Recap

ADC

Successive approximation, precision, resolution

Busy-wait synchronization

Nyquist Theorem

Must sample ADC at a regular rate, f_s

If signal frequencies exist f_{min} to f_{max}, make $f_s>2f_{max}$

How OC interrupts create a sampling rate

ISR execution speed short compared to period

Use of mailbox to pass data

Resolution is smallest change that can be distiguished

Accuracy is difference between truth and measured

Overview

Device driver

Serial communication; what does the frame look like

SCI shift register versus SCI data register

How RDRF is set; how RDRF is cleared

How TDRE is set; how TDRE is cleared

A **device driver** is a collection of software functions that allow higher level software to utilize an I/O device.

Collection of public methods (subroutines)
`SCI1_Init`
`SCI1_InChar`
`SCI1_OutChar`
Collection of private objects (subroutines, globals, I/O ports)
`SCI1CR2`
`SCI1BD`
`SCI1SR1`
`SCI1DRL`

complexity abstraction

divide a complex problem into simple subcomponents

functional abstraction

divide a problem into modules

grouped by function

8.2. Serial Communications Interface, SCI

baud rate total number of bits transmitted per second

M, selects 8-bit (M=0) or 9-bit (M=1) data frames.

A *frame* is the smallest complete unit of serial transmission.

bandwidth useful information transmitted per second.

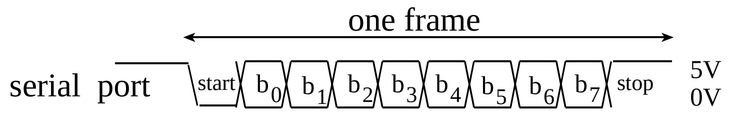
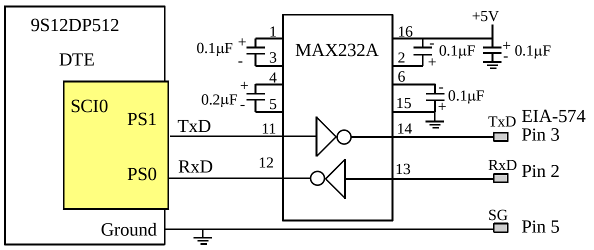
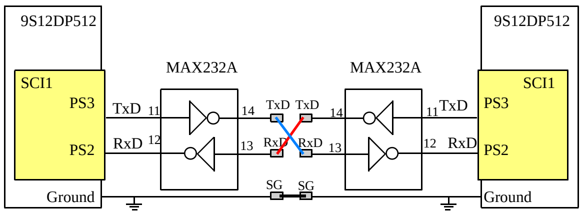


Figure 8.1. A serial data frame with M=0.





Transmitting in asynchronous mode (Lab 8 uses SCI1)

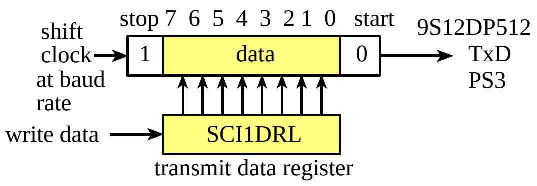


Figure 8.3. Data and shift registers implement the serial transmission.

The software writes to SCI1DRL, then
- 8 bits of data are moved to the shift register
- start and stop bits are added
- shifts in 10 bits of data one at a time on TxD line
- shift one bit per bit time (=1/baudRate)

Receiving in asynchronous mode (Lab 8 uses SCI1)

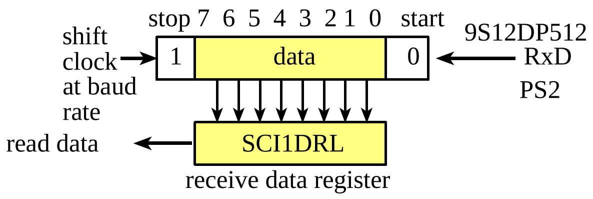


Figure 8.4. Data register shift registers implement the receive serial interface.

The receiver waits for the 1 to 0 edge signifying a start bit, then
- shifts in 10 bits of data one at a time from RxD line
- shift one bit per bit time (=1/baudRate)
- start and stop bits are removed
- checked for noise and framing errors
- 8 bits of data are loaded into the SCI1DRL

There are TWO SCI1DRL at the same address!
- Write cycles go to transmitter to be sent
- Read cycles come from receiver which were received
- *Can't look at SCI1DRL in the debugger, because*
  - a) Which one do you see?
  - b) Debugger takes your data

8.2.4. 9S12 SCI Details

address	msb															lsb	Name
$00D0	-	-	-	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**	SCI1BD

Address	Bit 7	6	5	4	3	2	1	Bit 0	Name
$00D2	LOOPS	SWAI	RSRC	M	WAKE	ILT	PE	PT	SCI1CR1
$00D3	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK	SCI1CR2
$00D4	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF	SCI1SR1
$00D5	0	0	0	0	0	BRK13	TXDIR	RAF	SCI1SR2
$00D6	R8	T8	0	0	0	0	0	0	SCI1DRH
$00D7	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0	SCI1DRL

Similar to Table 8.4. 9S12 SCI ports (SCI1).

SCI1BD

$$\text{SCI Baud Rate} = \frac{\text{MCLK}}{(16\bullet\text{BR})}$$

on 9S12DP512/9S12DG128

MCLK	= 24MHz (with PLL, in LOAD mode)
	= 8 MHz (otherwise)

BR is 13 bits

TE is the Transmitter Enable bit, and
RE is the Receiver Enable bit.

TDRE is the Transmit Data Register Empty flag.
set by the SCI hardware if transmit data register empty
if set, the software write next output to SCI1DRL
cleared by two-step software sequence
first reading SCI1SR1 with TDRE set
then SCI1DRL write

RDRF is the Receive Data Register Full flag.
set by hardware if a received character is ready to be read
if set, the software read next into from SCI1DRL
cleared by two-step software sequence
first reading SCI1SR1 with RDRF set
then SCI1DRL read

RIE is the Receive Interrupt Enable bit (Arm).
set and cleared by software
set to arm RDRF triggered interrupts
clear to disarm RDRF triggered interrupts

TIE is the Transmit Interrupt Enable bit (Arm).
set and cleared by software
set to arm TDRE triggered interrupts
clear to disarm TDRE triggered interrupts

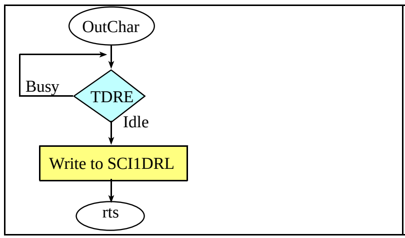
SCI1DRL register contains transmit and receive data
these two registers exist at the same I/O port address
Reads access the read-only receive data register (RDR)
Writes access the write-only transmit data register (TDR)

Busy-waiting, gadfly, or *polling* are three equivalent names
software continuously checks the hardware status waiting for it to be ready.

SCI I/O Programming.

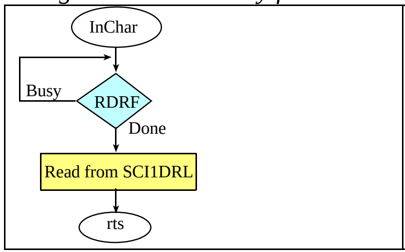
```
//SCI1BD is 13 bits, BR
```

```
//baud rate (bps) = 8000000/16/BR (38461.5)
// initialize SCI, assuming 8MHz E clock
void SCI1_Init(void){
  SCI1BD  = 13;   // 38400 bits/sec
  SCI1CR2 = 0x0C; // enable
}
```



```
#define TDRE 0x80
// Wait for buffer to be empty,
// then output
void SCI1_OutChar(char data){
  while((SCI1SR1&TDRE) == 0){};
  SCI1DRL = data;
}
```

Program 8.1. Assembly functions that implement serial I/O.



```
#define RDRF 0x20
// Wait for new input,
// then return ASCII code
char SCI1_InChar(void){
  while((SCI1SR1&RDRF) == 0){};
  return(SCI1DRL);
}
```

Profile is a debugging tool to measure

Where software is executing?
When does it execute?

Open Tut3 in TExaS,
show SCI functions
profile SCI_OutChar, what percentage of time is spent waiting for the SCI?
how do we recover this lost time waiting?

```
PTT      equ  $0240
DDRT     equ  $0242
         bset DDRT,#$01
         bclr PTT,#$01
         bset PTT,#$01
         bclr PTT,#$01
```

The bottom line

Serial transmission is one bit at a time
Baud rate is the total number of bits/sec
Bandwidth is the information transfer rate
Frame is 10 bits: Start,0,1,2,3,4,5,6,7,Stop
There are two shift registers and two data registers
Know RDRF and TDRE
What sets these bits?
How are the bits cleared.