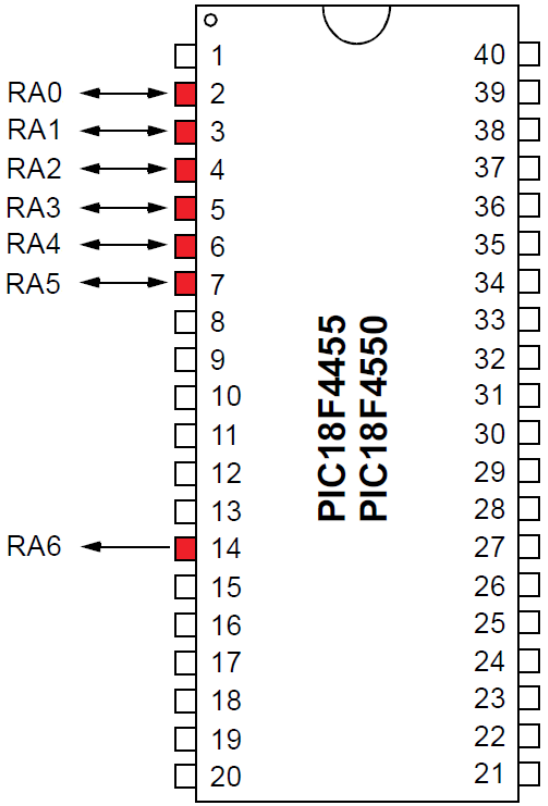
INSTITUTO FEDERAL
Rio Grande do Sul
Campus Farroupilha

PIC18F4550: Entadas e Saídas Digitais

Prof. Matheus Ribeiro

PORT A

- 7 pinos de entrada e saída
- Pinos RA0 a RA5 configurados como entradas analógicas na inicialização (POR)
- Pino RA6 utilizado como saída de realimentação do oscilador principal nos modos HS e XT (cristal)
- Registradores associados:
  - PORTA
  - LATA
  - TRISA



PORT A

Registrador TRISA

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0

- Responsável por configurar aos pinos como entradas ou saídas
  0 = saída, 1 = entrada

```
TRISA = 0b00001111;    // configura RA0 a RA3 como entradas
                       // configura RA4 a RA6* como saídas

TRISAbits.RA0 = 1;     // configura RA0 (pino 2) como entrada
TRISAbits.RA5 = 0;     // configura RA5 (pino 7) como saída
```

PORT A

Registrador PORTA

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0

- Responsável pela operação do pino como entrada ou saída
- Operação de escrita:

```
PORTA = 0b01010101;    // atribui níveis lógicos para todo porto
                       // RA0, RA2, RA4, RA6*: nível alto
                       // RA1, RA3, RA5: nível baixo

PORTAbits.RA1 = 1;     // RA1 (pino 3) = nível alto
PORTAbits.RA2 = 0;     // RA2 (pino 4) = nível baixo
```

PORT A

Registrador PORTA

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0

- Responsável pela operação do pino como entrada ou saída
- Operação de leitura:

```
X = PORTA;              // X (8 bits) recebe os estados lógicos
                        // de todos os pinos do porto A

Y = PORTAbits.RA4;      // Y recebe o estado lógico de RA4 (pino 6)
```

PORT A

Registrador LATA

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de escrita:

```
LATA = 0b01010101;      // RA0, RA2, RA4, RA6*: nível alto
                        // RA1, RA3, RA5: nível baixo

LATAbits.LATA1 = 1;     // RA1 (pino 3) = nível alto
LATAbits.LATA2 = 0;     // RA2 (pino 4) = nível baixo
```

PORT A

Registrador LATA

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de leitura:

```
X = LATA;                 // X (8 bits) recebe os estados lógicos
                          // de todos os pinos do porto A

Y = LATAbits.LATA4;       // Y recebe o estado lógico de RA4 (pino 6)
```

PORT A

Rotina de inicialização (conforme folha de dados)

```
PORTA = 0b00000000;        // limpa o porto A
LATA  = 0b00000000;        // limpa o latch do porto A (opcional)
ADCON1= 0b00001111;        // configura todos pinos como digitais
CMCON = 0b00000111;        // desliga módulo comparador (opcional)
                           // padrão na inicialização
TRISA = 0b11001111;        // configurar de acordo com a
                           // utilização dos pinos
```

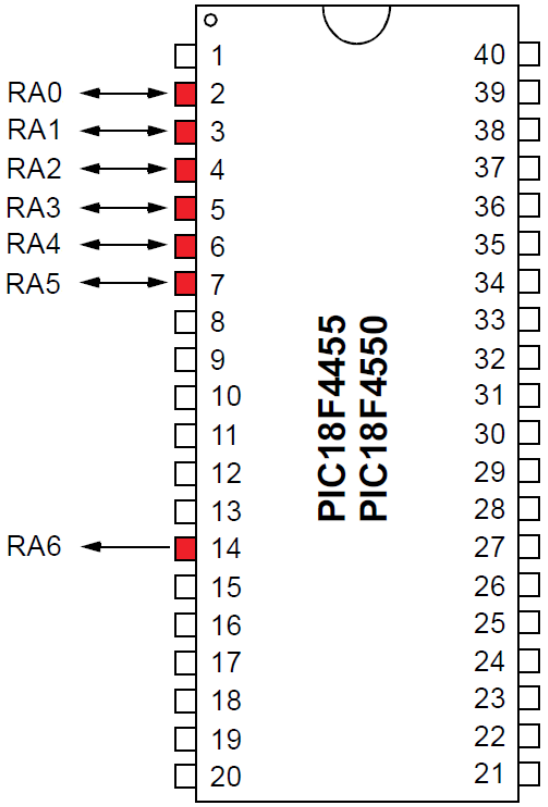
- O valor do dos pinos do PORTA é desconhecido na inicialização, então limpe o porto antes de configurar os pinos.
- Configure os pinos não utilizados como saídas no registrador TRISA.

PORT A

Exemplo de programa

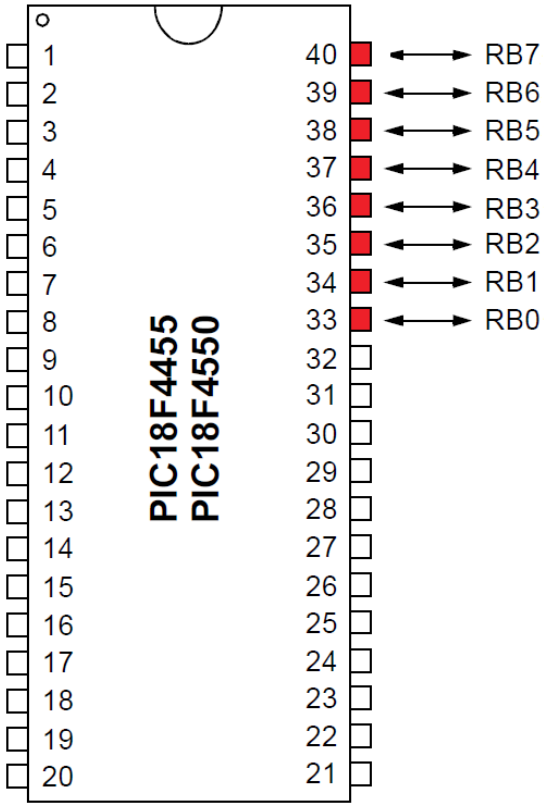
```
void main() {
    // Inicialização do porto A
    PORTA = 0b00000000;
    LATA  = 0b00000000;
    TRISA = 0b00000011;
    ADCON1= 0b00001111;

    while(1){
        if(PORTAbits.RA0 == 1){
            PORTAbits.RA3 = 1;
        }
        if(PORTAbits.RA1 == 1){
            PORTAbits.RA3 = 0;
        }
    }
}
```



PORT B

- 8 pinos de entrada e saída
- Ativação de resistores de pull-up
  (bit RBPU do registrador INTCON2)
- Pinos RB0 a RB4 configurados como entradas analógicas na inicialização (POR)
- ICSP nos pinos RB5, RB6 e RB7
- Registradores associados:
  - PORTB
  - LATB
  - TRISB



PORT B

Registrador TRISB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0

- Responsável por configurar aos pinos como entradas ou saídas
  0 = saída, 1 = entrada

```
TRISB = 0b00001111;    // configura RB0 a RB3 como entradas
                       // configura RB4 a RB7 como saídas

TRISBbits.RB0 = 1;     // configura RB0 (pino 33) como entrada
TRISBbits.RB5 = 0;     // configura RB5 (pino 38) como saída
```

PORT B

Registrador PORTB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0

- Responsável pela operação do pino como entrada ou saída
- Operação de escrita:

```
PORTB = 0b01010101;    // atribui níveis lógicos para todo porto
                       // RB0, RB2, RB4, RB6: nível alto
                       // RB1, RB3, RB5, RB7: nível baixo

PORTBbits.RB1 = 1;     // RB1 (pino 34) = nível alto
PORTBbits.RB2 = 0;     // RB2 (pino 35) = nível baixo
```

PORT B

Registrador PORTB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0

- Responsável pela operação do pino como entrada ou saída
- Operação de leitura:

```
X = PORTB;              // X (8 bits) recebe os estados lógicos
                        // de todos os pinos do porto B

Y = PORTBbits.RB4;      // Y recebe o estado lógico de RB4 (pino 37)
```

PORT B

Registrador LATB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de escrita:

```
LATB = 0b01010101;     // RB0, RB2, RB4, RB6: nível alto
                       // RB1, RB3, RB5, RB7: nível baixo

LATBbits.LATB1 = 1;    // RB1 (pino 34) = nível alto
LATBbits.LATB2 = 0;    // RB2 (pino 35) = nível baixo
```

PORT B

Registrador LATB

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de leitura:

```
X = LATB;                 // X (8 bits) recebe os estados lógicos
                          // de todos os pinos do porto B

Y = LATBbits.LATB4;       // Y recebe o estado lógico de RB4 (pino 37)
```

PORT B

Rotina de inicialização (conforme folha de dados)

```
PORTB = 0b00000000;        // limpa o porto B
LATB  = 0b00000000;        // limpa o latch do porto B (opcional)
ADCON1= 0b00001111;        // configura todos pinos como digitais
CMCON = 0b00000111;        // desliga módulo comparador (opcional)
                           // padrão na inicialização
TRISB = 0b11001111;        // configurar de acordo com a
                           // utilização dos pinos
```

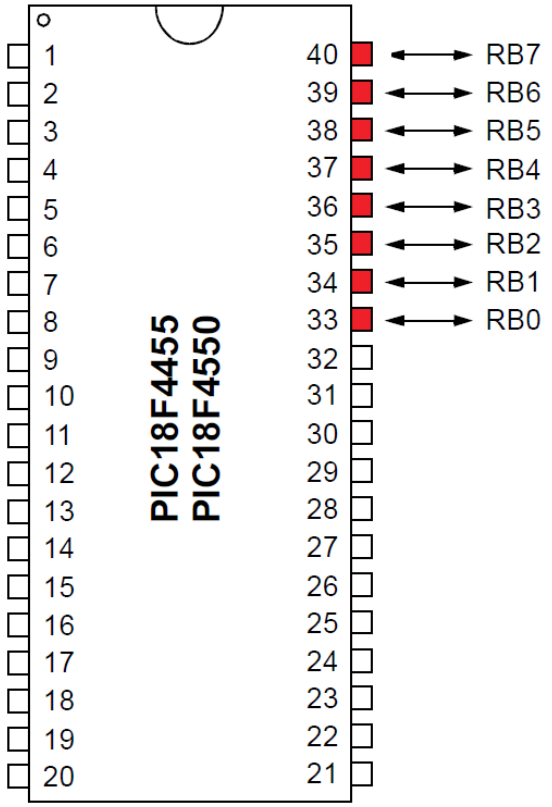
- O valor do dos pinos do PORTB é desconhecido na inicialização, então limpe o porto antes de configurar os pinos.
- Configure os pinos não utilizados como saídas no registrador TRISB.

PORT B

Exemplo de programa

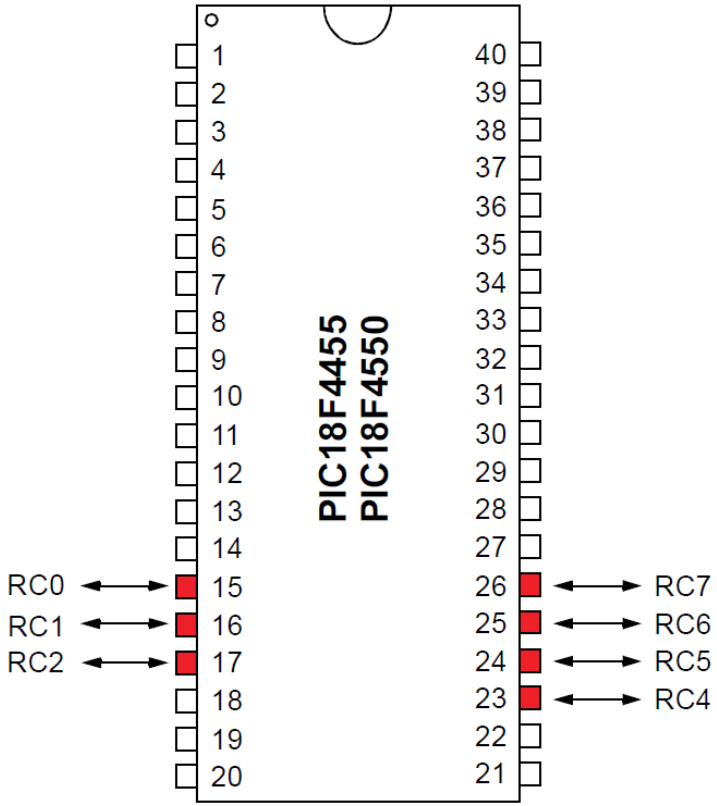
```
void main() {
    // Inicialização do porto B
    PORTB = 0b00000000;
    LATB  = 0b00000000;
    TRISB = 0b00010000;
    ADCON1= 0b00001111;

    while(1){
        if(PORTBbits.RB4 == 1){
            LATBbits.LATB0 = ~LATBbits.LATB0;
            while(PORTBbits.RB4 == 1);
        }
    }
}
```



PORT C

- 7 pinos de entrada e saída
- RC3 não está implementado
- RC4 e RC5 multiplexados com o módulo USB e podem ser usados somente como entradas digitais
- Registradores associados:
  - PORTC
  - LATC
  - TRISC



PORT C

Registrador TRISC

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TRISC7	TRISC6	-	-	-	TRISC2	TRISC1	TRISC0

- Responsável por configurar aos pinos como entradas ou saídas
  0 = saída, 1 = entrada

```
TRISC = 0b00001111;    // entradas: RC0 a RC5 (exceto RC3)
                       // configura RC6 e RC7 como saídas

TRISCbits.RC0 = 1;     // configura RC0 (pino 15) como entrada
TRISCbits.RC2 = 0;     // configura RC2 (pino 17) como saída
```

PORT C

Registrador PORTC

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTC7	PORTC6	PORTC5	PORTC4	-	PORTC2	PORTC1	PORTC0

- Responsável pela operação do pino como entrada ou saída
- Operação de escrita:

```
PORTC = 0b01010101;    // atribui níveis lógicos para todo porto
                       // RC0, RC2, RC4, RC6: nível alto
                       // RC1, RC5, RC7: nível baixo

PORTCbits.RC1 = 1;     // RC1 (pino 16) = nível alto
PORTCbits.RC2 = 0;     // RC2 (pino 17) = nível baixo
```

PORT C

Registrador PORTC

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTC7	PORTC6	PORTC5	PORTC4	-	PORTC2	PORTC1	PORTC0

- Responsável pela operação do pino como entrada ou saída
- Operação de leitura:

```
X = PORTC;              // X (8 bits) recebe os estados lógicos
                        // de todos os pinos do porto C

Y = PORTCbits.RC4;      // Y recebe o estado lógico de RC4 (pino 23)
```

PORT C

Registrador LATC

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LATC7	LATC6	-	-	-	LATC2	LATC1	LATC0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de escrita:

```
LATC = 0b01010101;      // RC0, RC2, RC4, RC6: nível alto
                        // RC1, RC5, RC7: nível baixo

LATCbits.LATC1 = 1;     // RC1 (pino 34) = nível alto
LATCbits.LATC2 = 0;     // RC2 (pino 35) = nível baixo
```

PORT C

Registrador LATC

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LATC7	LATC6	-	-	-	LATC2	LATC1	LATC0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de leitura:

```
X = LATC;                 // X (8 bits) recebe os estados lógicos
                          // de todos os pinos do porto C

Y = LATCbits.LATC4;       // Y recebe o estado lógico de RC4 (pino 23)
```

PORT C

Rotina de inicialização (conforme folha de dados)

```
PORTC = 0b00000000;        // limpa o porto C
LATC  = 0b00000000;        // limpa o latch do porto C (opcional)
TRISC = 0b11001111;        // configurar de acordo com a
                           // utilização dos pinos
```

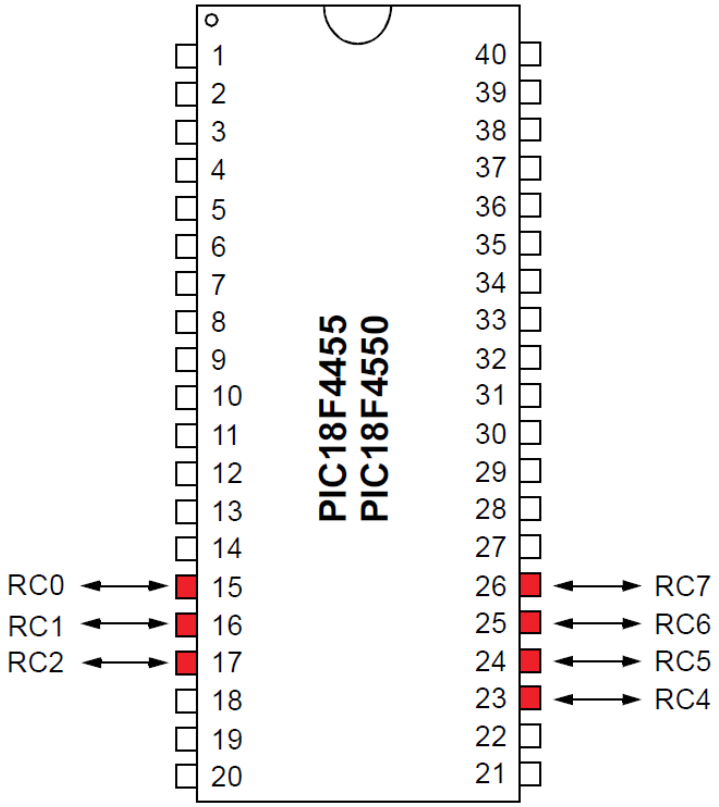
- O valor do dos pinos do PORTC é desconhecido na inicialização, então limpe o porto antes de configurar os pinos.
- Configure os pinos não utilizados como saídas no registrador TRISC.

PORT C

Exemplo de programa

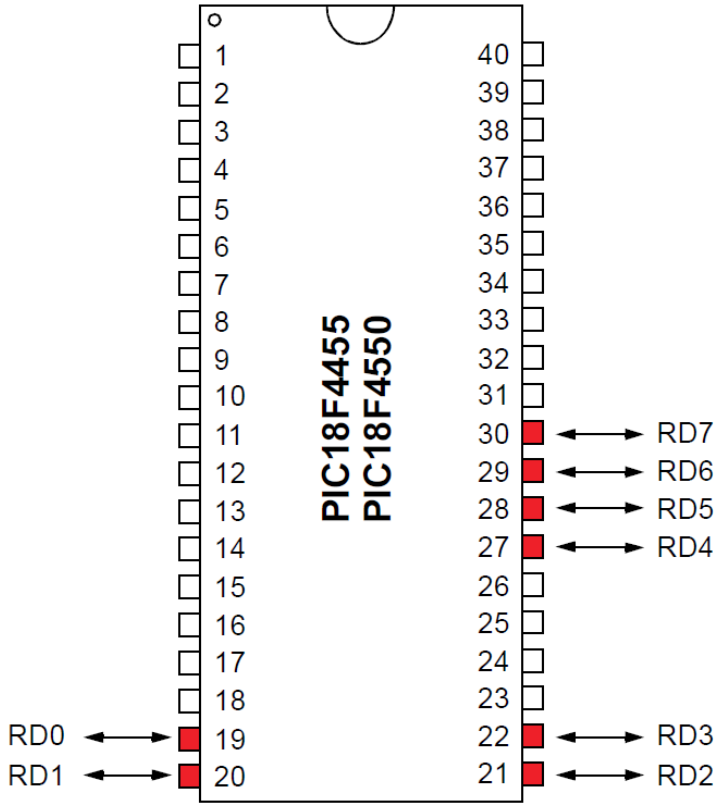
```
void main() {
    // Inicialização do porto C
    PORTC = 0b00000000;
    LATC  = 0b00000000;
    TRISC = 0b00000110;

    while(1){
        while(PORTCbits.RC1 == 1){
            PORTCbits.RC6 = 1;
        }
        while(PORTCbits.RC2 == 1){
            PORTCbits.RC7 = 1;
        }
        PORTCbits.RC6 = 0;
        PORTCbits.RC7 = 0;
    }
}
```



PORT D

- 8 pinos de entrada e saída
- Ativação de resistores de pull-up (bit RDPU do registrador PORTE)
- Registradores associados:
  - PORTD
  - LATD
  - TRISD



PORT D

Registrador TRISD

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
TRISD7	TRISD6	TRISD5	TRISD4	TRISD3	TRISD2	TRISD1	TRISD0

- Responsável por configurar aos pinos como entradas ou saídas
  0 = saída, 1 = entrada

```
TRISD = 0b00001111;    // configura RD0 a RD3 como entradas
                       // configura RD4 a RD7 como saídas

TRISDbits.RD0 = 1;     // configura RD0 (pino 19) como entrada
TRISDbits.RD5 = 0;     // configura RD5 (pino 28) como saída
```

PORT D

Registrador PORTD

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0

- Responsável pela operação do pino como entrada ou saída
- Operação de escrita:

```
PORTD = 0b01010101;    // atribui níveis lógicos para todo porto
                       // RD0, RD2, RD4, RD6: nível alto
                       // RD1, RD3, RD5, RD7: nível baixo

PORTDbits.RD1 = 1;     // RD1 (pino 20) = nível alto
PORTDbits.RD2 = 0;     // RD2 (pino 21) = nível baixo
```

PORT D

Registrador PORTD

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0

- Responsável pela operação do pino como entrada ou saída
- Operação de leitura:

```
X = PORTD;              // X (8 bits) recebe os estados lógicos
                        // de todos os pinos do porto D

Y = PORTDbits.RD4;      // Y recebe o estado lógico de RD4 (pino 27)
```

PORT D

Registrador LATD

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de escrita:

```
LATD = 0b01010101;      // RD0, RD2, RD4, RD6: nível alto
                        // RD1, RD3, RD5, RD7: nível baixo

LATDbits.LATD1 = 1;     // RD1 (pino 20) = nível alto
LATDbits.LATD2 = 0;     // RD2 (pino 21) = nível baixo
```

PORT D

Registrador LATD

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
LATD7	LATD6	LATD5	LATD4	LATD3	LATD2	LATD1	LATD0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de leitura:

```
X = LATD;                 // X (8 bits) recebe os estados lógicos
                          // de todos os pinos do porto D

Y = LATDbits.LATD4;       // Y recebe o estado lógico de RD4 (pino 27)
```

PORT D

Rotina de inicialização (conforme folha de dados)

```
PORTD = 0b00000000;        // limpa o porto D
LATD  = 0b00000000;        // limpa o latch do porto D (opcional)
TRISD = 0b11001111;        // configurar de acordo com a
                           // utilização dos pinos
```

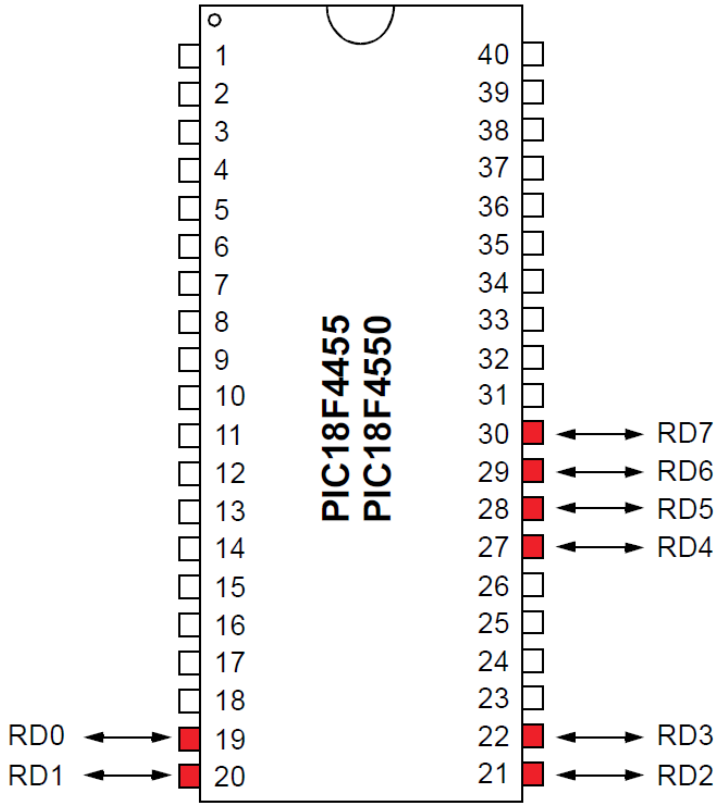
- O valor do dos pinos do PORTD é desconhecido na inicialização, então limpe o porto antes de configurar os pinos.
- Configure os pinos não utilizados como saídas no registrador TRISD.

PORT D

Exemplo de programa

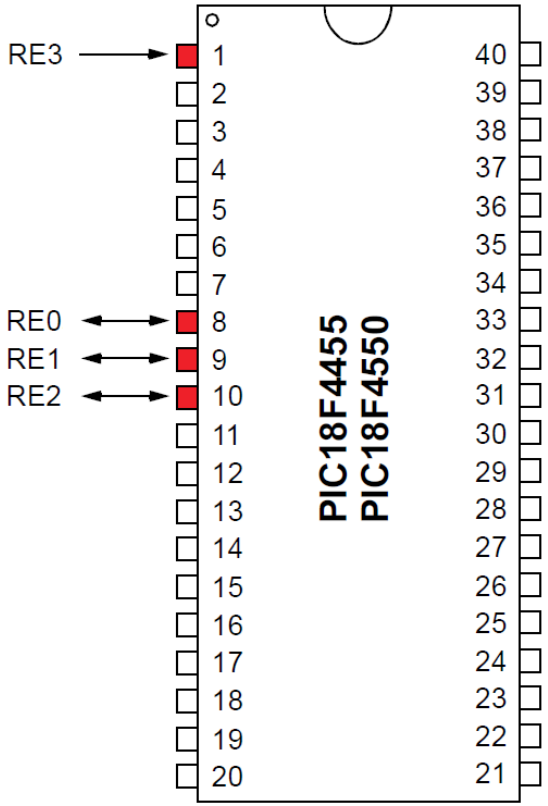
```
void main() {
    // Inicialização do porto D
    PORTD = 0b00000000;
    LATD  = 0b00000000;
    TRISD = 0b00000011;

    while(1){
        if(PORTDbits.RD0 == 1){
            PORTDbits.RD3 = 1;
        }
        if(PORTDbits.RD1 == 1){
            PORTDbits.RD3 = 0;
        }
    }
}
```



PORT E

- 4 pinos de entrada e saída
- RE0, RE1 e RE2 são configurados como entradas analógicas na inicialização (POR)
- RE3 é multiplexado com o módulo de reinicialização forçada (MASTER RESET) e só pode ser usado como entrada digital.
- Registradores associados:
  - PORTE
  - LATE
  - TRISE



PORT E

Registrador TRISE

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	TRISE2	TRISE1	TRISE0

- Responsável por configurar aos pinos como entradas ou saídas
  0 = saída, 1 = entrada

```
TRISE = 0b00000111;    // configura RE0 a RE3 como entradas

TRISEbits.RE0 = 1;     // configura RE0 (pino 8) como entrada
TRISEbits.RE2 = 0;     // configura RE2 (pino 10) como saída
```

PORT E

Registrador PORTE

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RDPU	-	-	-	PORTE3	PORTE2	PORTE1	PORTE0

- Responsável pela operação do pino como entrada ou saída
- Operação de escrita:

```
PORTE = 0b00000101;    // atribui níveis lógicos para todo porto
                       // RE0, RE2: nível alto
                       // RE1: nível baixo

PORTEbits.RE1 = 1;     // RE1 (pino 9) = nível alto
PORTEbits.RE2 = 0;     // RE2 (pino 10) = nível baixo
```

PORT E

Registrador PORTE

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RDPU	-	-	-	PORTE3	PORTE2	PORTE1	PORTE0

- Responsável pela operação do pino como entrada ou saída
- Operação de leitura:

```
X = PORTE;              // X (8 bits) recebe os estados lógicos
                        // de todos os pinos do porto E

Y = PORTEbits.RE1;      // Y recebe o estado lógico de RE1 (pino 9)
```

PORT E

Registrador LATE

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	LATE2	LATE1	LATE0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de escrita:

```
LATE = 0b01010101;      // RE0, RE2: nível alto
                        // RE1: nível baixo

LATEbits.LATE1 = 1;     // RE1 (pino 9) = nível alto
LATEbits.LATE2 = 0;     // RE2 (pino 10) = nível baixo
```

PORT E

Registrador LATE

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	LATE2	LATE1	LATE0

- Funciona como um backup do valor da saída e está isolado dos pinos físicos por um *buffer.* É útil para fazer a leitura de pinos configurados como saídas e operações *read-modify-write*.

- Operação de leitura:

```
X = LATE;                // X (8 bits) recebe os estados lógicos
                         // de todos os pinos do porto E

Y = LATEbits.LATE2;      // Y recebe o estado lógico de RE2 (pino 10)
```

PORT E

Rotina de inicialização (conforme folha de dados)

```
PORTE = 0b00000000;        // limpa o porto E
LATE  = 0b00000000;        // limpa o latch do porto E (opcional)
ADCON1= 0b00001111;        // configura todos pinos como digitais
CMCON = 0b00000111;        // desliga módulo comparador (opcional)
                           // padrão na inicialização
TRISE = 0b00000111;        // configurar de acordo com a
                           // utilização dos pinos
```

- O valor do dos pinos do PORTE é desconhecido na inicialização, então limpe o porto antes de configurar os pinos.
- Configure os pinos não utilizados como saídas no registrador TRISE.

PORT E

Exemplo de programa

```
void main() {
    // Inicialização do porto E
    PORTE = 0b00000000;
    LATE  = 0b00000000;
    TRISE = 0b00000011;

    while(1){
        if(PORTEbits.RE0 == 1){
            PORTEbits.RE2 = 1;
        }
        if(PORTEbits.RE1 == 1){
            PORTEbits.RE2 = 0;
        }
    }
}
```