 00000000 = 0x00
            0xFF,   // 000C1H: 11111111 = 0xFF
            0xFF,   // 000C2H: !!!!! ALWAYS SET TO 0xFF !!!!!
            0x85    // 000C3H: 10000101 = 0x85
        };

#pragma constseg = default
```

With the above described User Option Byte settings for the 78K0R/Kx3-L and the 78K0R/Ix3 Series the CPU and the peripherals are supplied by the $f_{CPU} = f_{PER} = f_{IH}/2$ = 8MHz/2 = 4MHz

The division ration of the CPU/peripheral hardware clock can be selected by the bits MDIV2, MDIV1 and MDIV 0 of the System Clock Control Register CKC.

Figure 5 Format of System Clock Control Register (CKC)

Address: FFFA4H After reset: 09H R/W[Note 1]

Symbol	<7>	<6>	<5>	<4>	3	2	1	0
CKC	CLS	CSS	MCS	MCM0	1	MDIV2	MDIV1	MDIV0

CLS	Status of CPU/peripheral hardware clock (f_{CLK})
0	Main system clock (f_{MAIN})
1	Subsystem clock (f_{SUB})

MCS	Status of Main system clock (f_{MAIN})
0	Internal high-speed oscillation clock (f_{IH})
1	High-speed system clock (f_{MX})

CSS	MCM0	MDIV2	MDIV1	MDIV0	Selection of CPU/peripheral hardware clock (f_{CLK})
0	0	0	0	0	f_{IH}
		0	0	1	$f_{IH}/2$ (default)
		0	1	0	$f_{IH}/2^2$
		0	1	1	$f_{IH}/2^3$
		1	0	0	$f_{IH}/2^4$
		1	0	1	$f_{IH}/2^5$
0	1	0	0	0	f_{MX}
		0	0	1	$f_{MX}/2$
		0	1	0	$f_{MX}/2^2$
		0	1	1	$f_{MX}/2^3$
		1	0	0	$f_{MX}/2^4$
		1	0	1	$f_{MX}/2^5$ [Note 2]
1 [Note 3]	× [Note 3]	×	×	×	$f_{SUB}/2$
Other than above					Setting prohibited

Notes **1.** Bits 7 and 5 are read-only.

2. Setting is prohibited when f_{MX} < 4 MHz.

3. Changing the value of the MCM0 bit is prohibited while CSS is set to 1.

Remarks 1. f_{IH}: Internal high-speed oscillation clock frequency

f_{MX}: High-speed system clock frequency

f_{SUB}: Subsystem clock frequency

2. ×: don't care

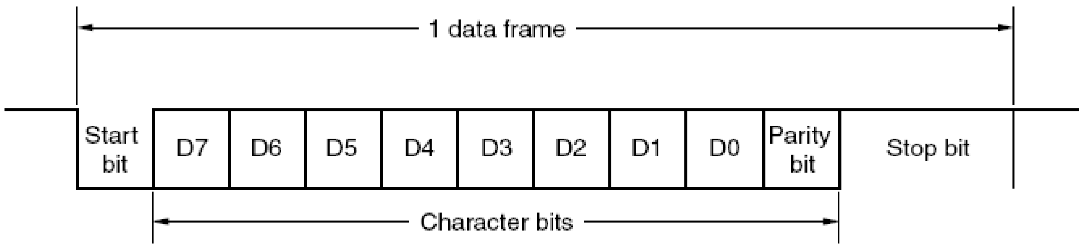
Select the CPU/Peripheral hardware clock to $f_{CPU} = f_{PER} = f_{IH}/1$ = 8MHz/1 = 8MHz by setting bits MDIV2, MDIV1, MDIV0 to 0, 0, 0 (Pls. refer to function main()).

```
void main(void)
{
    ...
    CKC = 0x08;    // Use on-chip high-speed oscillator fCPU = fIH/2^0 = 8MHz
    ...
}
```

(D.2) Initialization and interrupt servicing of INTP1

The external interrupt pin INTP1 is shared with the RxD1 pin of the UART1 of the Serial Array Unit and Port P3.1. Due to this, there is no need to consider extra wirings from INTP1 to RxD1.
A frame received via UART starts with the start bit reception, which normally starts with a falling edge. This falling edge will be detected by the external interrupt INTP1 which will generate an interrupt request.

Figure 6 Data Frame for MSB-first transmission/reception

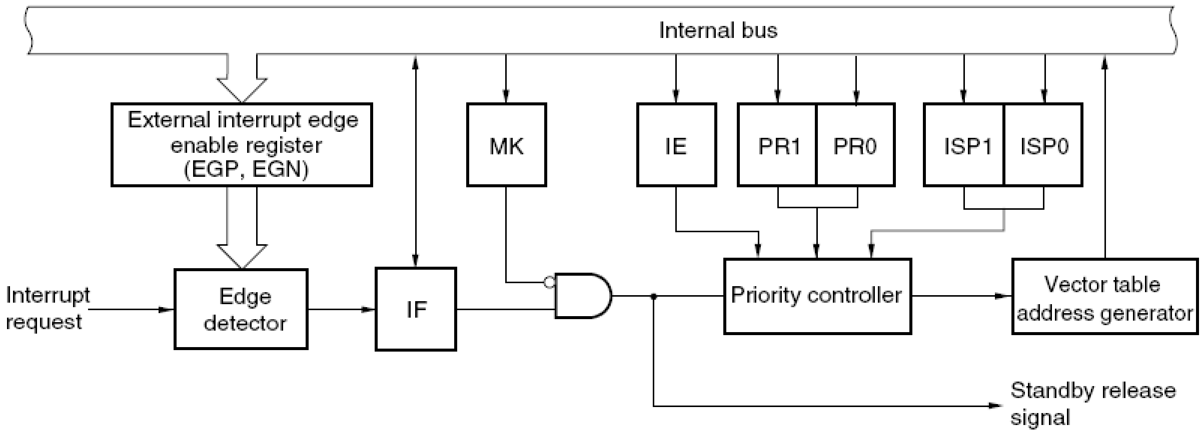


One data frame consists of the following bits.

- Start bit ... 1 bit
- Character bits ... 7 or 8 bits
- Parity bit ... Even parity, odd parity, 0 parity, or no parity
- Stop bit ... 1 or 2 bits

An external interrupt request will only force a STOP mode release if the according interrupt mask flag is enabled.

Figure 7 External maskable interrupt



The external interrupt must be initialized for falling edge detection. Due to the fact, that the external interrupt is only used for STOP mode release the necessary interrupt service routine will be empty. Furthermore, the RxD1/INTP1 pin and the TXD1 pin must be initialized in advance.

```
//*******************************************************************************
//** Port 3 initialization
//*******************************************************************************
void P3Init(void)
{
    // Set port output mode to normal output mode
    POM3_bit0 = 0;

    // Set port latch of TxD1 to 1
    P3_bit.no0 = 1;

    // Set port bit direction of TxD1 to output
    PM3_bit.no0 = 0;

    // Set port input mode to normal input mode
    PIM3_bit1 = 0;

    // Set port bit direction of RxD1/INTP1 to input
    PM3_bit.no1 = 1;

    // Use internal pull-up resistor
    PU3_bit.no1 = 1;
}


//*******************************************************************************
//** External Interrupt INTP1 initialization
//*******************************************************************************
void INTP1Init(void)
{
    // Enable external interrupt INTP1 for falling edge detection
    EGP0 &= 0xFD;
    EGN0 |= 0x02;

    PPR01 = 1;      // Default priority for INTP1 interrupt
    PPR11 = 1;
    PIF1  = 0;      // Clear INTP1 interrupt request flag
```

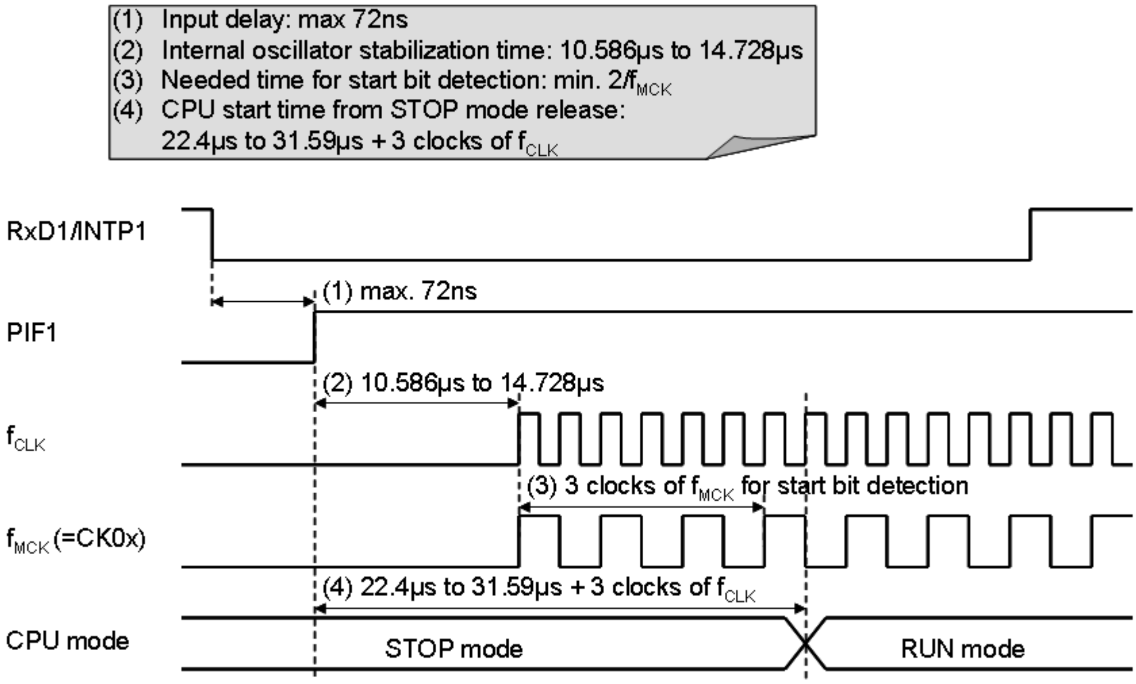
```
    PMK1  = 0;      // Enable INTP1 interrupt
}


//*****************************************************************************
//** External Interrupt INTP1 interrupt service routine
//*****************************************************************************
#pragma bank = 1    // Register bank switching for fast interrupt handling
                    // No pushing general purpose register contents to stack
                    // necessary
#pragma vector = INTP1_vect
__interrupt void IsrINTP1(void)
{
}                   // Empty interrupt service routine
```

(D.3) Initialization and UART1 interrupt servicing of the Serial Array Unit

In this example the UART1 is initialized for a baud rate of 9600 Bd, 8 data bits, no parity and 1 stop bit. The data transmission is done by polling the transmit interrupt request flag and the reception is done by receive interrupt servicing.

For setting up the baud rate special care has to be taken on the start bit detection, because the selected clock supply to the Serial Array Unit will influence secure start bit detection on the desired baud rate.

Figure 8 UART timing in case of STOP mode release on reception



The time for the start bit detection (3) can be shortened by selecting a Serial Array Unit operation clock division for f_{MCK} (CK0x) as small as possible. This can be done by the according setting in the Serial Clock Select Register (SPS0).
The length of the start bit must be at least the sum of the times as defined by (1) to (3).

Figure 9 Serial Clock Select Register 0 (SPS0)

Address: F0126H, F0127H (SPS0), F0166H, F0167H (SPS1) After reset: 0000H R/W

Symbol	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPS0	0	0	0	0	0	0	0	0	PRS 013	PRS 012	PRS 011	PRS 010	PRS 003	PRS 002	PRS 001	PRS 000

PRS 0p3	PRS 0p2	PRS 0p1	PRS 0p0	Section of operation clock (CK0p) Note 1				
					f_{CLK} = 2 MHz	f_{CLK} = 5 MHz	f_{CLK} = 10 MHz	f_{CLK} = 20 MHz
0	0	0	0	f_{CLK}	2 MHz	5 MHz	10 MHz	20 MHz
0	0	0	1	$f_{CLK}/2$	1 MHz	2.5 MHz	5 MHz	10 MHz
0	0	1	0	$f_{CLK}/2^2$	500 kHz	1.25 MHz	2.5 MHz	5 MHz
0	0	1	1	$f_{CLK}/2^3$	250 kHz	625 kHz	1.25 MHz	2.5 MHz
0	1	0	0	$f_{CLK}/2^4$	125 kHz	313 kHz	625 kHz	1.25 MHz
0	1	0	1	$f_{CLK}/2^5$	62.5 kHz	156 kHz	313 kHz	625 kHz
0	1	1	0	$f_{CLK}/2^6$	31.3 kHz	78.1 kHz	156 kHz	313 kHz
0	1	1	1	$f_{CLK}/2^7$	15.6 kHz	39.1 kHz	78.1 kHz	156 kHz
1	0	0	0	$f_{CLK}/2^8$	7.81 kHz	19.5 kHz	39.1 kHz	78.1 kHz
1	0	0	1	$f_{CLK}/2^9$	3.91 kHz	9.77 kHz	19.5 kHz	39.1 kHz
1	0	1	0	$f_{CLK}/2^{10}$	1.95 kHz	4.88 kHz	9.77 kHz	19.5 kHz
1	0	1	1	$f_{CLK}/2^{11}$	977 Hz	2.44 kHz	4.88 kHz	9.77 kHz
1	1	1	1	INTTM02 Note 2				
Other than above				Setting prohibited				

Notes 1. When changing the clock selected for f_{CLK} (by changing the system clock control register (CKC) value), do so after having stopped (ST0 = 000FH) the operation of the serial array unit (SAU). When selecting INTTM02 for the operation clock, also stop the timer array unit TAUS (TT0 = 00FFH).

2. SAU can be operated at a fixed division ratio of the subsystem clock, regardless of the f_{CLK} frequency (main system clock, subsystem clock), by setting the TIS02 bit of the TIS0 register of TAUS to 1, selecting $f_{SUB}/4$ for the input clock, and selecting INTTM02 using the SPS0 register. When changing f_{CLK}, however, SAU and TAUS must be stopped as described in Note 1 above.

Cautions 1. Be sure to clear bits 15 to 8 to "0".

2. After setting the PER0 register to 1, be sure to set the SPS0 register after 4 or more clocks have elapsed.

Remarks 1. f_{CLK}: CPU/peripheral hardware clock frequency

f_{SUB}: Subsystem clock frequency

2. p = 0, 1

```
//***************************************************************************
//** Serial Array Unit UART1 initialization
//***************************************************************************
Void UART1Init(void)
{
    // Switch on serial arry unit 0 input clock
    PER0_bit.no2 = 1;

    // If fCLK = Internal high-speed osc. clock (8MHz (max.)
    // fCK10 and fCK11 = fCLK/2^0 = 8MHz (max.) / 1 = 8MHz
    SPS0 = 0x0000;  // 0000000000000000 = 0000

    // UART mode, start trigger is falling edge of RxD, CK0 clock,
    // Start bit is detected on falling edge of RxD
    SMR02 = 0x0022; // 0000000000100010 = 0x0022
    SMR03 = 0x0122; // 0000000100100010 = 0x0122

    // Transmit and Receive Mode only mode,
    // Baud rate: 57600 Bd,
    // Data bits: 8,
    // Parity:    No,
    // Stop bits: 1,
    // Transmit LSB first
    SCR02 = 0x8097; // 1000000010010111 = 0x8097
    SCR03 = 0x4097; // 0100000010010111 = 0x4097

    // SDR[15...9] = INT[1/2 * ( fCLK/(2^SPS[3...0] * Baud rate) - 1 )] * 2
    // SDR[15...9] = INT[1/2 * ( 8MHz/(2^0 * 57600Bd) - 1)] * 2
    // SDR[15...9] = INT[1/2 * ( 8MHz/307200Bd - 1)] * 2
    // SDR[15...9] = INT[1/2 * ( 26.04Hz/Bd - 1)] * 2
    // SDR[15...9] = INT[12.5] * 2 = 12 * 2
    // SDR[15...9] = 24 = 0x18
    SDR02 = 0x8800; // 1000100000000000 = 0x8800
    SDR03 = 0x8800; // 1000100000000000 = 0x8800

    SOL0 = 0x0000;  // 0000000000000000 = 0x0000
    SO0  = 0x0004;  // 0000000000000100 = 0x0004
    SOE0 = 0x0004;  // 0000000000000100 = 0x0004

    STIF1   = 1;    // Set SAU0 ch-2 UART transmit interrupt request flag
                    // (Transmit path of UART1)
    SRIF1   = 0;    // Clear SAU0 ch-3 UART receive interrupt request flag
                    // (Receive path of UART1)
    SREIF1  = 0;    // Clear SAU0 ch-3 UART receive error int. request flag
                    // (Receive path of UART1 not used)

    STMK1   = 1;    // Mask SAU0 ch-2 UART transmit interrupt
                    // (Transmit path of UART1 not used)
    SRMK1   = 0;    // Unmask SAU0 ch-3 UART receive interrupt
                    // (Receive path of UART1)
    SREMK1  = 0;    // UnMask SAU0 ch-3 UART receive error interrupt
                    // (Receive path of UART1 not used)

    STPR01  = 1;    // Default priority for SAU0 ch-2 UART transmit interrupt
    STPR11  = 1;    // (Transmit path of UART1 not used)
    SRPR01  = 1;    // Default priority for SAU0 ch-3 UART receive interrupt
    SRPR11  = 1;    // (Receive path of UART1)
    SREPR01 = 1;    // Default priority for
    SREPR11 = 1;    // Mask SAU0 ch-3 UART receive error interrupt
```

```
                    // (Receive path of UART1)
    // Start Serial Array Unit 0 channel 2 and 3 operation
    SS0 = 0x000C;   // 0000000000001100 = 0x000C
}

//******************************************************************************
//** Serial Array Unit UART1 data transmission
//******************************************************************************
void DataTransmit(void)
{
    while(!STIF1)              // Wait for transmission is done
        ;
    STIF1=0;                   // Reset transmission done interrupt flag
    SDR02=TxDData;             // Transmit a byte
}

//******************************************************************************
//** Serial Array Unit UART1 receive interrupt service routine
//******************************************************************************
//#pragma bank = 0 (optional)
#pragma vector = INTSR1_vect
__interrupt void IsrSR1(void)
{
    RxDData=SDR03;
    RxDDataBuffer[BufferPointer++]=RxDData;

    if((RxDData==0x0D)||(BufferPointer>= sizeof(RxDDataBuffer)))
    {
        BufferPointer=0;
        RecDone=1;
    }
}

//******************************************************************************
//** Serial Array Unit UART1 receive error interrupt service routine
//******************************************************************************
//#pragma bank = 0 (optional)
#pragma vector = INTSRE1_vect
__interrupt void IsrSRE1(void)
{
    RxDData=SDR03;            // Receive a byte
}
```

(D.4) Main

The internal high-speed oscillation clock of 8MHz is supplied to the CPU and the peripherals. At first the port lines used by the UART1 and the external interrupt INTP1 will be initialized. After this the UART1 and the external interrupt INTP1 it selves will be initialized.

Additionally several port lines will be used to show the chronology of that what happens during STOP mode release by frame reception of the Serial Array Unit in UART mode, in case that the Serial Array Unit operation will not be stopped before entering the STOP mode.

Port12.1 will be used to indicate that CPU operation starts.

The CPU clock will be output at port P14.0 to indicate, when the on-chip oscillator operation starts. Each received string will be retransmitted in order to show, that no data byte is missed.

```
//******************************************************************************
//** Main loop
//******************************************************************************
void main(void)
{
    // Disable all interrupts
    _DI();

    // Use on-chip high-speed oscillator fCPU = fIH/2^0 = 8MHz
    CKC = 0x08;

    // Initialize Port 3
    P3Init();

    // Initialize Serial Array Unit
    UART1Init();

    // Initialize External Interrupt INTP1 for falling edge detection
    INTP1Init();

    // Reset reception completion flag
    RecDone = 0;

    // Initialize port to indicate stop mode release
    PM12_bit.no0 = 0;
    P12_bit.no0  = 0;

    // Output internal high-speed clock
    PM14_bit.no0= 0;
    P14_bit.no0 = 0;
    CKS0 = 0x80;

    // Endless loop
    while(1)
    {
        // Disable all interrupts
        _DI();

        // Unmask INTP5 interrupt
        PIF5  = 0;
        PMK5  = 0;

        // Reset port bit to indicate that STOP mode will be entered
        P12_bit.no0 = 0;

        // Switch to stop mode
        __stop();

        // Set port bit to indicate that STOP mode is released
        P12_bit.no0 = 1;

        // Mask INTP5 interrupt
        PMK5  = 1;

        // Enable all interrupts
```

```
    _EI();

    // Wait until complete string received
    while(!RecDone)
    {
        _NOP();
    }

    // Transmit received string
    STIF1   = 1;    // Set SAU0 ch-2 UART transmit interrupt request flag
                    // (Transmit path of UART1) Used in DataTransmit()
    i = 0;
    do
    {
        TxDData=RxDDataBuffer[i];
        DataTransmit();
        i++;
    }while(TxDData!=0x0D);

        ;
    STIF1 = 0;
    // End of transmission of received string

    // Reset reception completion flag
    RecDone= 0;
}
```

(E) Valid Specification

Item	Date published	Document No.	Document Title
1	October 2008 or later	U19291E	Preliminary User's Manual 78K0R/Kx3-L
2			

(F) Revision History

Item	Date published	Document No.	Comment
1	April 2009	U19742EE1V0AN00	1st Release