5 UART Programming

내용

- UART 소개
- UART 외부 환경
- UART 구성
- UART 프로그래밍 모델
- UART1 주요 레지스터
- 실습 프로그램 uart.h, uart.c

UART 소개

- UART(Universal Asynchronous Receiver/Transmitter)는 serial line을 통한 비동기 데이터 송수신 방식임
- AM335x는 6개의 UART(UART0-UART5)를 제공
- 각 UART는 64-byte의 receiver FIFO, 64-byte의 transmitter FIFO를 가지고 있음
- FIFO level 기반 interrupt/DMA trigger 기능이 있음
- Divisor 기반 baud rate(300 bps-3,686,400 Mbps) 설정 기능이 있음
- Data format: data 5/6/7/8 bits, parity even/odd/none, stop 1/1.5/2 bit(s)
- Flow control: Hardware (RTS/CTS), software (XON/XOFF)

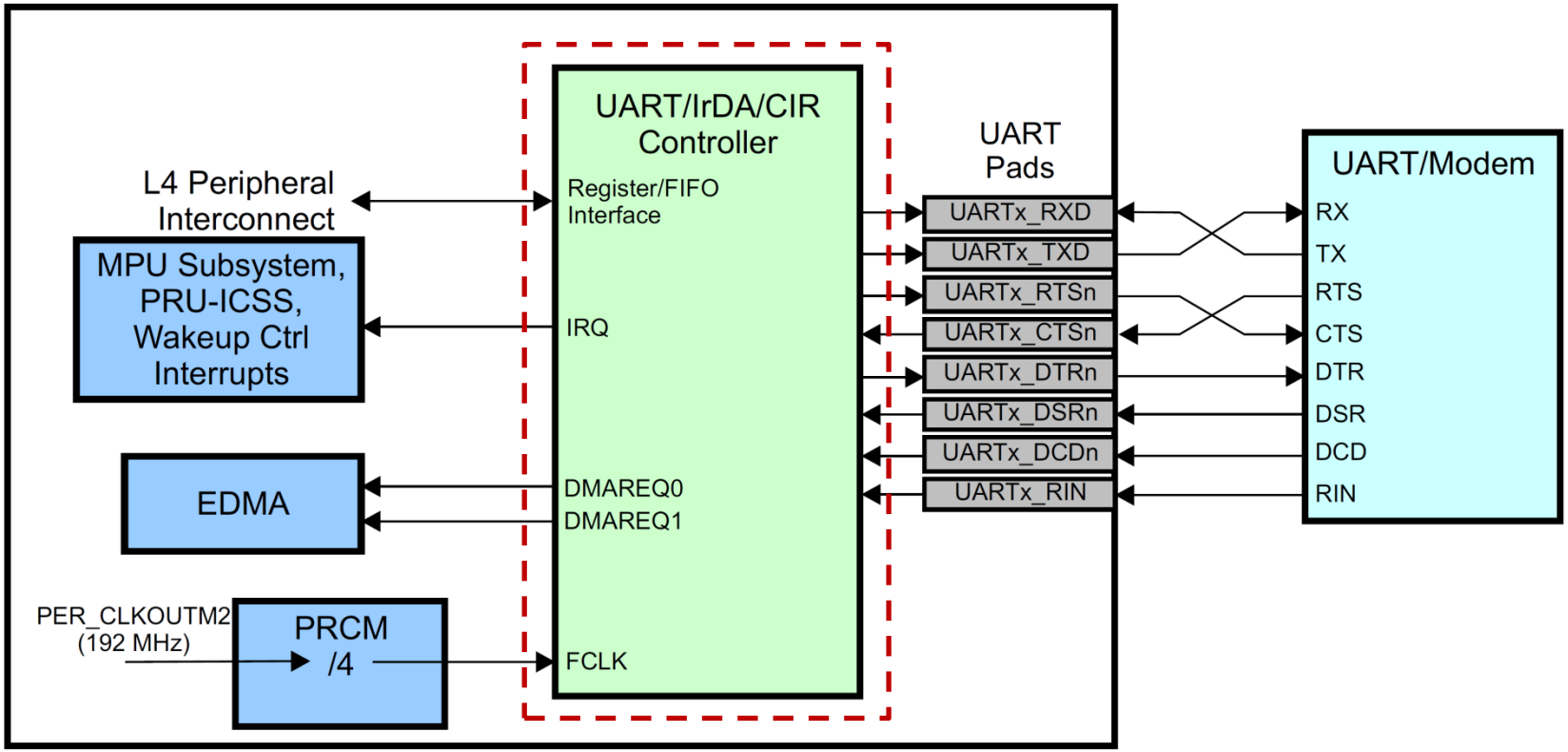
UART 외부 환경

① Clock
- Interface clock (100 Mhz)
- Functional clock (48 Mhz)

② Interrupt
- 1 interrupt to MPU Subsystem

③ DMA
- 2 DMA requests(TX/RX) to EDMA

UART 외부 환경 - 그림


UART/IrDA/CIR
Controller
Register/FIFO
Interface
IRQ
DMAREQ0
DMAREQ1
FCLK
L4 Peripheral
Interconnect
MPU Subsystem,
PRU-ICSS,
Wakeup Ctrl
Interrupts
EDMA
PER_CLKOUTM2
(192 MHz)
PRCM
/4
UART
Pads
UARTx_RXD
UARTx_TXD
UARTx_RTSn
UARTx_CTSn
UARTx_DTRn
UARTx_DSRn
UARTx_DCDn
UARTx_RIN
UART/Modem
RX
TX
RTS
CTS
DTR
DSR
DCD
RIN

UART 구성

① FIFO management: 2개의 FIFO를 사용하여 data의 receive 및 transmit을 관리하는 부분

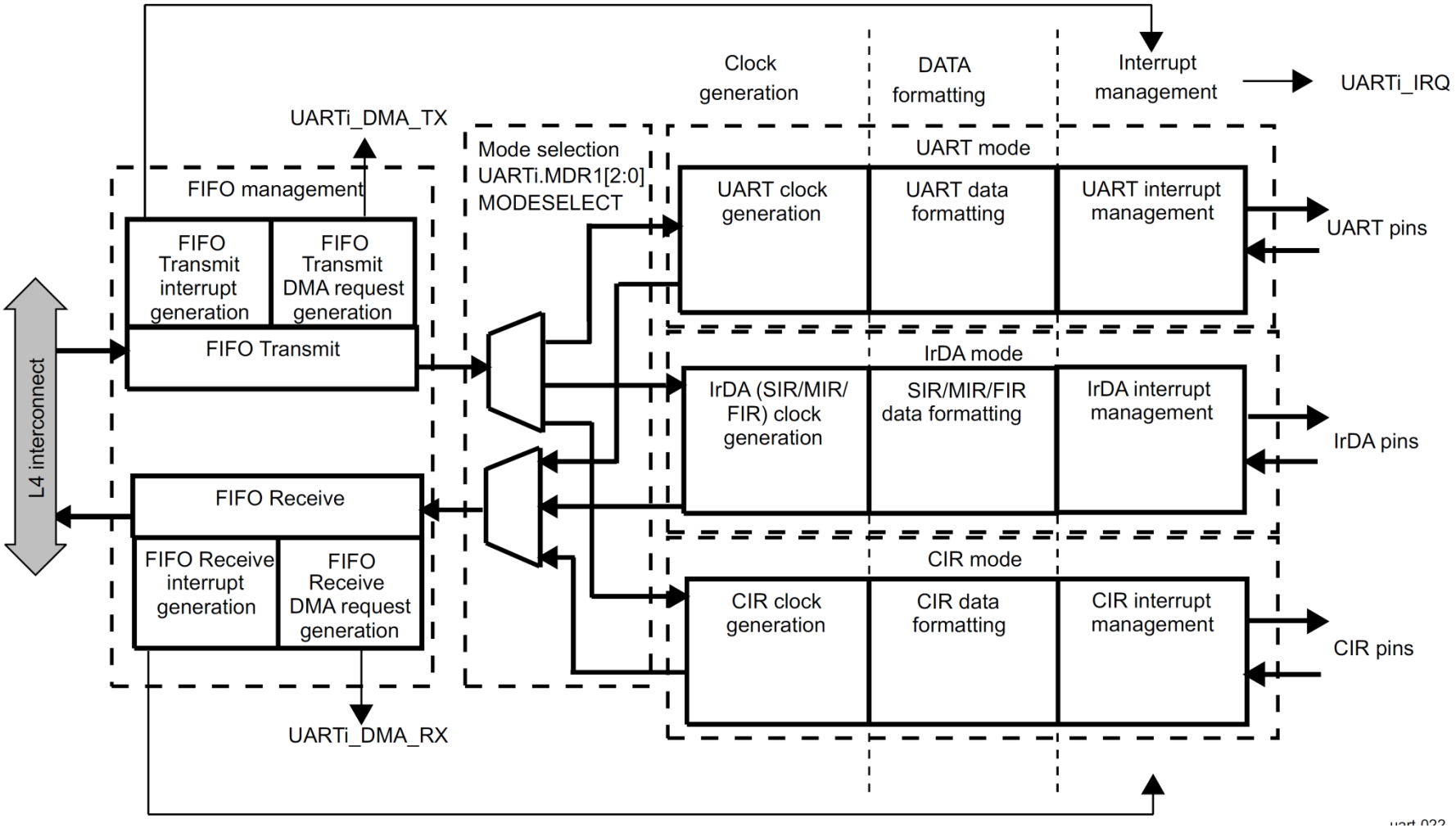
② Mode selection: UART/IrDA/CIR의 mode 선택을 담당하는 부분

③ Clock generation: Baud rate 등을 담당하는 부분

④ Data formatting: Data format을 담당하는 부분

⑤ Interrupt management: Interrupt 관리를 담당하는 부분

UART 구성 - 그림


Clock generation
DATA formatting
Interrupt management
UARTi_IRQ
UARTi_DMA_TX
Mode selection
UARTi.MDR1[2:0]
MODESELECT
UART mode
FIFO management
UART clock generation
UART data formatting
UART interrupt management
UART pins
FIFO Transmit interrupt generation
FIFO Transmit DMA request generation
FIFO Transmit
IrDA mode
L4 interconnect
IrDA (SIR/MIR/FIR) clock generation
SIR/MIR/FIR data formatting
IrDA interrupt management
IrDA pins
FIFO Receive
CIR mode
FIFO Receive interrupt generation
FIFO Receive DMA request generation
CIR clock generation
CIR data formatting
CIR interrupt management
CIR pins
UARTi_DMA_RX

UART 프로그래밍 모델

① Polling 프로그래밍
- 프로그램이 FIFO에 입력된 문자가 있는지 혹은 FIFO에 빈 자리가 있는지를 확인함
- 위 조건을 만족할 때 까지 기다리다가 조건이 만족되면 FIFO에 data를 read/write하여 입출력을 수행함

② Interrupt 프로그래밍
- Data 입출력이 FIFO의 정해진 threshold level을 지나칠 때 UART 하드웨어가 interrupt를 발생시킴
- 실제 data의 입출력은 interrupt handler 프로그램이 처리함

③ DMA 프로그래밍
- Data 입출력이 FIFO의 정해진 threshold level을 지나칠 때 UART 하드웨어가 DMA를 요청함
- 실제 data의 입출력은 DMA 프로세서가 수행함

참고: 본 슬라이드에서는 polling 프로그래밍을 다룸

Polling 프로그래밍

① UART 초기화
- UARTi의 register를 초기화한다
- UARTi의 baud rate 및 data format을 설정한다

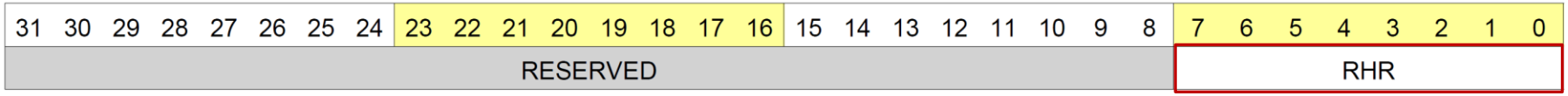
② UART receive
- UARTi_LSR_REG를 읽어서 FIFO가 empty가 아닐 때까지 기다린다
- UARTi_RHR_REG에서 하나의 문자(=1-byte)를 read한다

③ UART transmit
- UARTi_LSR_REG를 읽어서 FIFO가 full이 아닐 때 까지 기다린다
- UARTi_THR_REG에 하나의 문자(=1-byte)를 write한다

UART1 주요 레지스터

레지스터 이름	주소	레지스터 설명
UART1_RHR_REG	0x44e00900	UART1 Receive holding register
UART1_THR_REG	0x44e00900	UART1 Transmit holding register
UART1_LSR_REG	0x44e00914	UART1 Line status register

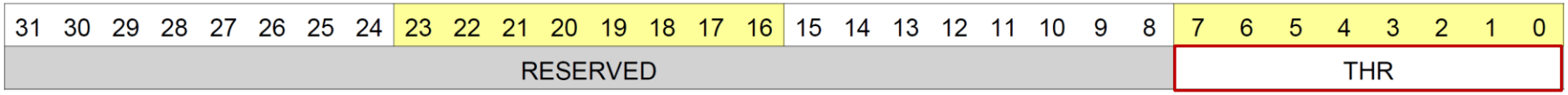
RHR_REG: Receive Holding Register



Bits	Field Name	Description	Type	Reset
31:8	Reserved	Read returns 0.	R	0x000000
7:0	RHR	Receive holding register	R	0x-

- RX input에서 data가 serial로 들어오면 이를 shift하여 64-byte FIFO에 추가됨
- FIFO에 저장된 byte들은 순서대로 RHR_REG로 이동됨

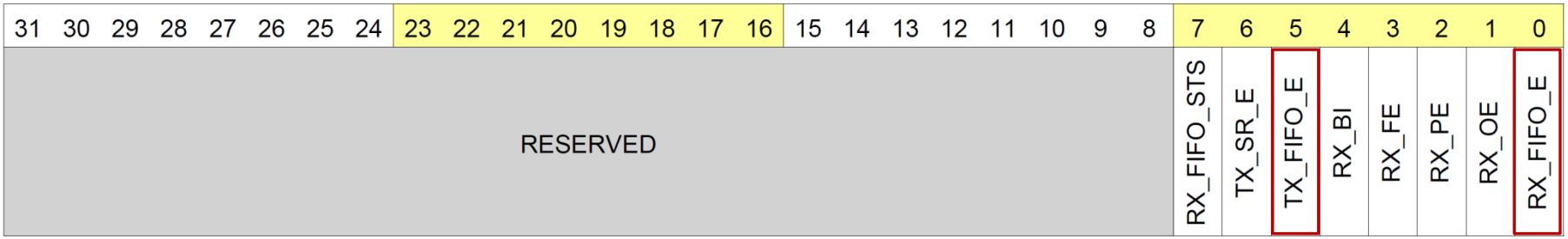
THR_REG: Transmit Holding Register



Bits	Field Name	Description	Type	Reset
31:8	Reserved	Write has no functional effect.	W	0x000000
7:0	THR	Transmit holding register	W	0x-

- THR_REG에 write된 1 byte는 64-byte 크기의 transmit FIFO로 이동됨
- FIFO에 저장된 byte들은 순서대로 shift되어 serial로 변환된 후 TX output으로 출력됨

LSR_REG: Line Status Register



- RX_FIFO_E (초기값은 0x0)
  - 0x0: receive FIFO가 empty라는 의미
  - 0x1: receive FIFO가 empty가 아니라는 의미
- TX_FIFO_E (초기값은 0x1)
  - 0x0: transmit FIFO가 full이라는 의미
  - 0x1: transmit FIFO가 full이 아니라는 의미

보기 프로그램 uart.h

```
int UART_getc(void);
int UART_getc_noblock(void);
unsigned int UART_gethex(void);
char *UART_gets(char *str);

int UART_putc(int c);
int UART_puts(const char *s);
void UART_clear(void);
void UART_goto(int x, int y);
int UART_sprintf(char *out, const char
  *format, ...);
int UART_printf(const char *format, ...);
```

보기 프로그램 uart.c - 1/3

```
#include <uart.h>

#define IO_READ(addr)         (*(volatile unsigned int *)(addr))
#define IO_WRITE(addr, val)   (*(volatile unsigned int *)(addr) = (val))

#define UART1_RHR_REG    0x44e09000 // UART1 Receive holding register
#define UART1_THR_REG    0x44e09000 // UART1 Transmit holding register
#define UART1_LSR_REG    0x44e09014 // UART1 Line status register

#define LSR_RX_FIFO_E    0x01    // LSR receive fifo empty
#define LSR_TX_FIFO_E    0x20    // LSR transmit fifo empty

...
```

실습 프로그램 uart.c - 2/3

```
int UART_getc(void)
{
  unsigned int c;

  // Read UART1_LSR_REG and check LSR_RX_FIFO_E bit.
  while ((IO_READ(UART1_LSR_REG) & LSR_RX_FIFO_E) == 0);

  // Read a character from UART1_RHR_REG.
  c = IO_READ(UART1_RHR_REG);

  // Echo back the character.
  UART_putc(c);

  return (int) c;
}


...
```

보기 프로그램 uart.c - 3/3

```
int UART_putc(int c)
{
  // if c is a '\r' or a '\n', put a '\r' and a '\n'.
  if ((c == '\r') || (c == '\n')) {
    c = '\r';

    // Read UART1_LSR_REG and check LSR_TX_FIFO_E bit.
    while ((IO_READ(UART1_LSR_REG) & LSR_TX_FIFO_E) == 0);

    // Write a character to UART1_THR_REG.
    IO_WRITE(UART1_THR_REG, c);

    c = '\n';
  }
  // Read UART1_LSR_REG and check LSR_TX_FIFO_E bit.
  while ((IO_READ(UART1_LSR_REG) & LSR_TX_FIFO_E) == 0);

  // Write a character to UART1_THR_REG.
  IO_WRITE(UART1_THR_REG, c);

  return c;
}

...
```

참고 문헌

1. Texas Instruments, Sitara™ AM335x ARM® Cortex™-A8 Microprocessors (MPUs), SPRS717F, April 2013.
2. Texas Instruments, AM335x ARM® Cortex™-A8 Microprocessors (MPUs) Technical Reference Manual, SPRUH73J, December 2013. (Chapter 19)