

PIC PIANO

Пианино на ОСРВ



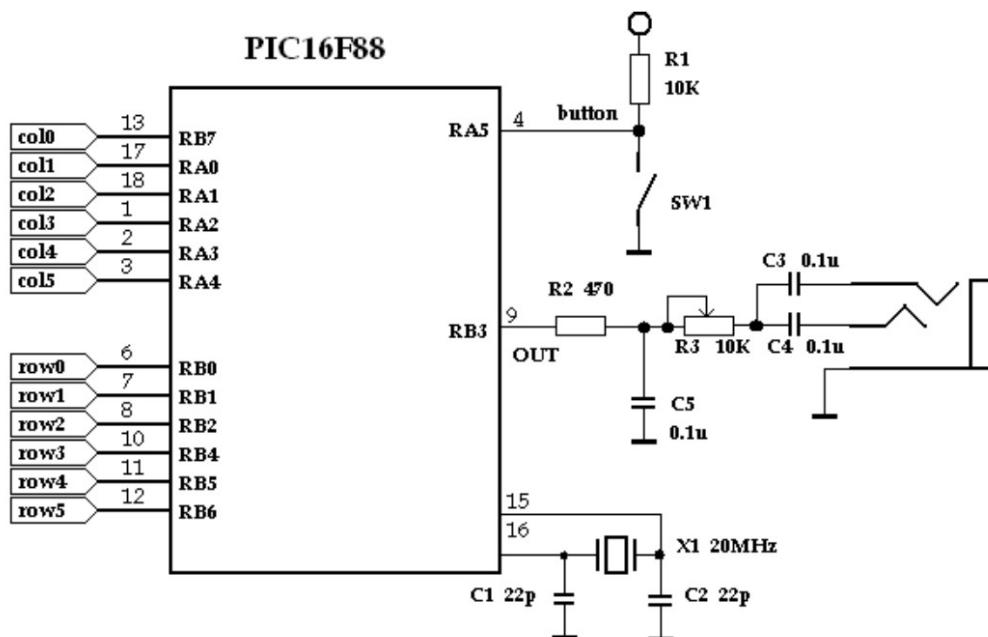
PIANO PIC HOBBY PROJECT (Rayns Ranel)
https://www.youtube.com/watch?v=ophqt_RmiS0

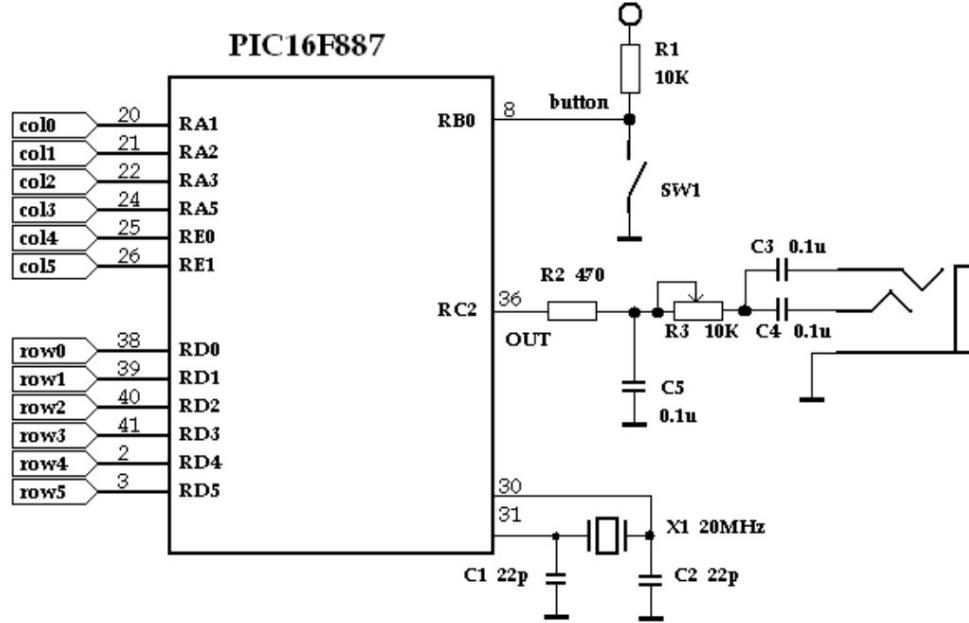
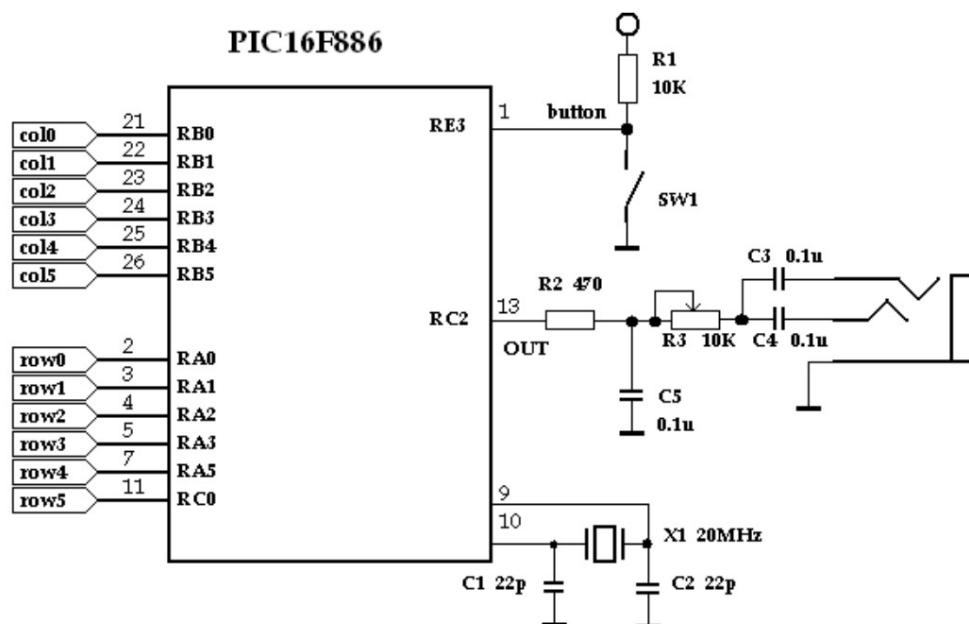
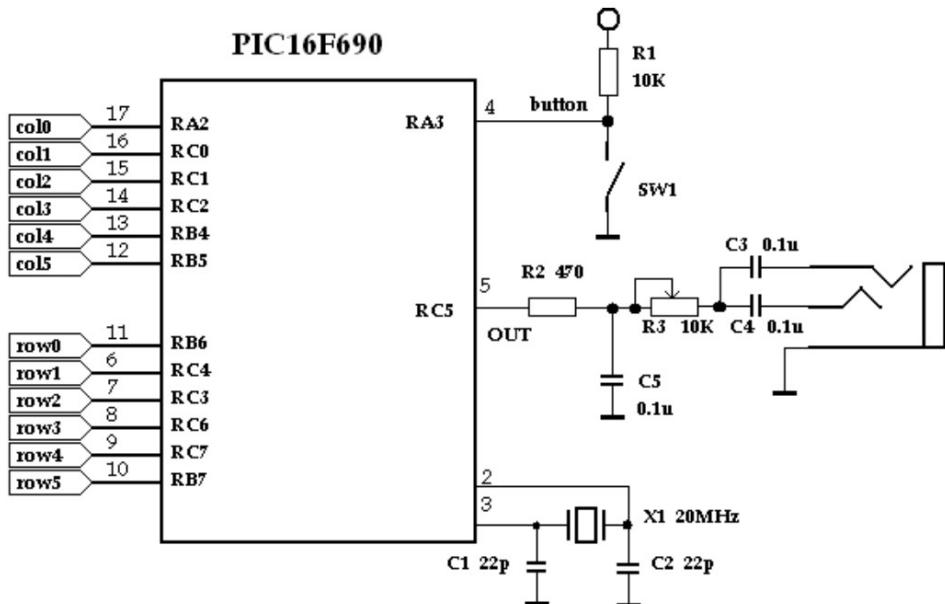
Здесь 2-х минутное видео с демонстрацией того, что описывается в примере (пианист из меня, конечно, никакой).

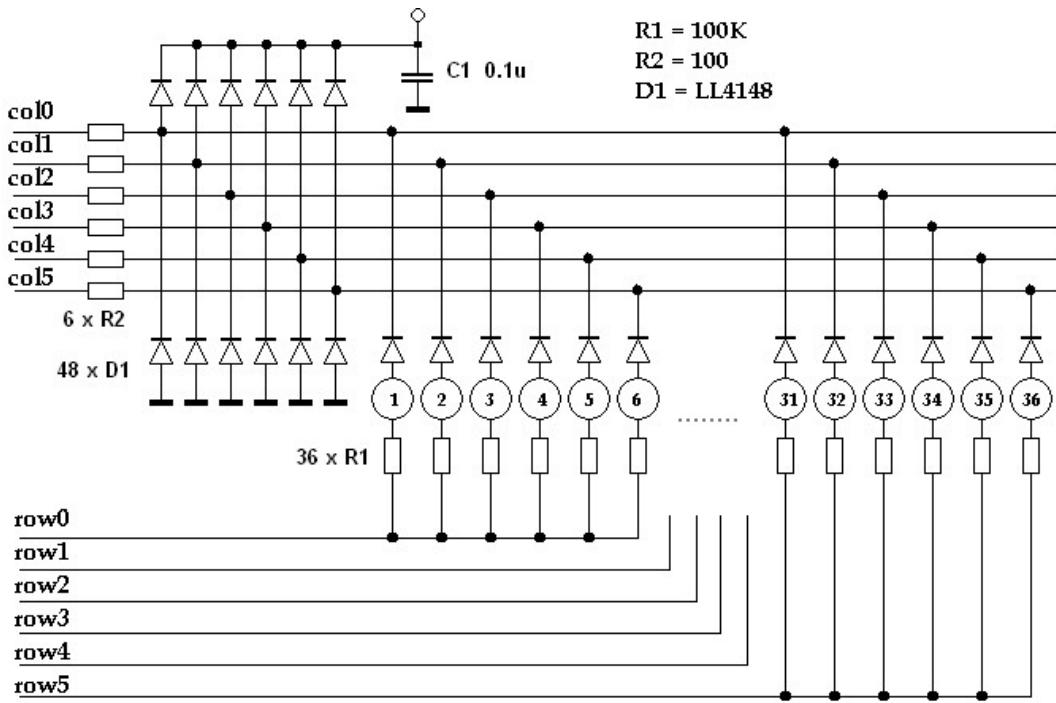
Введение

В данной статье рассматривается возможность обработки сенсорной клавиатуры с применением АЦП. В качестве примера разработаем программу «Пианино», обрабатывающую 36 сенсорных кнопок (3 октавы). Для интереса сделаем его многоголосым. В качестве аппаратной базы будем использовать демо-платы из набора pickit2 на базе контроллеров PIC16F690, PIC16F887 или PIC16F886.

В опытном образце применен контроллер PIC16F88. Честно признаюсь: проверял только на нем, поскольку других под рукой не оказалось. По мере их доставания буду проверять. А если кто-нибудь собирает пианино и обнаружит, что что-то не работает, не стесняйтесь ругаться на osa@pic24.ru, будем исправлять.







Init()

Весь текст я здесь приводить не буду (его можно посмотреть в исходных текстах, прилагаемых к статье), т.к. из-за того, что эта функция предусматривает работу на 4-х разных контроллерах (16F886, 16F887, 16F690 и 16F88), то код ее довольно громоздкий из-за наличия условных директив `#ifdef...#endif`.

Скажу только, что в этой функции производится инициализация:

- потоков ввода/вывода;
- АЦП;
- таймеров;
- модуля ШИМ;
- прерываний.

Конфигурация OSA

Для конфигурирования работы операционной системы в нашем проекте воспользуемся утилитой OSACfg_Tool.

1. Выбираем папку, где располагается наш проект

Для этого в самом верху окна справа нажимаем кнопку Browse. Там выбираем путь к файлу OSACfg.h - путь к нашему проекту (`«C:\TEST\PICKIT2\PIANO»`). Нажимаем OK. Если файл еще не создан, то программа спросит у Вас, действительно ли Вы хотите создать этот файл. Смело отвечаем «Yes» и идем дальше.

2. Выбираем имя проекта

В поле Name можно ввести имя проекта. Этот пункт необязателен, а имя вводится исключительно для наглядности, чтобы не путаться потом, какой файл от какого проекта. Мы введем в эту строку «ПИАНИНО».

3. Выбираем платформу

Также необязательный пункт. Служит только для того, чтобы пользователь при конфигурировании файла в реальном времени наблюдал предполагаемый расход оперативной памяти операционной системой. Для успокоения выберем платформу: 14-бит (PIC12, PIC16)(ht-picc). Теперь при изменении настроек мы автоматически в рамке RAM statistic будем видеть, сколько байтов в каком банке памяти израсходовано.

4. Конфигурируем наш проект

Учитывая, что мы решили не использовать приоритеты (т.е. все задачи сделать равноприоритетными), можно установить галочку напротив пункта Disable priority. Это сократит размер кода ядра операционной системы и ускорит работу планировщика.

Далее, нам обязательно нужно выбрать количество задач ОС, которые будут работать одновременно. В нашем случае - 3 (по количеству задач, создаваемых сервисом OS_Task_Create; как уже было сказано раньше, 4-я задача у нас не является задачей ОС и располагается в обработчике прерывания). Учитывая, что сама программа использует много переменных, есть смысл все системные переменные затолкать в какой-нибудь верхний банк памяти, например bank2 (в поле OSA variables bank).

Теперь нам нужно сказать системе, что нам требуются два статических таймера для работы. В поле Static timers устанавливаем 2 и в таблице ниже вводим имена идентификаторов статических таймеров: ST_KEYBOARD и ST_BUTTON. Кроме того, учитывая, что нам не потребуются задержки длиннее 256 системных тиков, установим тип статического таймера char.

И последнее: для ускорения обработки сервиса OS_Timer установим галочку напротив пункта Use in-line OS_Timer().

5. Сохраняем и выходим

Жмем на кнопку Save, чтобы сохранить отредактированный файл конфигурации, и выходим из программы нажатием на кнопку Exit.

Прошивка контроллера

Сборка проекта

Для работы с проектом нам нужно иметь установленную интегрированную среду [MPLAB IDE](#), установленный компилятор [HI-TECH PICC STD](#) (PRO-версия не подойдет).

Скачиваем, если еще не скачали, [файлы операционной системы OSA](#), распаковываем этот архив на диск C: (должна получиться папка C:\OSA).

Распаковываем файл [piano.rar](#) в папку C:\TEST\PICKIT2. При этом внутри создается папка PIANO. В MPLAB IDE открываем проект, в названии которого присутствует номер контроллера, который Вы собираетесь использовать: 886, 887, 690 или 88. Например, для демо-платы на базе 16F887 нам нужно открыть файл pk2_piano_887.mcp.

Примечание. При распаковке в другую папку, отличную от C:\TEST\PICKIT2\PIANO, нужно будет через меню Project\Build options...\Project в закладке Directories в списке include-путей заменить путь к файлам проекта на тот, куда Вы распаковали файлы из архива.

Выполняем сборку нажатием **Ctrl+F10**.

Прошивка

Здесь все просто:

1. подключаем программатор;
2. в меню «Programmer\Select» programmer выбираем PicKit2;
3. В настройках «Programmer→Settings» выбираем <3-State on <Release from Reset>
4. запускаем программирование «Programmer\Program»;
5. освобождаем вывод MCLR «Programmer\Release from reset».

Вот и все!

Схемы подключения клавиатурной матрицы

Здесь я просто приведу к каким выводам подключается матрица, чтобы работала наша программа. Рассмотрены контроллеры PIC16F88, PIC16F690, PIC16F886, PIC16F887.

Важное примечание: на выводах, используемых для чтения сенсорных датчиков (*col1*, *col2* и т.д. - аналоговые входы) не должно висеть ничего, кроме самих датчиков, т.к. любой внешний элемент будет вносить свои емкость и сопротивление.

Аналоговые входы контроллера дополнительно защищены от статики согласно документа от Microchip [Layout and Physical Design Guidelines for Capacitive Sensing \(PDF\)](#). Хоть контроллер и имеет встроенную защиту, тем не менее, во-первых, в некоторых случаях ее может оказаться недостаточно, а во-вторых, защиту имеют не все выводы (например, RA4 ее не имеет).

Заключение

Я уверен, что если у читателя хватило терпения дочитать досюда, то у него без труда хватит терпения собрать матрицу и клавиатуру и, подключив все это дело к демо-плате из набора PicKit2, насладиться игрой на собственноручно сделанном музыкальном инструменте.

Итак, в статье были рассмотрены две интересные для многих начинающих программистов темы: обработка кнопок и генерация звука. Причем рассматривались не просто кнопки, а сенсорные кнопки, что расширяет область их применения (они могут работать и в сырости, и в пыли); и мы разобрались с генерацией не просто звука, а многоканального звука. Уверен, что знания, полученные при прочтении данного пособия, а возможно и какие-то программные наработки, расширят Ваши возможности и позволят свои программы украшать интересными интерфейсными решениями.

Зачем мы воспользовались RTOS?

- Во-первых, конечно же, для того, чтобы не думать о том, как бы нам самим реализовать распараллеливание процессов.
- Во-вторых, четко разбили все на задачи, каждая из которых получилась простой наглядной. Кроме того, любая из задач может быть с легкостью модифицирована под конкретные нужды.
- Наконец, сделали наши задачи независимыми, что позволит внедрять их в другие программы, или расширить возможности этой (например, можно дописать задачу, отсылающие задаче Task_Sound сообщения, чтобы просто проигрывать любую зашитую в контроллер мелодию).

Наша программа написана с тем учетом, чтобы она могла быть легко модифицирована под любые пожелания:

- хотим кнопки перевесить на другие выводы контроллера - на здоровье: правим массивы **ROWS** и **COLUMNS** в файле **const.h**;
- хотим больше кнопок - пожалуйста, переопределяем константы **KBD_ROWS** и **KBD_COLUMNS** и добавляем данные о новых выводах в массивы **ROWS** и **COLUMNS**;
- хотим больше каналов - увеличиваем константу **MAX_CHANNELS** и снижаем частоту семплирования;
- хотим новых инструментов - правим массив **SAMPLES** в файле **sinus.c**.
- и т.д.

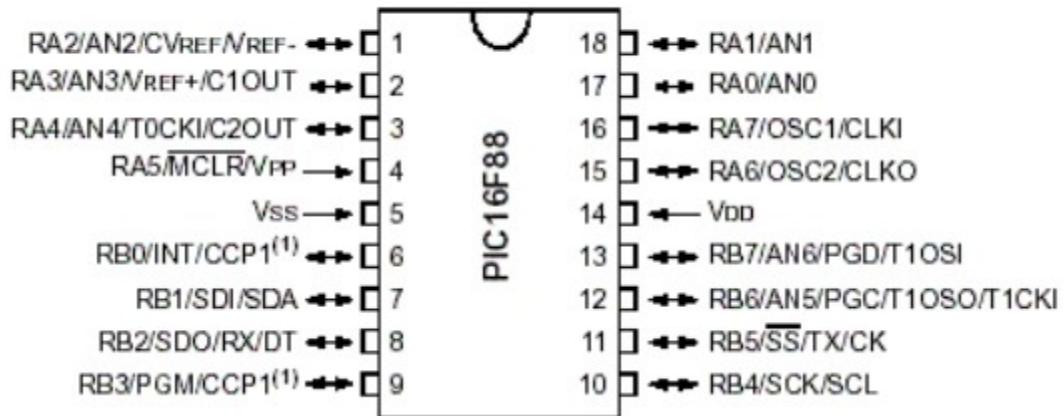
Все ограничивается только фантазией!

Удачи!

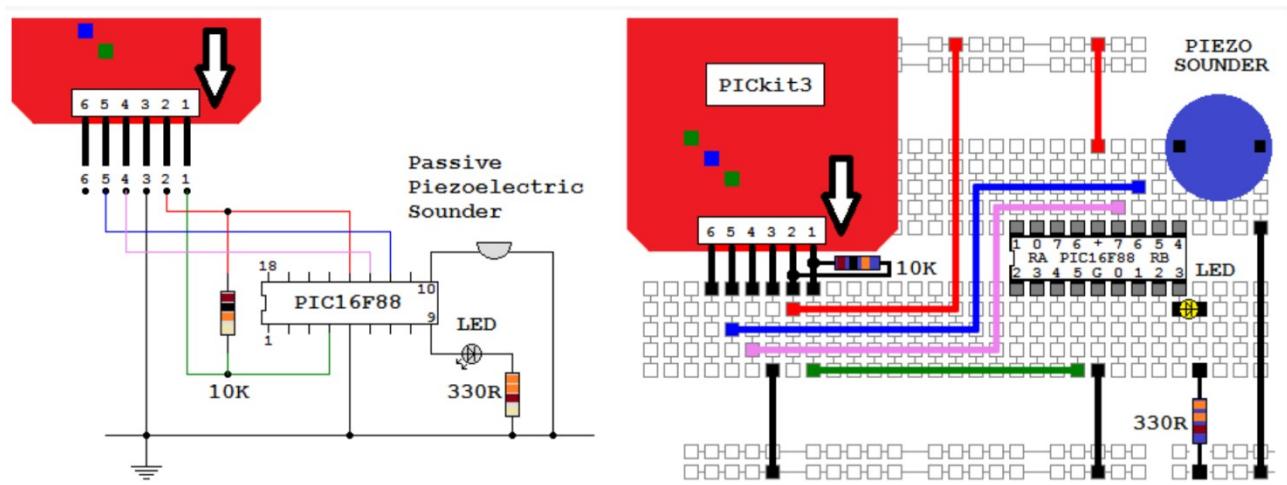
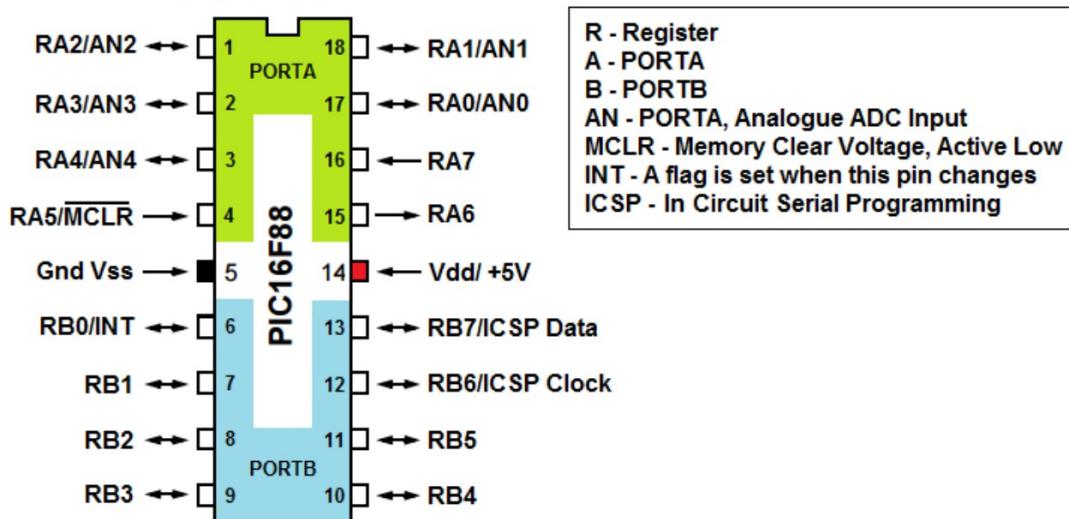
Російські монети «1 рубль» (білі) і «50 копійок» (жовті)

Виктор Тимофеев, апрель 2009
osa@pic24.ru

18-Pin PDIP, SOIC

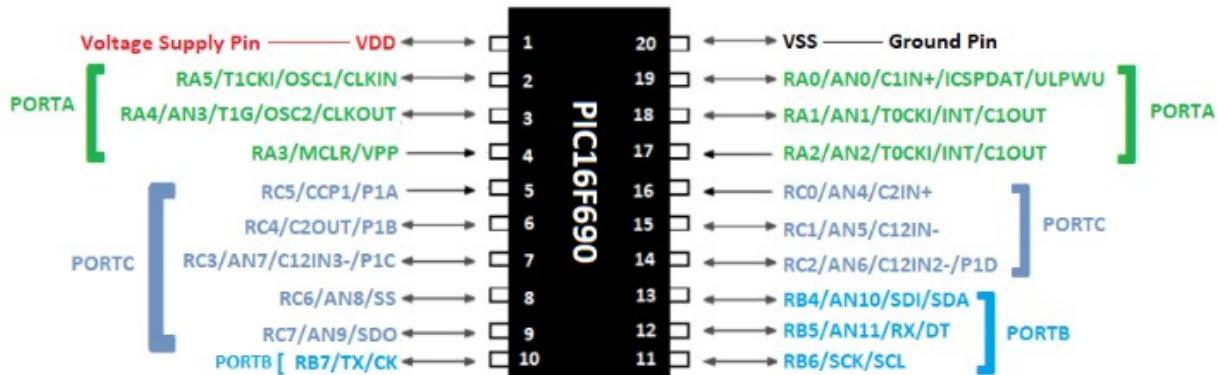


18-Pin PDIP, SOIC

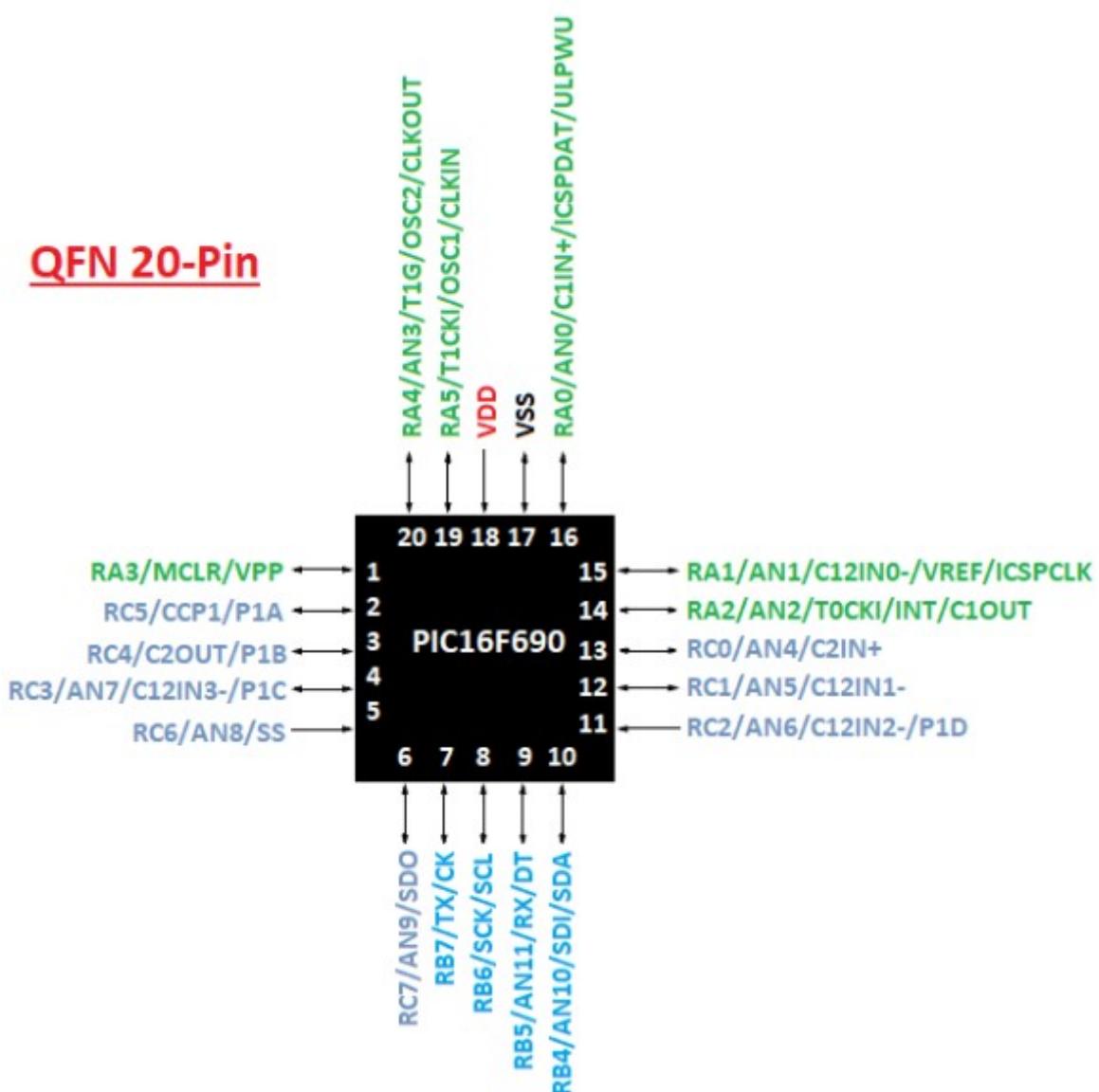


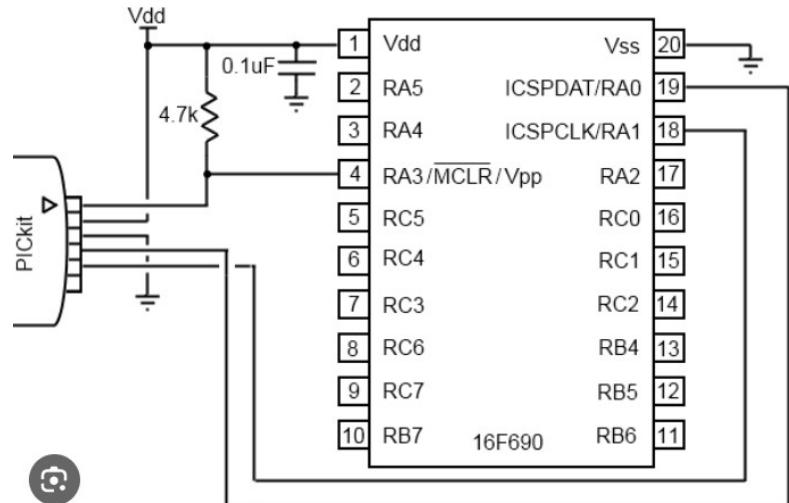
<https://reviseomatic.org/rOmV4/rOmV4/page/136/PIC16F88>

PDIP 20-Pin



QFN 20-Pin

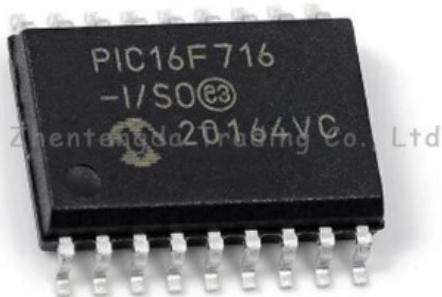




PIC16F88-I SO

US \$3.65 Shipping: US \$0.96

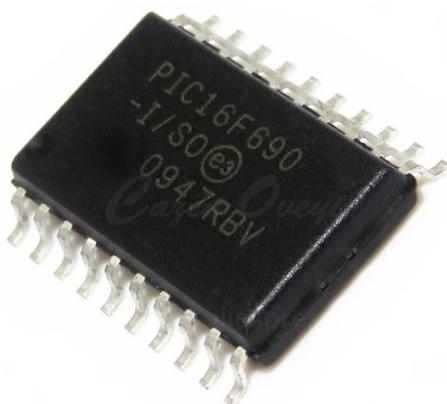
<https://www.aliexpress.com/item/1005003943541335.html>



PIC16F690-I SO (SOIC-20)

US \$0.79 Shipping: US \$0.05

<https://www.aliexpress.com/item/1005005048044296.html>



PIC16F887-I/PT (QFP-44)

US \$1.78 Free Shipping

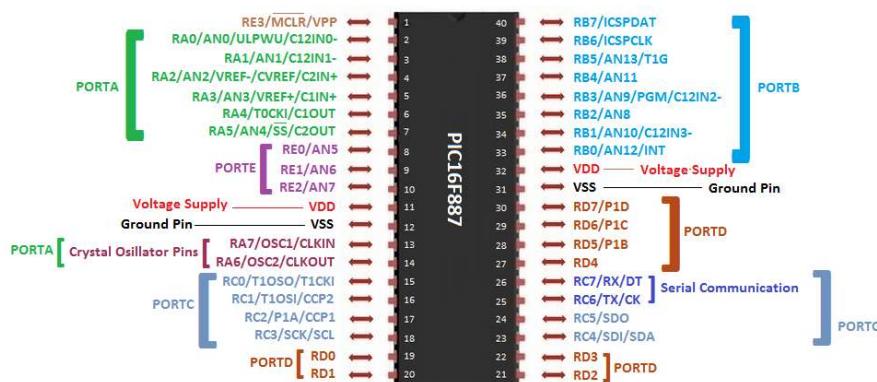
<https://www.aliexpress.com/item/1005005547689669.html>

PIC16F882/883/888

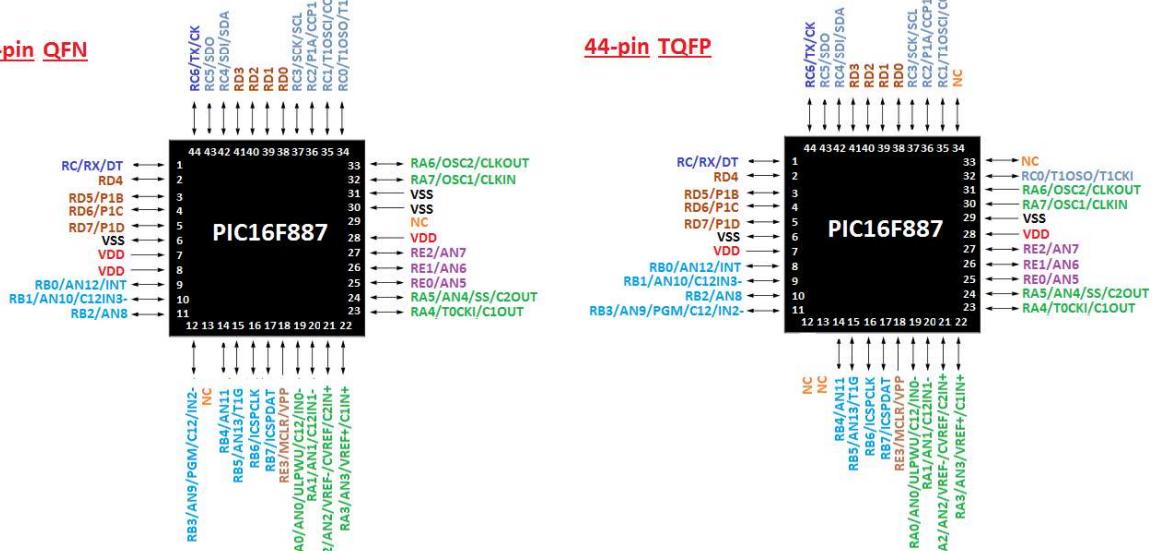
RE3/MCLR/VPP	1	RB7/ICSPDAT
RA0/AN0/ULPWU/C12IN0-	2	RB6/ICSPCLK
RA1/AN1/C12IN1-	3	RB5/AN13/T1G
RA2/AN2/VREF-/CVREF/C2IN+	4	RB4/AN11/P1D
RA3/AN3/VREF+/C1IN+	5	RB3/AN9/PGM/C12IN2-
RA4/T0CKI/C1OUT	6	RB2/AN8/P1B
RA5/AN4/SS/C2OUT	7	RB1/AN10/P1C/C12IN3-
VSS	8	RB0/AN12/INT
RA7/OSC1/CLKIN	9	VDD
RA6/OSC2/CLKOUT	10	VSS
RC0/T1OSO/T1CKI	11	RC7/RX/DT
RC1/T1OSI/CCP2	12	RC6/TX/CK
RC2/P1A/CCP1	13	RC5/SDO
RC3/SCK/SCL	14	RC4/SDI/SDA



40 pin PDIP

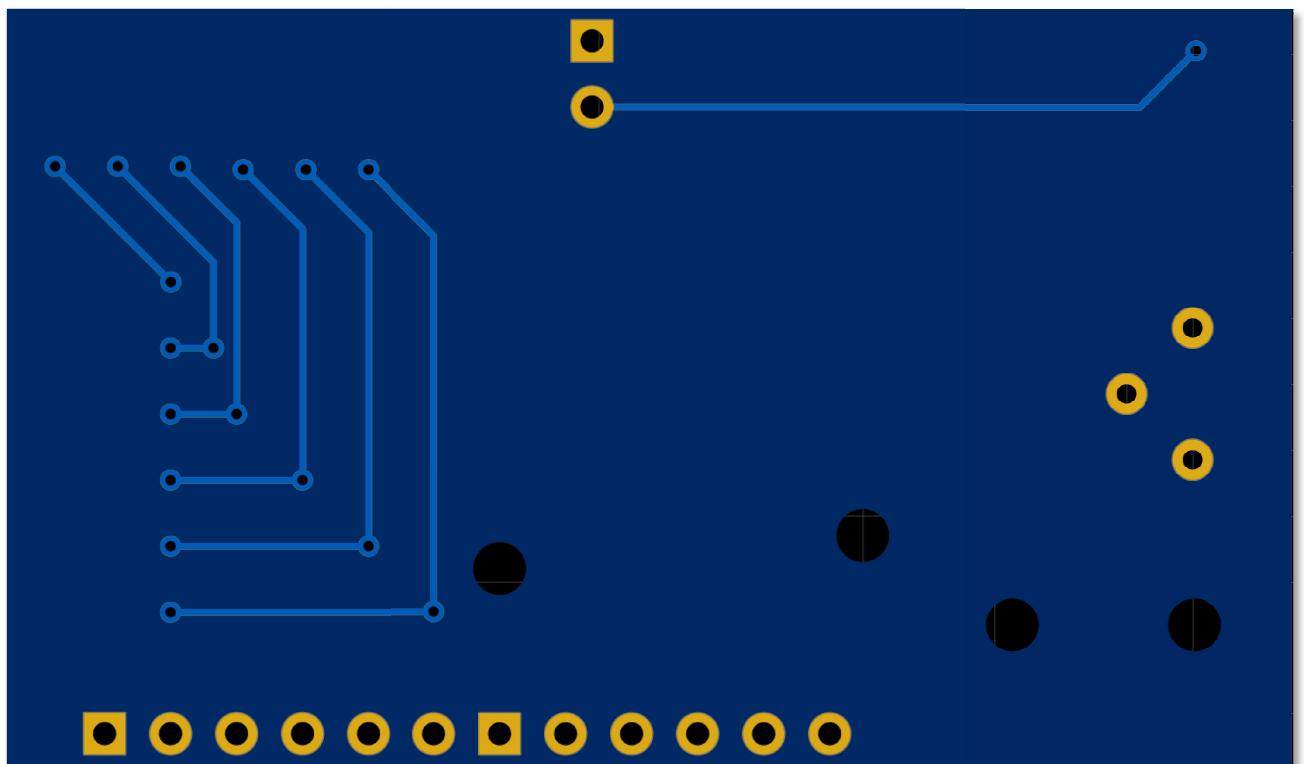
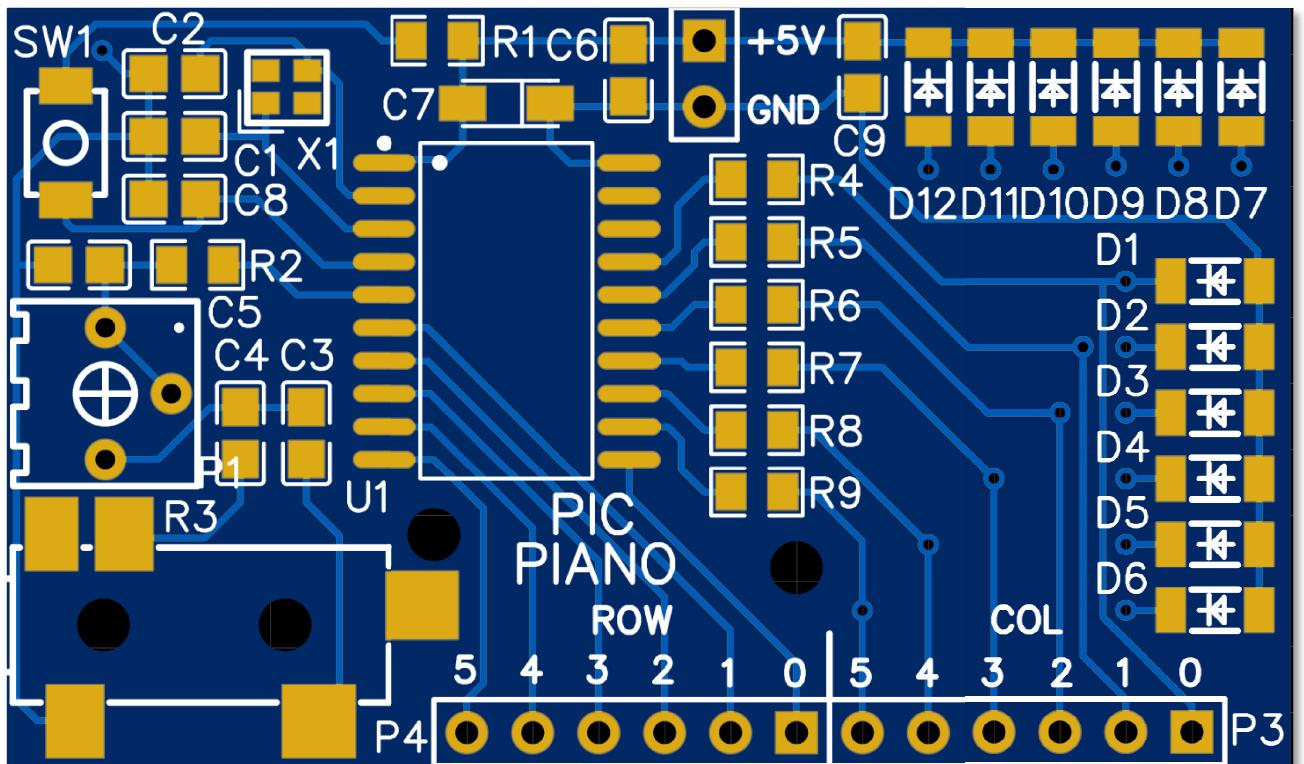


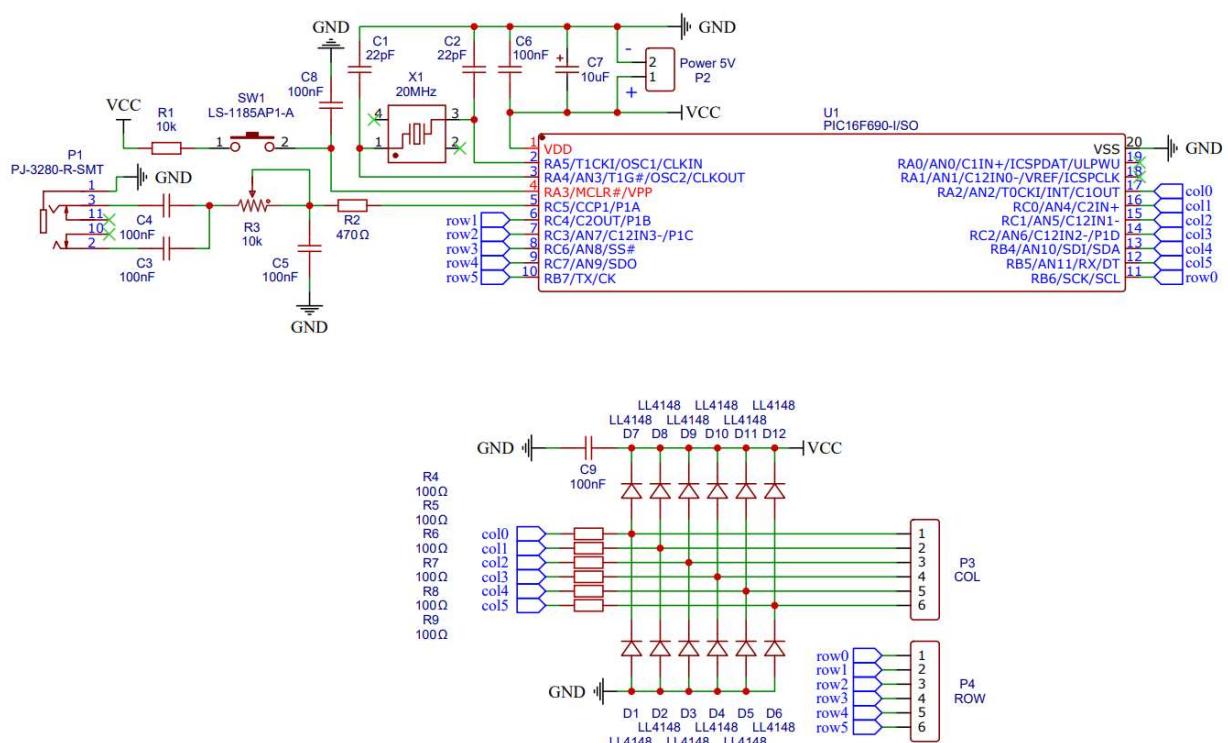
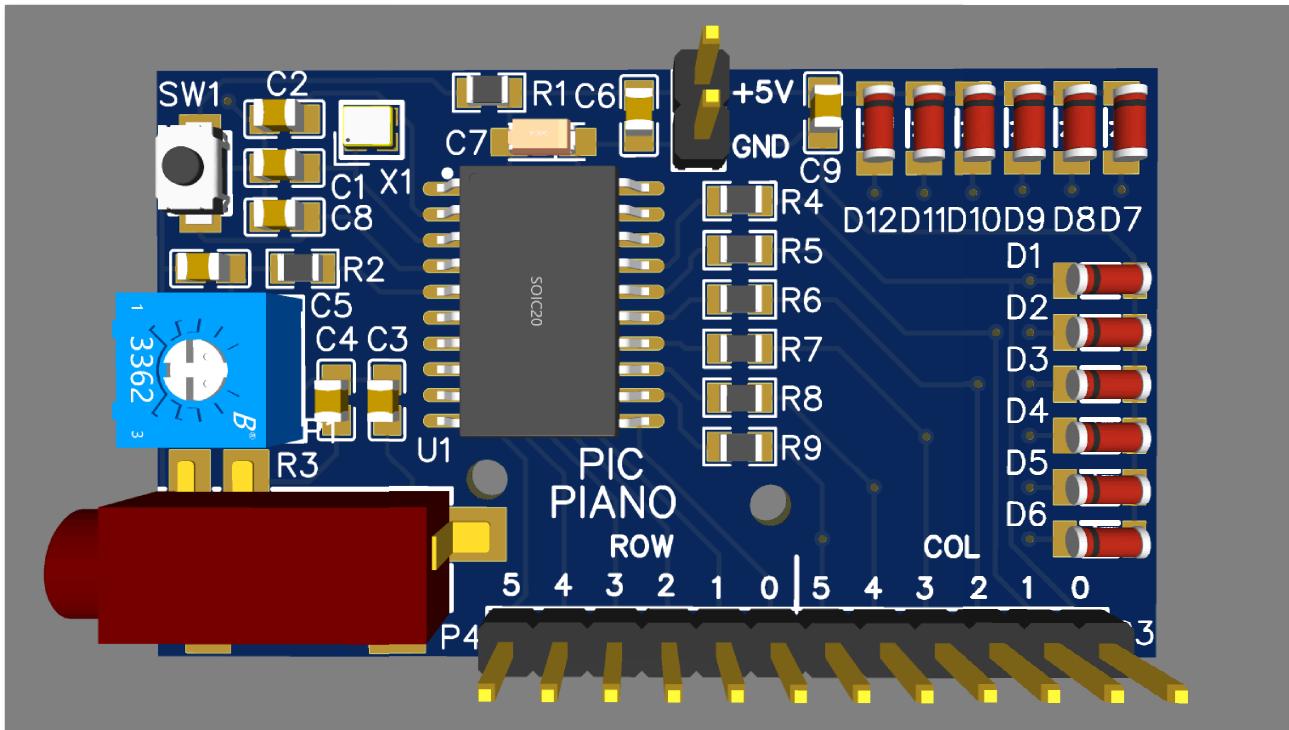
44-pin QFN



PIC16F887 Pinout

Order 1 (PIC16F690)





<https://www.radiokot.ru/forum/viewtopic.php?f=20&t=30312>

b612 (Ср май 19, 2010 11:03:49)

Уважаемые,

Кто собирал пианино <http://www.pickit2.ru/doku.php/проекты:пианино> ?

Я сделал на 690-ом пике.

Проблема: Не работает нижняя часть нижней октавы.

Посмотрел осциллографом, нет сигнала на 11 ножке (RB6 aka ROW0)

изза этого эти ноты постоянно гудят. Остальные работают нормально.

PIC исправный, проверено.

PIC16F690 FUSE “PIC PIANO”

Device Configuration Words may be edited here at the bit level. Refer to device datasheet for specific configuration bit functions.																			
	Name	Address	Value	Bit Edit															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CONFIG	2007	00C2	-	-	-	-	0	0	0	0	1	1	0	0	0	0	1	0

REGISTER 14-1: CONFIG – CONFIGURATION WORD (ADDRESS: 2007h)

Reserved	Reserved	FCMEN	IESO	BOREN1 ⁽¹⁾	BOREN0 ⁽¹⁾	<u>CPD</u> ⁽²⁾	<u>CP</u> ⁽³⁾	MCLRE ⁽⁴⁾	PWRTE	WDTE	FOSC2	FOSC1	FOSC0			
bit 13																bit 0

- bit 13-12 **Reserved:** Reserved bits. Do Not Use.
- bit 11 **FCMEN:** Fail-Safe Clock Monitor Enabled bit
1 = Fail-Safe Clock Monitor is enabled
0 = Fail-Safe Clock Monitor is disabled
- bit 10 **IESO:** Internal External Switchover bit
1 = Internal External Switchover mode is enabled
0 = Internal External Switchover mode is disabled
- bit 9-8 **BOREN<1:0>:** Brown-out Reset Selection bits⁽¹⁾
11 = BOR enabled
10 = BOR enabled during operation and disabled in Sleep
01 = BOR controlled by SBOREN bit (PCON<4>)
00 = BOR disabled
- bit 7 **CPD:** Data Code Protection bit⁽²⁾
1 = Data memory code protection is disabled
0 = Data memory code protection is enabled
- bit 6 **CP:** Code Protection bit⁽³⁾
1 = Program memory code protection is disabled
0 = Program memory code protection is enabled
- bit 5 **MCLRE:** RA3/MCLR/VPP pin function select bit⁽⁴⁾
1 = RA3/MCLR/VPP pin function is MCLR
0 = RA3/MCLR/VPP pin function is digital input, MCLR internally tied to VDD
- bit 4 **PWRTE:** Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled
- bit 3 **WDTE:** Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled and can be enabled by SWDTEN bit (WDTCON<0>)
- bit 2-0 **FOSC<2:0>:** Oscillator Selection bits
111 = RC oscillator: CLKOUT function on RA4/AN3/T1G/OSC2/CLKOUT pin, RC on RA5/T1CKI/OSC1/CLKIN
110 = RCIO oscillator: I/O function on RA4/AN3/T1G/OSC2/CLKOUT pin, RC on RA5/T1CKI/OSC1/CLKIN
101 = INTOSC oscillator: CLKOUT function on RA4/AN3/T1G/OSC2/CLKOUT pin, I/O function on RA5/T1CKI/OSC1/CLKIN
100 = INTOSCIO oscillator: I/O function on RA4/AN3/T1G/OSC2/CLKOUT pin, I/O function on RA5/T1CKI/OSC1/CLKIN
011 = EC: I/O function on RA4/AN3/T1G/OSC2/CLKOUT pin, CLKIN on RA5/T1CKI/OSC1/CLKIN
010 = HS oscillator: High-speed crystal/resonator on RA4/AN3/T1G/OSC2/CLKOUT and RA5/T1CKI/OSC1/CLKIN
001 = XT oscillator: Crystal/resonator on RA4/AN3/T1G/OSC2/CLKOUT and RA5/T1CKI/OSC1/CLKIN
000 = LP oscillator: Low-power crystal on RA4/AN3/T1G/OSC2/CLKOUT and RA5/T1CKI/OSC1/CLKIN

- Note**
- 1: Enabling Brown-out Reset does not automatically enable Power-up Timer.
 - 2: The entire data EEPROM will be erased when the code-protect is turned off.
 - 3: The entire program memory will be erased when non code-protect is turned off.
 - 4: When MCLR is asserted in INTOSC or RC mode, the internal clock oscillator is disabled.

PIC16F88 FUSE “PIC PIANO”

Configuration Word Editor																			
Device Configuration Words may be edited here at the bit level. Refer to device datasheet for specific configuration bit functions.																			
	Name	Address	Value	Bit Edit															
				15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	CONFIG1	2007	2F02	-	-	1	0	1	1	1	1	0	0	0	0	0	1	0	
	CONFIG2	2008	0000	-	-	-	-	-	-	-	-	-	-	-	-	-	0	0	

Unimplemented bits are displayed in the Value column as selected in menu Tools > Display Unimplemented Config Bits

Save **Cancel**

Order 2 (PIC16F88)

Product Details

Price



	PCB Prototype	\$2.00
	Order #: Y24-2179435A	5pcs
	Build Time: 3-4 days	
Product Details		

Неділя, 19 травня 2024 -249.80
Інтернет-магазини -249.80
JLCPCB.COM, 8613738561292
04:51

