Programming the Microchip Pic 16f84a Microcontroller As a Signal Generator Frequencies in Railway Automation

High School of Transport "Todor Kableshkov"
1574 Sofia, 158 Geo Milev str.

Ivan Velev

Abstract

Using a new method of obtaining a signal frequency by a programmable microcontroller Microchip PIC 16F84A, data communication and management of projects in the transport of high responsibility. It is operated in generating regular rectangular pulse Ti = TA, and their duration is programmable and determines the frequency of the signal. The microcontroller has its own LC clock, but for stable operation it is advisable to use an external crystal resonator. We use system environment MPLab of Microchip, the creation of the program itself. The program is written in assembly language, creating a file with the extension. asm.

The advantages of this method are: simple structure, low cost, high reliability.

The computer model was tested in the software Proteus7.2.

Laboratory tests were performed with a physical model to confirm reliable operation of the microcontroller.

Key words: *generator of frequency, microcontroller, programming, MPLab*

1. Introduction

The theme of the report is formulated as a result of research work of the microcontroller Microchip PIC16F84A.

PIC16F84A is an 8-bit RISC-company microcontroller Microchip, shown in Fig. 1 belongs to the family of PIC16X8X PIC16FXXX. In addition, with this group belong also: PIC16F83, PIC16CR83, PIC16F84, PIC16CR84.

The difference between the microcontrollers are mostly expressed in their operation and in the capacity of the memory, and in that some of it is reprogrammable, and the others -. Of these microcontrollers, PIC16F84A has the best features, namely:

maximum clock frequency of 20MHz (which corresponds to a minimum instruction cycle - 200ns);

a set of instructions 35, each of which has a length of one word; All instructions are executed by the microprocessor cycle with the exception of the transitions that take two cycles;

capacity of the program (Flash) memory - 1024 words;

68 bytes of RAM data items;

64 bytes EEPROM memory data items;

separate 14-bit instructions for highway and 8-bit data

hardware stack with eight levels;

Four sources of interrupting, respectively:

- From an external source (INT);
- Overflow of Timer 0;
- In case of the bits 4 to 7 of the PORT B;
- Upon completion of the entry in the EEPROM memory;

More than 1000 cycles erase / write FLASH program memory and 1,000,000 to EEPROM memory;
During storage of the data stored in non-volatile memory EERROM - more than 40 years;
Frequency of the supply voltage - from 2 to 5,5 V;
Consumption:
- Offer a small current of 2mA of power at 5 V and a clock frequency of 4 MHz
- Current 15tsA at 2V supply and clock frequency 32kHz;
- Current 0.5 / uA in SLEEP mode and supply voltage 2V.

Operating temperature range:
- 0 to +70 ° C - for commercial purposes;
- From -40 to +70 ° C - for industrial purposes.

In PIC microcontrollers and in particular in P1C16F84A basic connection diagram shown in Fig. 1 contains frequency element (depending on the type of generator) and elements that make up the chain reset (FT). As a timing element in this case is used quartz resonator - X1 running at 4MHz, This is the clock frequency that will work itself microcontroller.

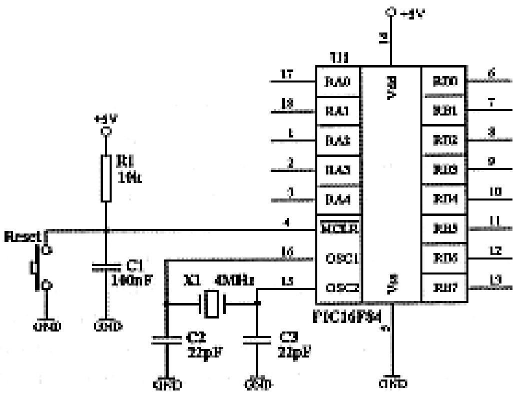

Fig.1

Depending on the needs, can be used, and other resonators, but it should be borne in mind that these will inevitably be accompanied with a change in timing of the program in question. This is cause for ¬ From clock directly dependent time that runs every hin ¬ ling or alteration, and hence the program. Provided circuit reset (FT) includes "Reset" button and RC built elements Rl, C1. By pressing the "Reset" is carried FT MCLR. and the purpose of the capacitor C1 to filter out transient input interference that microcontroller restarted. Also, thanks to the use of this capacitor, this artificial cc realized externally FT at power (POR).
In principle, the supply voltage to the microcontroller may be varied between 2 and 5V, but in most cases (including here) is applied 5V. This value is suitable for two reasons: first, it can relatively easily be obtained by using the popular stabilizer 7805 and a second, relatively large part of the digital integrated circuits (which may be connected to microcontroller and to work together with it), using such power. An exemplary embodiment of the pattern, which can be prepared feeder voltage - is shown in Fig. 2.

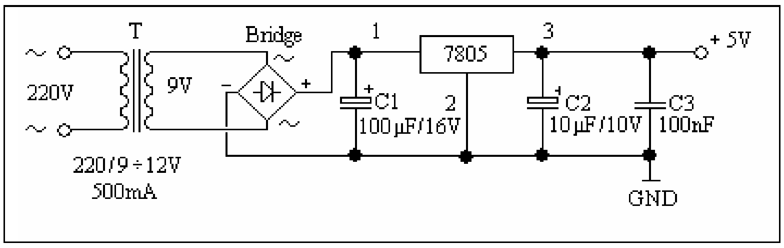

Fig.2

Here, the desired value or 5V CE prepared beforehand line voltage falls to a value of about 9-12V, and then stand by Graecia bridge circuit (Budge) and filtered by electrolytic capacitor C1, and thus formed upright and filtered voltage is fed to compensation stabilizer 7805 that supports (stable) value of 5V, capacitors C2 and NW are to be used for additional filtration and bipolar capacitor prevents the microcontroller from RF interference coming from the mains. Thus, to achieve a better effect, it is recommended that the capacitor to be located as close to microcontroller (dandruff when installing the circuit board).

2. Materials and Method

Generating A Square Wave With A PIC-Microcontroller

With this realization, we will demonstrate how a conclusion from the microcontroller can receive and highlight, the simplest possible signal - a series of regular, rectangular pulses. I will explain in more detail how it works with PIC16F84A microcontroller. The duration of the pulses in this case can be set programmatically, and since it is a regular pulse (duration and pulse pause pa equal to Ti = TA. When specifying its duration will be very clearly defined and signal frequency (F = l / 2 Ti). this signal must be received on conclusion RB0. Having in mind that this series is actually a sequence of zeros and ones of the same length, then the receiving terminal and RB0, sequential programs need to reset and assigned to a unit

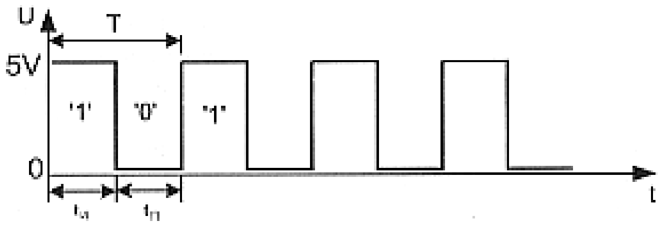

Fig.3

Resetting the conclusions it with instructions "BCF PORTB, 0," and the establishment in one - with similar such "BSF PORTB, 0". To obtain, however, the desired duration of the pulse
(Tи) and pause (Tп), respectively, the desired frequency, it is necessary after every down or reset the terminal to be measured given time interval (also called time-delay) equal to duration pulse (or pause). Once the algorithm description, which will generate a pulse signal is addressed and the program through which this algorithm will be realized.
At first LIST directive is through the type of the microcontroller (CPU).
On the next line, they shared program code includes the header file, "p16F84A.inc", contains the definitions of all SFR registers (of the corresponding microcontroller) with their bits. This and all other header files used for granted as they are set in the middle and MPLAB are recorded on the hard disk for installation in. The Directive "__CONFIG_CP_OFF & _WDT_0FF & _PWRTE_ON & XT OSC", to ASM-file information is added to the configuration of the microcontroller, which is loaded directly used charter recording software.
In this case, the protection code (Code Protection) and timer WDT is disabled. "CP_ OFF & _WDT_OFF", timer power-included "_PWRTE ON", and the type of generator c "_XT_OSC" quartz resonator. In the section "Defining Constants and Variables" Now therefore define the addresses used in the program general-purpose registers (GPR), plus the values of the various constants. In this case, the specific program used a GPR register

(Counter) and two constants w and f. These constants will be used when setting a destination (d), in the instruction requiring it. For example, if the result of the operation must be recorded in the working register (W), instead of "dccfsz Counter, 0" will record "dcctsz Counter, w". Similar will be recorded, where the result should be recorded in a register f, participating in the operation "dccfsz Counter, 1" - "dccfsz Counter, f". On the following lines define the start address of the recording of the program in the microcontroller and will move to the beginning of the program. In this case, the instruction "goto start" will be located of address 000h, corresponding to the reset vector. Thus, in the FT of the microcontroller will execute this instruction first, thanks to which, the program will start from the address specified by the label "start". The need for this conversion lies in the fact that the interrupt vector is located at address 0004h, three upstanding cell vector FT. In this sense, if used continuously (they currently do not have), address 0004h is reserved and can not contain any instructions from the main program, so it should be skipped. To avoid fragmenting the program however, the straight ss jump remaining three cells located between two vectors.
Thus the user program starts immediately after the address corresponding to the interrupt vector. The program may be conditionally divided into three parts: "Start the program", "Used subroutines" (they shared them can be assigned to program service of interrupt) and "Main Program"
At the beginning of the program shall be called. starting initialization microcontroller. It generally lies in charge of the register (OPTION_REG) - log options and initialization of input and output ports.

When changing the status of individual bits can OPTION_REG:
- Authorize or prohibit the pull (pull-up) registers - bit 7;
- Sets the pulse for switching from terminal RV0) - bit 6;
- Selects the source clock so TMR0 - bit 5;
- Sets the pulse of TMR0 when working with an external clock 4-bit
- Selects the divisor function - bit 3;
- Sets the coefficient of the divisor - bits 2-0.

```
        list            p=16F84A        ; Задава типа на процесора
          #include      <p16F84A.inc>   ; Дефиниции за SFR регистрите
  __CONFIG      _CP_OFF         & _WDT_OFF & _PWRTE_ON & _XT_OSC
;*******Дефиниране на константи и променливи**************************
wEQU    0
f EQU   1
Counter EQU   H'0C'
;*********************************************************************
; Потребителска програма
;*********************************************************************
  ORG   H'0000'                 ; Вектор на НУ
  goto  start                   ; Преход към началото на програмата
;*********************************************************************
; Начало на програмата
;*********************************************************************
start     bsf     STATUS,RP0            ; Избор на Банка 1
  movlw  B'11010111'            ; 11010111b -> W
  movlw  OPTION_REG   ; (W) -> OPTION_REG
  bcf    STATUS,RP0             ; Избор на Банка 0
  clrf   PORTB          ; 00h -> PORTB
  bsf    STATUS,RP0             ; Избор на Банка 1
  movlw  B'11111110'            ; 11111110b -> W
  movwf  TRISB                  ; RB7-RB1-Входове, RB0-Изход
  bcf    STATUS,RP0             ; Избор на Банка 0
  goto   main
;*********************************************************************
; Използвани подпрограми
;*********************************************************************
;------   Подпрограма "Wait",    формираща времезадръжка от 460 ms ------
Wait      movlw  D'7'                   ; 7 -> W
  movwf  Counter        ; (W) -> Counter
  clrf   TMR0                   ; 00h -> TMRO
```

```
again     bcf      INTCON, T0IF   ; Нулирай бит T0IF
loop      btfss    INTCON, T0IF   ; Бит T0IF=1?
  goto    loop                    ; Не, провери отново
  decfsz  Counter,f               ; Да, Counter=(Counter)=1, Counter=0?
  goto    again                   ; Не, изпълни цикъла отново
  return                          ; Връщане към главната програма
;*****************************************************************
; Главна програма
;*****************************************************************
main      bsf      PORTB,0              ; RBO=1
  call    Wait                    ; Изчакай 460ms
  bcf     PORTB,0                 ; RBO=0
  call    Wait                    ; Изчакай 460ms
  goto    main                    ; Повтори цикъла отново
  END                             ; Директива за край на програмата
```

In this program are selected and are important following options:

- Internal clock module Timer 0 - TMRO content will be increased every cycle of the instruction;
- Work with the divisor Timer 0;
-Factor of division: 1:256.

Although OPTION_REG loaded at the beginning of the program if at work and have to change one or the above options, there is no problem it is loaded with a new value. The only thing that should not be forgotten is the registry entries OPTION_REG be preceded by a choice of Bank 1, as it is situated in it. The same goes for the record registers the direction of data TR1SA and TRISB.

As in the case of the microcontroller is used only conclusion RB0, the only thing that should be done in the init is the conclusion is defined as output (lines 22,23). You will note only that the reset bit of the register for the direction of data defining the terminal as an output, and the establishment of the same bit to "1", defined as an input terminal. If all terminals of a given port are used as inputs, the latter does not need to be initialized at the start of the program because the FT in the microcontroller, and all terminals are defined as inputs.

After starting initialization microcontroller begins execution of the main program (lines 42-46). The latter consists of an endlessly repeating cycle ss, wherein ss performed sequentially down to "1" and the reset terminal RB0.

To set the pulse duration and the pause time (time outputting RBO is maintained at "1" or "0") after each instruction for setting or resetting the terminal, it is necessary to dispense the time interval of a duration equal to the pulse duration (or pause). For this purpose, after each of these instructions subroutine "Wait" (Wait - wait) by which a measured time interval desired.

Subroutine "Wait" is implemented using TMR0 (Timer 0) and can say it is used in almost all of the programs and can realize time intervals of varying duration. This program measures time by counting each full to register TMR0 (by value FFh to 00). When it is used the fact that the TMR0 overflow takes some time, depending on the selected division ratio, the initial value loaded into TMR0 register and a cycle time of instructions (when working on Timer 0 with internal clock). The relationship between these three parameters and time prepalvaps of TMR0 can cc gave the following relationship:

$$Tover = [256 - TMR0] \cdot \frac{1}{Kps} \cdot \frac{4}{Fosc}$$

wherein:

Tover - time which overflows (Overflow) TMR0 [ms];

(TMR0) - initial value loaded into register TMR0;

Kps - coefficient of division (for example, if a selected ratio 1:256, 1/Kps = 256);

4/F osc - a cycle of instruction [ms];

F osc - speed, microcontroller [MHz].

Using this expression and maneuvering some of the parameters (most often these are Kps or (TMR0), can obtain many different values for the time interval Tover, Taking into account the values of the parameters that are valid for consideration program: Kps = l: 256, Fosc = 4MHz, (TMR0) = 0 we get;

$$Tover = [256 - 0] \cdot 256 \cdot \frac{4}{4} = 65536 \mu s,$$

Which means that an overflow TMR0 in the case would take approximately 65,5 ms. In subroutine "Wait" for the right moment to full KBC TMR0 cc judged by determine flags bit T0IF (INTCON <2>) to "1". This practice test is implemented in cycles, located in rows 34-35 or program. An adequate examination, however, before entering into this cycle should be beaten T0IF reset "bcf 1NTCON. T0IF ".

In any overflow TMR0, the value of register "Counter" (with which he is charged at the beginning of the subroutine - lines 30, 31) is reduced by one. If the result of this operation is different from zero, the execution of the subroutine continues ("golo again"), but otherwise the place back to the main program - "return" (46). Thus, in the subroutine "Wait" to form

a two-cycle:

- Through the first monitor overflow TMRO,
- By the second - for their number.

Since the value loaded into register "Counter" is 7 and the total time that will be measured by the program in this case is 7x65, 5ms, approximately 460m from the foregoing that, by changing the value loaded into register "Counter", the total time can be varied in the range of 65,5 ms (at Counter = l) to 16,7 s (at Counter = 255).

This does not have time limits that can be obtained through this program. In case you need a time interval less than 65,5 ms can maneuver through Tover and if necessary time interval greater than 16,7 s, you just have to organize one, two or as many additional cycles need to monitor number of overflow TMR0. For example, using one more cycle time interval of maximum length will be: 255x255x65, 5ms, i.e. about 4259s (approximately 1 hour and 20 minutes).

Considering the time interval provided by one examined subroutine, we can determine the frequency of the output signal received on conclusion RB0, namely:

$$f = \frac{1}{T} = \frac{1}{tu + tn} = \frac{1}{460.10^{-3} + 460.10^{-3}} \approx 1Hz$$

If necessary, the preparation of a higher (lower) than the frequency need to be reduced (increased), the length of the time interval formed by the subroutine "Wait".

To be loaded into the microcontroller need to convert the file into machine code. This is also done in the system environment MPLab, as the new file already obtained an extension *hex*.

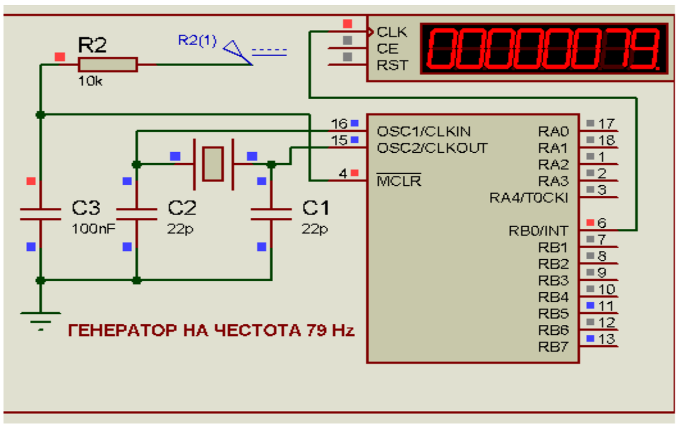

Fig.4

Fig 4 shows how to generate a frequency 79 by microcontroller PIC16F84A and is committed to the program Proteus 7.0

Reference to a sites:

[1] PIC16F84A Data Sheet - Microchip-www.microchip.com
[2] www.constructor.bg/mk/