Autopilot

Topics

•I/O-Ports	- buttons and led
•Serial communication RS232	- communication with PC
•Address and Data Bus	-key board and display
•PWM Pulse Width Modulation	-servo
•I2C Inter-Integrated Circuit	-compass

Autopilot

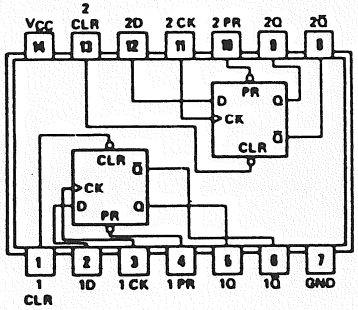
DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS WITH PRESET AND CLEAR

74

FUNCTION TABLE

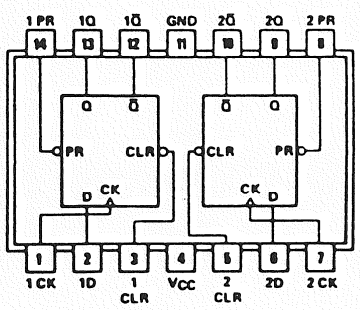
INPUTS				OUTPUTS	
PRESET	CLEAR	CLOCK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H*	H*
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

See pages 6-46, 6-50, 6-54 and 6-56



SN5474 (J)
SN54ALS74 (J)
SN54AS74 (J)
SN54HC74 (J)
SN54LS74A (J, W)
SN54S74 (J, W)

SN7474 (J, N)
SN74ALS74 (N)
SN74AS74 (N)
SN74HC74 (N)
SN74LS74A (J, N)
SN74S74 (J, N)



SN5474 (W)

Autopilot

Memory map over some registers

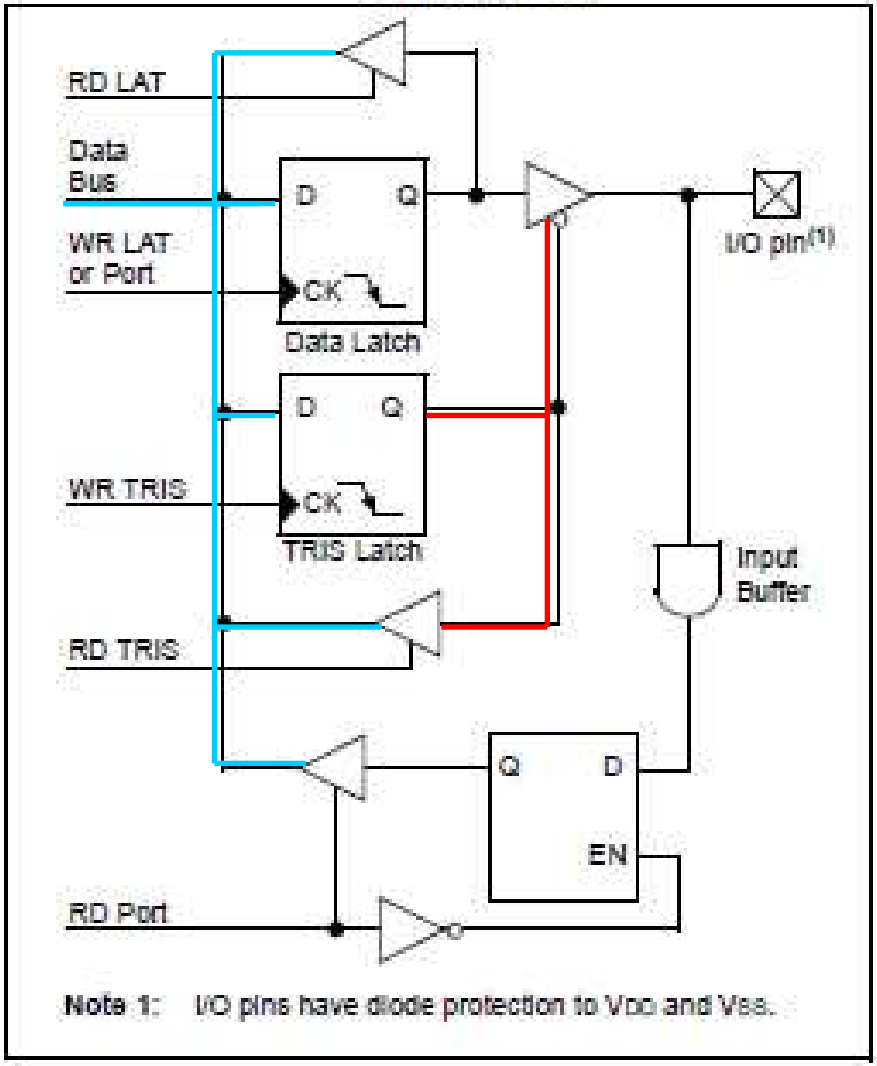
PIC18F2525

F95h	TRISD(3)
F94h	TRISC
F93h	TRISB
F92h	TRISA
F91h	—(2)
F90h	—(2)
F8Fh	—(2)
F8Eh	—(2)
F8Dh	LATE(3)
F8Ch	LATD(3)
F8Bh	LATC
F8Ah	LATB
F89h	LATA
F88h	—(2)
F87h	—(2)
F86h	—(2)
F85h	—(2)
F84h	PORTE(3)
F83h	PORTD(3)
F82h	PORTC
F81h	PORTB
F80h	PORTA

Autopilot

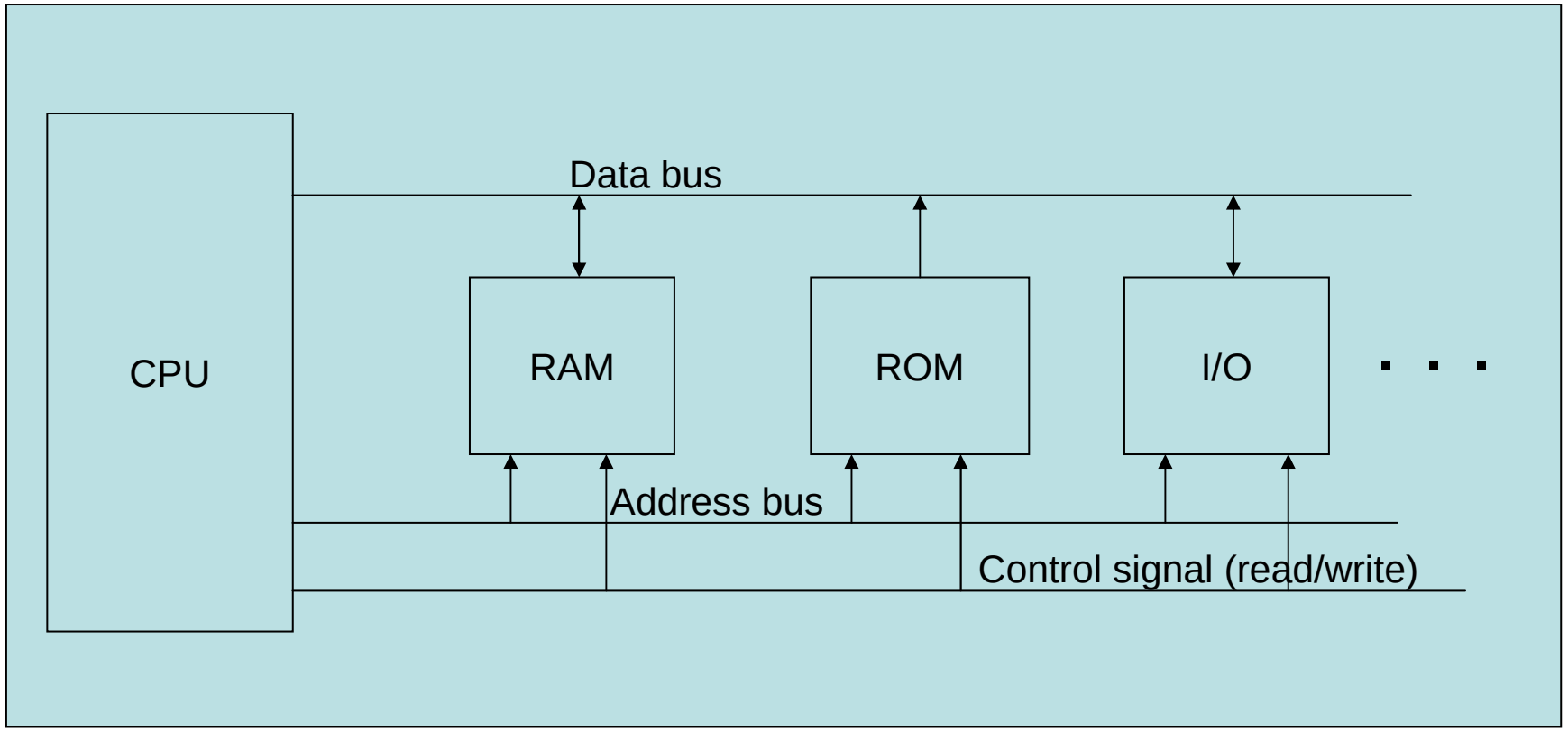
```
struct Ports{
   unsigned char A;
   unsigned char B;
} *rpPorts ;

void main(void){

    unsigned char I = 0;
    unsigned char *pTrisB = 0xf93;
    unsigned char *pPortA  = 0xf80;

   rpPorts  =  0xf80;
   *pTrisB = 0x00;
   *0xf93 = 0xf0;

    while(1){
        i++;
        *pPortA = i;
        rpPorts->B = i;
    }
}
```

FIGURE 9-1: GENERIC I/O PORT OPERATION

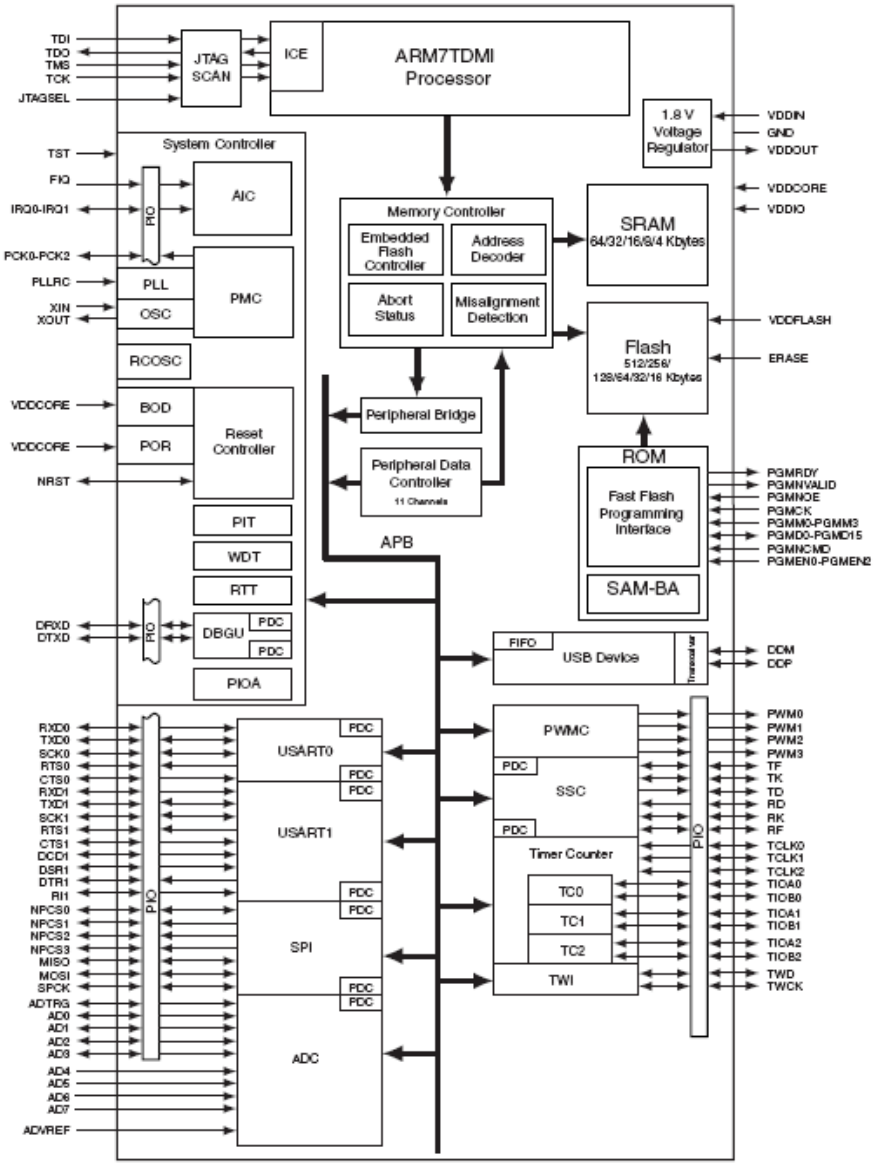


Autopilot

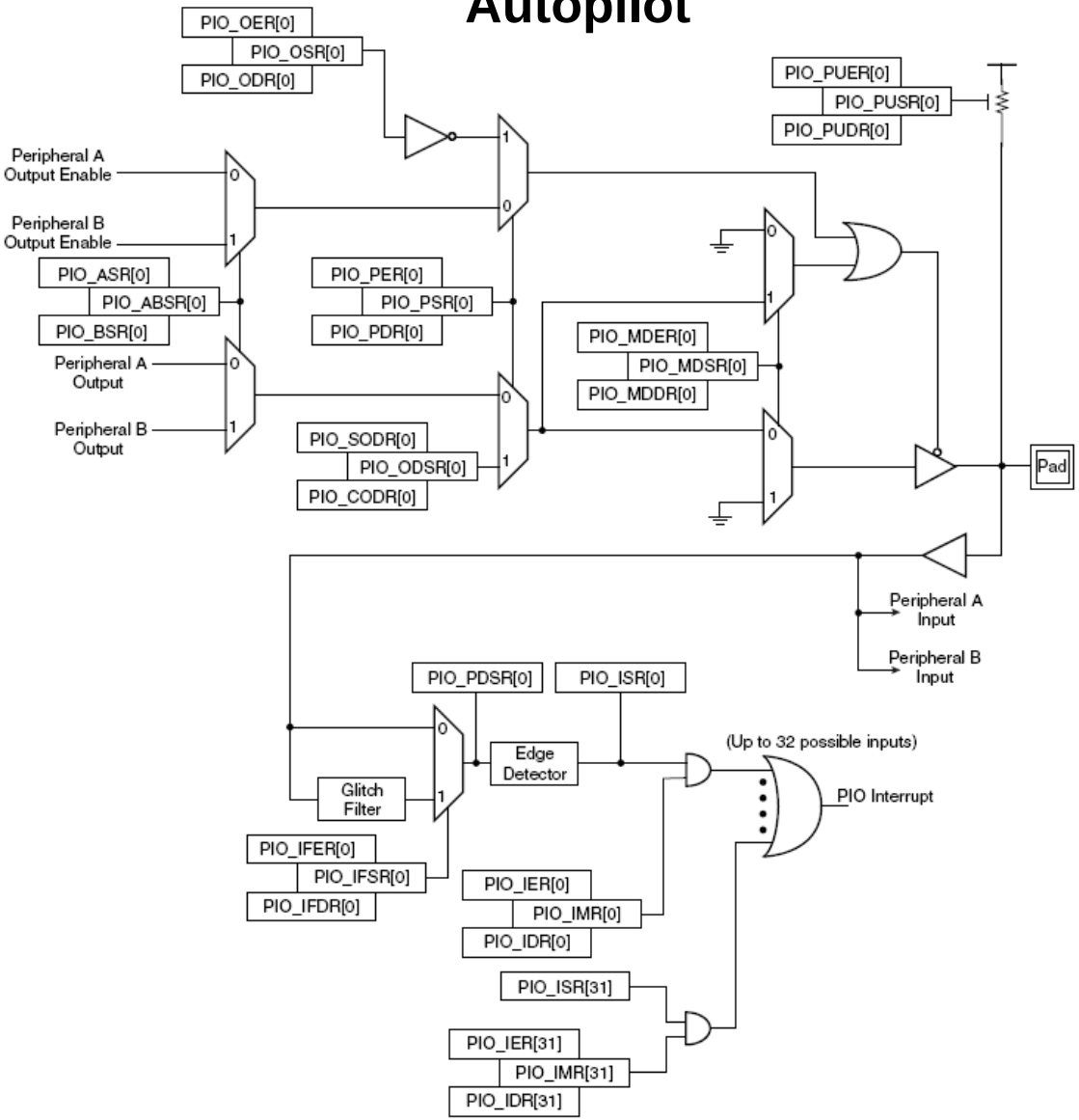
Micro computer/controller



Autopilot


TDI
TDO
TMS
TCK
JTAGSEL
JTAG SCAN
ICE
ARM7TDMI Processor
1.8 V Voltage Regulator
VDDIN
GND
VDDOUT
VDDCORE
VDDIO
System Controller
TST
FIQ
IRQ0-IRQ1
PIO
AIC
PCK0-PCK2
PLLRC
XIN
XOUT
PLL
OSC
PMC
RCOSC
VDDCORE
VDDCORE
NRST
BOD
POR
Reset Controller
PIT
WDT
RTT
DRXD
DTXD
PIO
DBGU
PDC
PDC
PIOA
Memory Controller
Embedded Flash Controller
Address Decoder
Abort Status
Misalignment Detection
SRAM 64/32/16/8/4 Kbytes
Flash 512/256/ 128/64/32/16 Kbytes
VDDFLASH
ERASE
Peripheral Bridge
Peripheral Data Controller
11 Channels
APB
ROM
Fast Flash Programming Interface
SAM-BA
PGMRDY
PGMNVALID
PGMNOE
PGMCK
PGMM0-PGMM3
PGMD0-PGMD15
PGMNCMD
PGMEN0-PGMEN2
FIFO
USB Device
Transceiver
DDM
DDP
RXD0
TXD0
SCK0
RTS0
CTS0
RXD1
TXD1
SCK1
RTS1
CTS1
DCD1
DSR1
DTR1
RI1
NPCS0
NPCS1
NPCS2
NPCS3
MISO
MOSI
SPCK
ADTRG
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
ADVREF
PIO
USART0
USART1
SPI
ADC
PWMC
SSC
Timer Counter
TC0
TC1
TC2
TWI
PWM0
PWM1
PWM2
PWM3
TF
TK
TD
RD
RK
RF
TCLK0
TCLK1
TCLK2
TIOA0
TIOB0
TIOA1
TIOB1
TIOA2
TIOB2
TWD
TWCK

Autopilot


PIO_OER[0]
PIO_OSR[0]
PIO_ODR[0]
PIO_PUER[0]
PIO_PUSR[0]
PIO_PUDR[0]
Peripheral A Output Enable
Peripheral B Output Enable
PIO_ASR[0]
PIO_ABSR[0]
PIO_BSR[0]
PIO_PER[0]
PIO_PSR[0]
PIO_PDR[0]
PIO_MDER[0]
PIO_MDSR[0]
PIO_MDDR[0]
Peripheral A Output
Peripheral B Output
PIO_SODR[0]
PIO_ODSR[0]
PIO_CODR[0]
Pad
Peripheral A Input
Peripheral B Input
PIO_PDSR[0]
PIO_ISR[0]
(Up to 32 possible inputs)
Edge Detector
Glitch Filter
PIO Interrupt
PIO_IFER[0]
PIO_IFSR[0]
PIO_IFDR[0]
PIO_IER[0]
PIO_IMR[0]
PIO_IDR[0]
PIO_ISR[31]
PIO_IER[31]
PIO_IMR[31]
PIO_IDR[31]

Autopilot

```
typedef struct _AT91S_PIO{
        AT91_REG        PIO_PER;
        AT91_REG        PIO_PDR;
        AT91_REG        PIO_PSR;
        AT91_REG        Reserved_0
        AT91_REG        PIO_OER;
        .
        .
} *AT91PS_PIO;
#define AT91C_BASE_PIOA  (AT91PS_PIO) 0xfffff400
```

```
void Config_Ports(void){
        // pointer to PIO data structure
        volatile AT91PS_PIO   pPIO = AT91C_BASE_PIOA;

        // PIO Enable Register - allow PIO to control pins
        pPIO->PIO_PER = ???????????; // port 17, 18, 19, and 20

        // PIO Output Enable Register
        pPIO->PIO_OER = ???????????; // port 17 and 18
}

void Yellow_Led_On(void){
         volatile AT91PS_PIO   pPIO = AT91C_BASE_PIOA
         pPIO->???? = ?????  ;
}
```

Autopilot

27.6 Parallel Input/Output Controller (PIO) User Interface

Each I/O line controlled by the PIO Controller is associated with a bit in each of the PIO Controller User Interface registers. Each register is 32 bits wide. If a parallel I/O line is not defined, writing to the corresponding bits has no effect. Undefined bits read zero. If the I/O line is not multiplexed with any peripheral, the I/O line is controlled by the PIO Controller and PIO_PSR returns 1 systematically.

Table 27-2. Register Mapping

Offset	Register	Name	Access	Reset
0x0000	PIO Enable Register	PIO_PER	Write-only	–
0x0004	PIO Disable Register	PIO_PDR	Write-only	–
0x0008	PIO Status Register	PIO_PSR	Read-only	(1)
0x000C	Reserved			
0x0010	Output Enable Register	PIO_OER	Write-only	–
0x0014	Output Disable Register	PIO_ODR	Write-only	–
0x0018	Output Status Register	PIO_OSR	Read-only	0x0000 0000

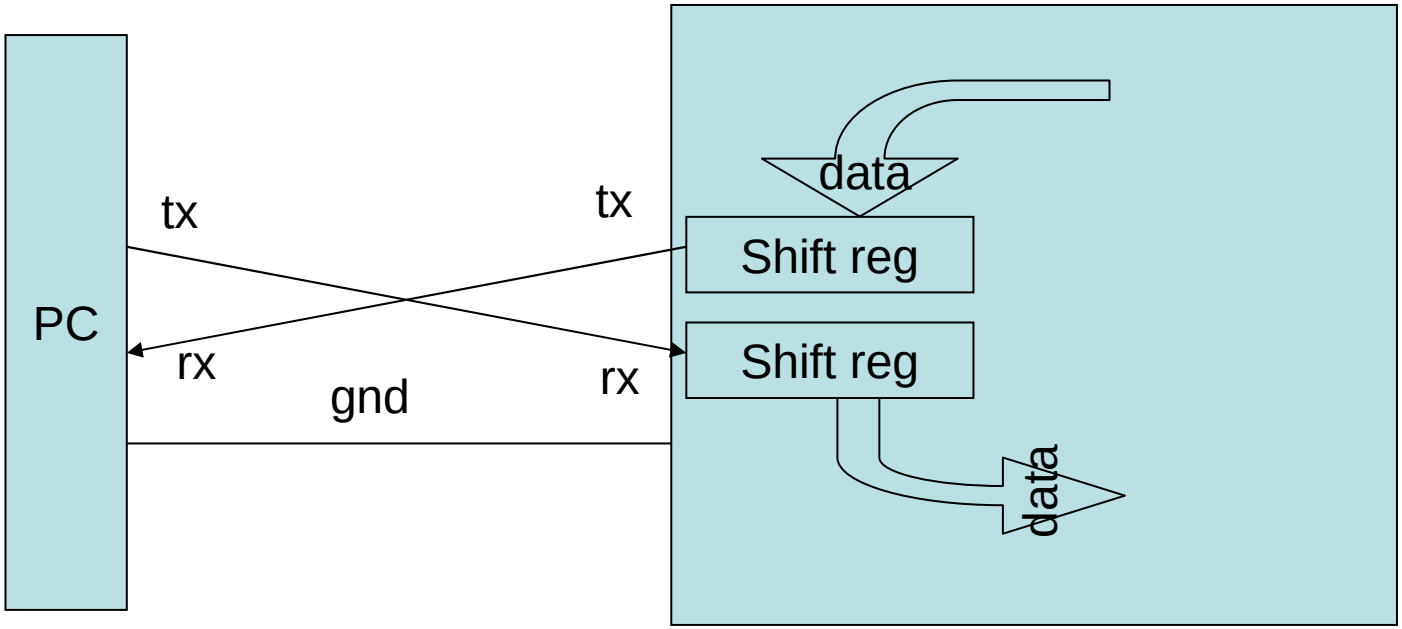
Autopilot

Any questions?

More info about digital theory:
http://www.opamp-electronics.com/tutorials/digital_theory.htm

Autopilot

•Serial communication RS232 - PC



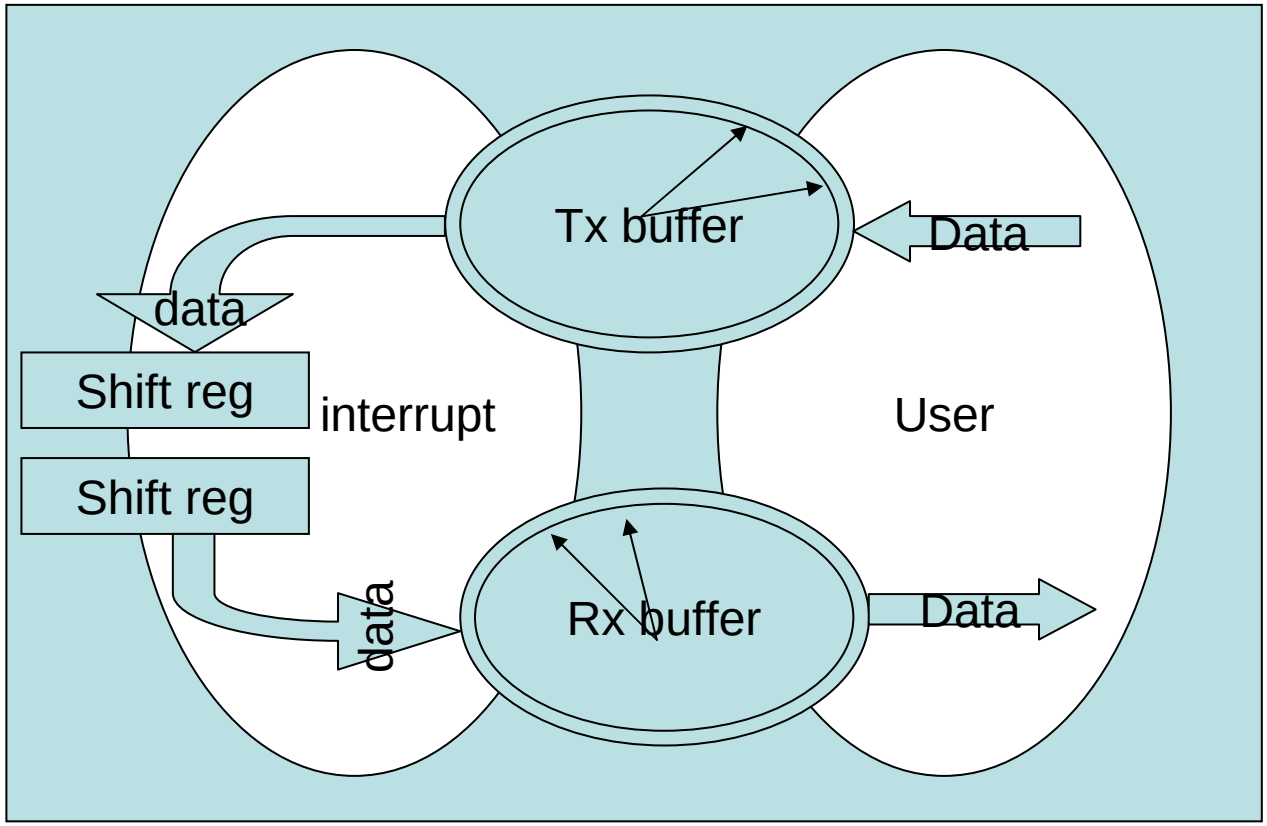
Autopilot

- Not so good programming

```
//write char to UART0
void write_char_USART0(unsigned char ch)
{
  while (!(u_pUSART0->US_CSR & AT91C_US_TXRDY) == 1);
  u_pUSART0->US_THR  = ((ch & 0x1FF));
}
```

Autopilot

•Serial communication by using interrupt and ring bufferts



Autopilot

```
char TxFifoQueue[ChBufferSize]; //              Global Variables
char RxFifoQueue[ChBufferSize];
int TxIndexIn = 0, RxIndexIn = 0, TxIndexOut = 0 ,RxIndexOut = 0;

void Usart0IrqHandler (void) {

   volatile AT91PS_USART pUsart0 = AT91C_BASE_US0;// create a pointer to USART0 structure

// determine which interrupt has occurred
   if ((pUsart0->US_CSR & AT91C_US_RXRDY) == AT91C_US_RXRDY) {
// we have a receive interrupt,
        RxFifoQueue[RxIndexIn] = pUsart0->US_RHR;
        RxIndexIn = (RxIndexIn+1)%ChBufferSize;
   } else if ((pUsart0->US_CSR & AT91C_US_TXEMPTY) == AT91C_US_TXEMPTY) {
// we have a transmit interrupt
        if(TxIndexOut != TxIndexIn){ // check if more characters in queue
              pUsart0->US_THR = TxFifoQueue[TxIndexOut];
             TxIndexOut = (TxIndexOut+1)%ChBufferSize;
         }
         else {
              pUsart0->US_IDR = ~AT91C_US_RXRDY;// disable all interrupts except RXRDY
         }
     }//else if
}
```

Autopilot

```
void PutChUsart0(char Ch){

volatile AT91PS_USART pUsart0 = AT91C_BASE_US0;

Copy Ch to buffer
Move the ”pointer”
pUsart0->US_IER = AT91C_US_TXEMPTY; // enable tx interrupts
}

char GetChUsart0(void){

Copy from buffer to temp
Move the ”pointer”
return temp;
}

int Char_In_Usart0(void){
 return something
}
```

Autopilot

Any questions?

Autopilot

Wire-Wrap®

The Wire-Wrap wrapping technique

Wire-Wrap wrapping process

Reliable standard connection (not modified): The standard wrapping bit wraps only the stripped part of the wire on the terminal.

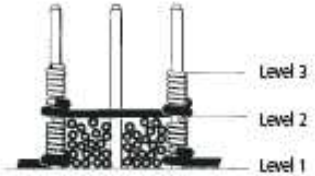
Reliable modified connection: The insulation of a connection made on the second level may overlap the last turn of the connection wrapped on level one. The modified wrapping bit will wrap one turn of insulation at the base of the connection.

The **modified C.S.W. bits** cut the wire, strip the insulation and wrap the wire in one operation.

Numbers of turns of conductor	Related to conductor diameter
8	0,25 mm (30 AWG*)
7	0,32 mm (28 AWG)
6	0,4 mm (32 AWG)
5	0,5 mm (24 AWG)
4	0,65 mm (22 AWG)
4	0,8 mm (20 AWG)
4	1,2 mm (18 AWG)

* AWG = American Wire Gauge is nationally accepted for conductor diameter definition

The golden rules of wire wrapping



1. Only two connections on the same terminal (Level 3 is kept as a reserve for change and repair).
2. Wrap both ends of a wire on the same level.
3. Wrap long wires first.
4. Removal of connections: simply unwrap and do not reuse the unwrapped wire.

Correct dressing of the wire around a terminal.

Incorrect dressing of the wire. No clearance between wire and terminal.

Autopilot

Wrong wire wrapping

Overwrap:
Too much backforce or unproperly selected bit.

Spiraled connection:
The tool has been pulled backwards during the wrapping process.

Insufficient insulation for a modified connection:
The wire has not been inserted deep enough in the wire slot, or has slipped out just before wrapping.

Too much „Pig Tail":
The last turn of connection is not formed against the terminal damaged wrapping bit ratio between terminal width and thickness too large.

Autopilot

Any questions?