Micromint Chips

PicStic™

Subminiature Control Computer

FEATURES

• Small size—less than 0.85 in.^{2} (0.60″ × 1.40″ × 0.31″)
• Wide power-supply voltage range: 4.5–18 V
• Low power consumption—less than 2-mA operating current, 1-µA typical standby at +5 V (400 µA when using onboard regulator)
• EEPROM-programmable 4-MHz PIC16F84 processor: 1024 × 14 EEPROM program memory, 64 × 8 EEPROM data memory
• Battery-backable real-time clock/calendar (PS2)
• 2-channel, 12-bit 0–5-V ADC (PS3)
  Max. Sample Rate: ~5200 samples/second
  (Sampled using CALL AD0 in PicBASIC)
• Pin-compatible with Parallax BASIC Stamp I
• 8 bidirectional, bit-programmable high-current I/O iines: 25-mA sink per pin, 20-mA source per pin
• 2 additional, bit-programmable I/O lines and 4 interrupt sources: assembly language addressable only
• Thermally protected onboard regulator supports high-current externally connected devices (see chart)
• Data EEPROM typically 1,000,000 erase/write cycles
• Programmable in BASIC, C, and assembly language
• Compiler-provided serial-communications routines: 300--9600 bps, depending on compiler
• Usable for low-cost data acquisition and control
• Industrial temperature range available (minimum quantity applies)

DESCRIPTION

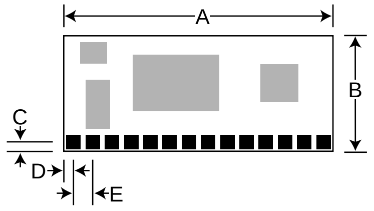
PicStic is a low-cost, industrially oriented controller on a 0.85-square-inch SIP (PicStics are manufactured both with and without pins). Including options, PicStic incorporates digital inputs and outputs, analog inputs, real-time monitoring, power-input regulation, and serial communication (provided through software) in a single module. PicStic can be used independently or networked together.

PicStic offers both compatibility and improved performance and comes in three versions: PS1, PS2, and PS3. The PS1, PS2, and PS3 are all pin-compatible with the Parallax BASIC Stamp I.

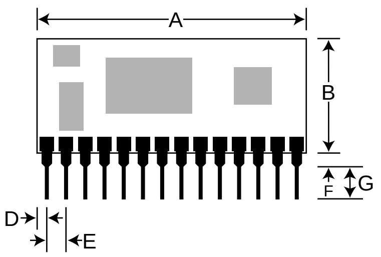
The PicStic1 is a straight one-for-one programmable replacement for the BS1.

The PicStic2 has all the features of the PicStic1 plus a real-time clock/calendar that keeps time in terms of the year, month, day of the month, day of the week, hour, minute, and seconds. The clock always runs while the PicStic2 is powered. An optional 3-V lithium battery maintains the clock when power is off. The battery, which is approximately 0.6″ in diameter, can be mounted on the front or back of the PicStic2.

The PicStic3 has all the features of the PicStic1 plus a 2-channel, 12-bit ADC. The compilers contain library routines for reading the ADC and real-time clock. The ADC can be sampled at 5200 samples/sec.



Actual Size



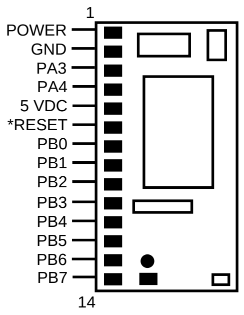
Dim	Inches	Millimeters
A	1.400	35.560
B	0.590	14.986
C	0.050	1.270
D	0.050	1.270
E	0.090	2.286
F	0.050	1.270
G	0.320	8.128

While PicStic is BASIC Stamp I pin-compatible, you can program it in more than BASIC. Unlike the customized hardware of the BASIC Stamp, PicStic uses a generic reprogrammable PIC16F84 processor and a customized compiler. Additional features and improvements typically involve recompiling your program, not buying new hardware.

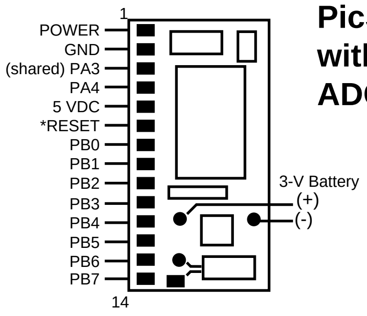
The single major advantage of PicStic is that you get an **additional** one (PS2) or two (PS1 and PS3) **I/O lines and access to the four PIC16F84 interrupt sources** when using assembly language (or an assembly-language call appended to PICBASIC or C). With an ADC, interrupts, and 10 I/O lines, the PS3 provides a powerful little controller for cost-conscious applications.

Using the PICBASIC compiler, PicStic is 100% Basic Stamp I compatible. As a bonus, it's also at least **15 times faster** for the same crystal speed. It has twice the typically available program space (see our separate comparison sheet (in "Freqent Questions About PicStic") for more details).

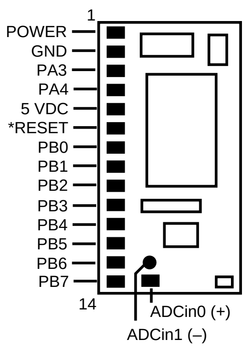
PicStic1



PicStic2 with clock

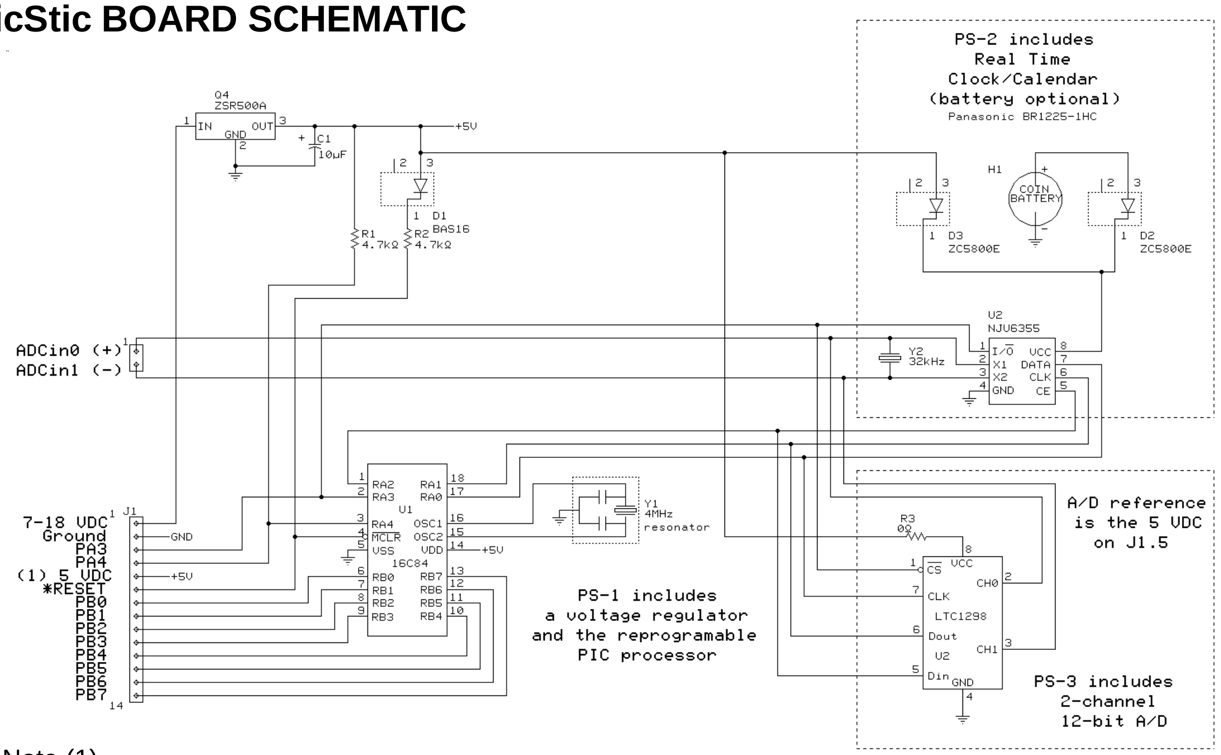


PicStic3 with ADC



Top view

PicStic BOARD SCHEMATIC


PS-2 includes Real Time Clock/Calendar (battery optional)
Panasonic BR1225-1HC
Q4 ZSR500A
IN OUT GND
C1 10µF
+5V
D1 BAS16
R1 4.7kΩ
R2 4.7kΩ
H1 COIN BATTERY
D3 ZC5800E
D2 ZC5800E
U2 NJU6355
Y2 32kHz
I/O X1 X2 GND VCC DATA CLK CE
ADCin0 (+)
ADCin1 (-)
7-18 VDC
Ground
PA3
PA4
(1) 5 VDC
*RESET
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
J1
GND
+5V
U1 16C84
RA2 RA3 RA4 MCLR VSS RB0 RB1 RB2 RB3
RA1 RA0 OSC1 OSC2 VDD RB7 RB6 RB5 RB4
Y1 4MHz resonator
R3 0Ω
A/D reference is the 5 VDC on J1.5
PS-1 includes a voltage regulator and the reprogramable PIC processor
U2 LTC1298
CS VCC CLK Dout Din GND CH0 CH1
PS-3 includes 2-channel 12-bit A/D

Note (1)
If an unregulated voltage is connected to J1.1, J1.5 is a regulated 5-VDC output (~100 mA).
If J1.1 is unconnected, the user must connect a regulated 5 VDC to J1.5.

Absolute Maximum Ratings

Supply Voltage (Pin 1)	+18.0V	Lead Temperature	260°C
Digital Input Voltage	0V to +5.5V	Operating Temperature	0°C to +70°V
Analog Input Voltage	-.05V to +5.5V		(-40°C to +85°C available by special order)
Storage Temperature	-25°C to +100°C		

Exceeding these values may result in permanent damage to the device.

Pin Description

Pin	Signal	Description
AIN0, AIN1		on a PS3, the 2 channel, 12-bit ADC inputs are directly above Pins 14-13, respectively.
1	POWER	Unregulated power in. Accepts 7-18 VDC input. May be left un-connected if 5V is applied to pin 5.
2	GND	System ground. On PS3, it is the combined digital and analog ground.
3	PA3	Bi-directional I/O line available on PS1 and PS3. I/O line shared with RTC on PS2. (See section 4.2 for PA3 and PA4 programming details).
4	PA4	Bi-directional I/O line. (See section 4.2 for PA3 and PA4 programming details).
5	+5V	5V input/output. If an unregulated voltage is applied to pin 1, then this pin is the +5V output. The current Drawn from this pin should be limited to reduce the PicStic regulator's power dissipation. If no Voltage is applied to pin 1, then a regulated voltage (4.5V to 5.5V) should be applied to this pin for PS1/3. PS2 requires a minimum of 4.9V. For the PS3, the voltage at this pin is the V_{ref} of the ADC.
6	*RESET	Reset input. A logic low applied to this pin resets all I/O functions.
7-14	PB0-PB7	General-purpose I/O. Each pin can sink 25 mA and source 20 mA. Total current should not exceed 50mAsinking and 40mA sourcing. PB0 is a direct interrupt input for assembly-language programs.

A/D Converter Characteristics

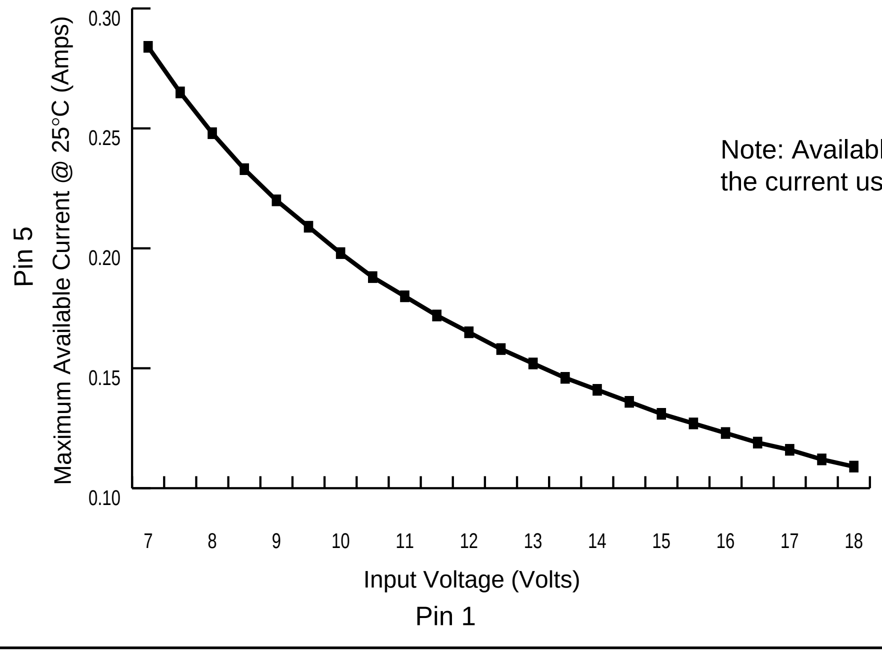
Characteristics	Min	Typical	Max	Units	Condition
Analog Input Impedance		250		KΩ	
Analog Input Range		-0.5 to V_{ref} +0.5		V	
Linearity Error		±3/4		Bits	
Offset and Gain Error		±2		Bits	
Resolution	12			Bits	
Sample Rate			~5200		Using PicBASIC
Voltage Reference	4.5	5.0	5.5	V	V_{ref} is V_{cc}

DC Electrical Characteristics

Operating Temperature T_a= 0°C to 70°C
Operating Voltage V+ = 9 V, GND = 0 V

Characteristic	Min	Typ	Max	Units	Condition
Supply Voltage (pin 1)	7	9	18	V	
Supply Current (I_{CC})		<2	4.5	mA	V_{CC} = 5.5 V
Memory:					
Data EEPROM Endurance		1,000,000			E/W cycles
Program EEPROM Endurance		1000			Programming E/W cycles
Digital Inputs:					
Input Low Voltage (V_{il})	0		0.9	V	
Input High Voltage (V_{ih})	1.9		5.5	V	
Output Low Voltage (V_{ol})		0.45	0.6	V	I_{ol} = 8.5 mA
Output High Voltage (V_{oh})	4.3			V	I_{ol} = –3.0 mA

Maximum available current draw from pin 5 with input power applied to pin 1.



Note: Available current does not include the current used by any I/O pins.

Example

Input voltage = 10 V + 2 I/O pins sinking 15 mA each

Current available = Max. Current (from chart) – Current used by I/O pins
I = 195 mA – 30 mA
= 165 mA

Typically, you shouldn't use more than ⅔ of the maximum available current. Therefore,

I_{MAX} = 165 × ⅔
= ~100 mA

1.0 PicStic OVERVIEW

PicStic is a low-cost, industrially oriented controller on a 0.85-square-inch SIP. At its core is a Microchip Technology PIC16F84 RISC processor which includes onchip RAM, EEPROM, and other features. Optional PicStic peripherals include an NJU6355 real-time clock and an LTC1298 A/D converter.

1.1 PIC16F84 PROCESSOR

The PIC16F84 is a high-performance, low-cost, CMOS, fully-static 8-bit microcontroller with 1KB × 14 EEPROM program memory and 64 bytes of EEPROM data memory. Its high performance is due to instructions that are all single word (14-bit wide), which execute in single cycle (1 µs at 4-MHz clock), except for program-branches which take two cycles (800 ns).

The PIC16F84 has 4 interrupt sources and an 8-level hardware stack. The peripherals include an 8-bit timer/counter with an 8-bit prescaler (effectively a 16-bit timer) and 13 bidirectional I/O pins. The high-current drive (25 mA maximum sink, 20 mA maximum source) of the I/O pins helps reduce external drivers and therefore, system cost.

1.2 NJU6355 REAL-TIME CLOCK

The NJU6355 series is a serial I/O real-time clock. It contains a quartz crystal oscillator, a shift register, a voltage regulator, a voltage detector, and an interface controller. The NJU6355 requires only four microprocessor bits for data transfer. The microprocessor receives data anytime it is required.

The operating voltage is 2.0–5.5 V. Consequently, the NJU6355 counts accurate time data, even during backup. (The clock can only be read or written to when VCC is >= 4.75 V.) Since current consumption during backup is less than 3 µA, backup can be done over a long period of time with a small battery.

1.3 LTC1298 A/D CONVERTER

The LTC1298 is a micropower, 12-bit, successive approximation sampling A/D converter. It nominally consumes 350 µA of supply current when sampling at 11.1 kHz. Supply current drops linearly as the sample rate is reduced. The ADC automatically powers down when not performing conversions, drawing only leakage current.

The LTC1298 contains a 12-bit, switched-capacitor ADC, a sample-and-hold, and a serial port. It has a 2-channel input multiplexer and can convert either channel with respect to ground or the difference between the two. The reference input is tied to the supply pin.

2.0 PicStic SOFTWARE

When it comes from the factory, the PicStic has no software on the board itself. Code is developed using cross-development tools running on a desktop PC and is programmed into the PicStic for execution. There are several development environments from which to choose.

2.1 ASSEMBLY

Any cross-assembler capable of creating code for the PIC16F84 processor can be used to write assembly language programs for the PicStic. Microchip's assembler is available at no cost from their BBS and Web site. More information about the PIC instruction set and how to connect to the Microchip BBS may be found in the *Microchip Data Book*. Their Web site may be contacted at <http://www.mchip.com/microchip/>.

Other cross-assemblers are available including one from Parallax that enhances the PIC instruction set with one more familiar to 8051 programmers. Contact Parallax for more information.

A cross-assembler also comes with the Micromint PICBASIC package.

2.2 PICBASIC

Micromint's PICBASIC compiler allows the use of BASIC Stamp-compatible programs on the PicStic, but with much higher execution speed. PICBASIC also provides the capability to include custom assembly language routines for time-critical tasks.

The PICBASIC compiler was originally designed to be Basic Stamp 1 compatible. However because it is a compiler, it can have additional commands and enhancements without changing the hardware. Scheduled enhancements include PEEK, POKE, 9600 bps communications, and DTMF transmissions. The following command list should, therefore, be presumed to the minimum command list.

Please contact Micromint for more information about PICBASIC

PICSTIC PICBASIC COMMANDS AND ASSEMBLY ROUTINE CALLS

Flow Control

IF…THEN	GOTO LABEL if condition is true.
BRANCH	Computed GOTO (equivalent to ON…GOTO).
GOTO	Jump to the code at LABEL.
GOSUB	Call the subroutine at LABEL (nesting up to 4 levels, numbers of subroutines only limited by the available code space).
RETURN	Return from subroutine.

Looping

FOR…NEXT	Repeat block of code x number of times.

Assignment

LET	Perform math and assign the result to a variable.
LOOKUP	Retrieve a value from a table based on an indexed stored variable.

Digital I/O

INPUT	Make a pin an input.
OUTPUT	Make a pin an output.
LOW	Make a pin an output and set it low.
HIGH	Make a pin an output and set it high.
TOGGLE	Make a pin an output and toggle its state.
REVERSE	If a pin is an output, make it an input. If a pin is an input make it an output.
PULSOUT	Generate a pulse (10µs resolution).
PULSIN	Measure the width of an input pulse (10µs resolution).
BUTTON	Debounce a switch and perform autorepeat if held. Can also branch the program when a specific state occurs.

Analog I/O

PWM	Output a PWM pulse train.
POT	Read a potentiometer.

Asynchronous Serial I/O

SERIN	Asynchronous serial input (300, 600, 1200, and 2400 bps; N81).
SEROUT	Asynchronous serial output (300, 600, 1200, and 2400 bps; N81).

Sound

SOUND	Generate tones or white noise.

EEPROM Access

EEPROM	Define the initial contents of the EEPROM.
READ	Read a byte from the EEPROM into a variable.
WRITE	Write a byte into the EEPROM.

Time Delay

PAUSE	Pause the execution of the program (1-ms, 65536 ms maximum).

Power Control

NAP	Power down the processor for a short period of time (1-ms resolution, 65536 ms maximum).
SLEEP	Power down the processor for a short period of time (1-s resolution, 65536 ms maximum)
END	Stop execution of the program and enter low-power mode.

Callable clock Routines for PicStic 2

ClockSet	Set the onboard clock/calendar
ClockGet	Read the onboard clock/calendar

Callable ADC Routines for PicStic 3

AD0	Read analog input 0
AD1	Read analog input 1
AD	Read the difference between input 0 and 1 (0=+ and 1=-)

I/O Co-processor for PicStic 4

PASS	Assembly routine to communicate to the I/O Co-processor

2.3 C

The integrated C development environment gives developers the capability to quickly produce efficient code from an easily maintainable high-level language. The compiler includes built-in functions to access the PIC hardware such as **READ_ADC** to read a value from the ADC. Discrete I/O is handled by describing the port characteristics in a PRAGMA. Functions such as INPUT and OUTPUT_HIGH properly maintain the tristate registers. Variables including structures may be directly mapped to memory such as I/O ports to best represent the hardware structure in C. The microcontroller clock speed may be specified in a PRAGMA to permit built-in functions to delay for a given number of micro- or milliseconds. Serial I/O functions allow standard functions such as GETC and PRINTF to be used for RS-232-like I/O. The hardware serial transceiver is used for applicable parts when possible. For all other cases, a software serial transceiver is generated by the compiler. The standard C operators and the special built-in functions are optimized to produce very efficient code for the bit and I/O functions normally required for these microcontrollers.

Functions may be implemented inline or separate. Function parameters are passed in reusable registers. Inline functions with reference parameters are implemented efficiently with no memory overhead.

During the linking process, the program structure including the call tree is analyzed. Functions that call one another frequently are grouped together in the same page. Calls across pages are handled automatically by the tool transparent to the user. RAM is allocated efficiently by using the call tree to determine how locations can be reused·

3.0 PROGRAMMING THE PICSTIC

The PicStic uses Microchip Technology's PIC16F84 EEPROM microcontroller which can be reprogrammed hundreds of times. These programs can be created by a number of sources. Assembly-language programs can be written in Microchip's native instruction set or Parallax's 8051-like instruction set. Compilers such as C, PICBASIC, or fuzzy logic can be used as well. The single requirement of any programming environment is that the end result is a PicStic-compatible Intel hex file.

Programming a PicStic is done serially, involving only five signal connections. The five signals are power (+5 V), ground (0 V), and *MCLR (which must be pulled to +12 V), and port pins RB7 (serial data) and RB6 (serial clock).

The PicStic Development package includes the programmer board and necessary software to program the PicStic. (Note: the Parallax BASIC Stamp programmer cannot be used for programming PicStic.) The PicStic grammer cannot be used for programming PicStic.) The PicStic can be programmed with most other PIC16F84 programmers by making a simple five-wire DIP-to-SIP adapter. The five signals from your programmer's 16F84 DIP programming socket are wired to a 14-pin PicStic SIP socket as shown in the table.

Most assemblers and compilers enable you to take advantage of a variety of 16F84 configuration options. These include code protection, powerup timer, watchdog timer, oscillator designation (PicStic uses the XT), and user programming of the 64 bytes of additional EEPROM data memory. Be aware that you may have to set these conditions on the compiler for proper PicStic programming

For a more involved description of the 16F84 programming algorithm and technology, refer to the *Microchip Data Book*.

16F84 Programming DIP socket pin#	Name	Function	PicStic's 14-pin SIP socket pin#
4	*MCLR	reset	6
5	V_{SS}	GND	2
12	RB6	CLK	13
13	RB7	DATA	14
14	V_{DD}	5 V	5

4.0 PICSTIC HARDWARE

We provide sample code for much of PicStic's hardware features. It is beyond the scope of this document to provide full details about each device. However, we have tried to include enough information to allow the programmer to develop code in other languages than that used for the sample code.

4.1 SERIAL CONNECTIONS

PicStics have no specific pins for serial I/O. The compilers allow the user to designate the physical pin locations of serin and serout. The circuits in Figure 4.1 should be used to interface the PicStic to RS-232C serial port.

While we recommend the use of a proper level-shifting serial interface, PicStic also works in other BASIC Stamp serial configurations.

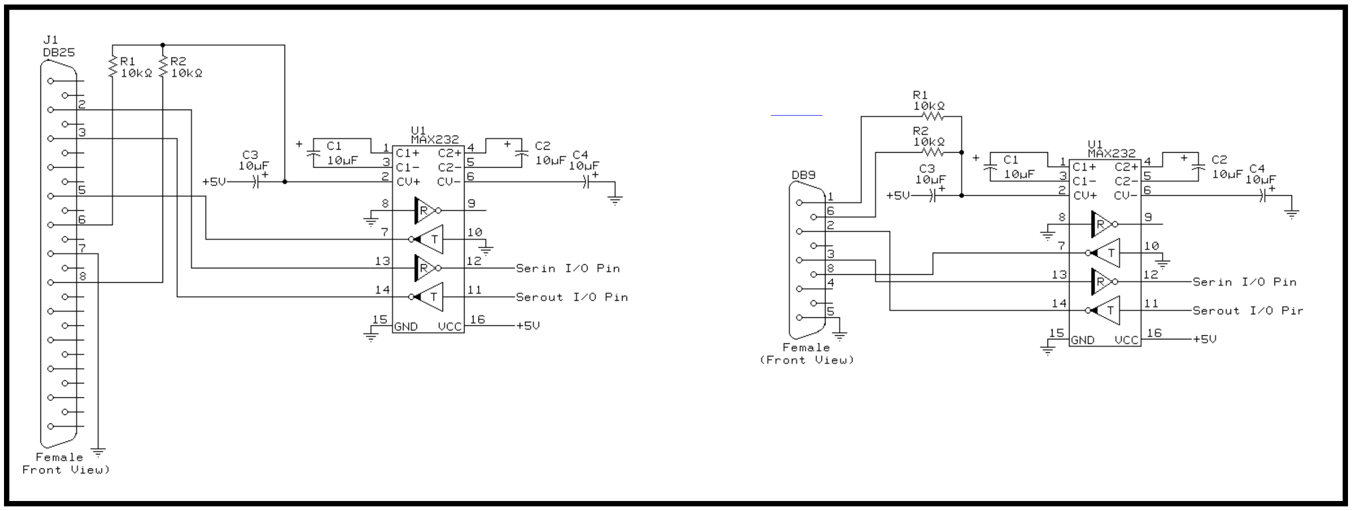


Figure 4.1: PicStic serial connections.

4.2 PA3 AND PA4

The PA3 and PA4 I/O lines are directly accessible from assembly language and C. However, additional assembly-language routines are necessary to access these lines from PICBASIC.

The following program demonstrates how to use the extra PicStic I/O bits. PicStic 1 and PicStic 3 have PA3 available, while all three PicStics have PA4 available.

PB0 controls the mode. When PB0=0, PA4 becomes an input and PA3 becomes an output. In both cases, the output is set to the opposite of the input. PB7 outputs a 1200-bps serial message that indicates what is happening.

Note that if you tie PA3 to PA4 with a 1-kΩ resistor, the output self toggles the input. The toggle speed is relatively slow due to the 1200-bps serial output routine.

```
symbol  mode = pin0
input   0

start:  if mode=0 then m0

m1:  call Tst_PA3
     if bit0=1 then m11
m10: serout 7,T1200,("PA3=0, so setting PA4=1",
       13,10)
     call Set_PA4
     goto start
m11: serout 7,T1200,("PA3=1, so setting PA4=0",
       13,10)
     call Clr_PA4
     goto start

m0:  call Tst_PA4
     if bit1=1 then m01
m00: serout 7,T1200,("PA4=0, so setting PA3=1",
       13,10)
     call Set_PA3
     goto start
m01: serout 7,T1200,("PA4=1, so setting PA3=0",
       13,10)
     call Clr_PA3
     goto start
end

asm
;
```

```
;***********************************************
; Set PA3
;
_Set_PA3 bsf  PORTA,3 ; Set PA3
         bsf  STATUS,5; Change to Bank1
         bcf  TRISA,3 ; Direction of PA3 is output
         bcf  STATUS,5; Change to Bank0
         goto done    ; Return to PICBASIC
;
;***********************************************
; Clear PA3
;
_Clr_PA3 bcf  PORTA,3 ; Clear PA3
         bsf  STATUS,5; Change to Bank1
         bcf  TRISA,3 ; Direction of PA3 is output
         bcf  STATUS,5; Change to Bank0
         goto done    ; Return to PICBASIC
;
;***********************************************
; Test PA3
;
_Tst_PA3 bsf  STATUS,5 ; Change to Bank1
         BSF  TRISA,3  ; Direction of PA3 is input
         bcf  STATUS,5 ; Change to Bank0
         bcf  _B0,0    ; Clear bit 0 of B0
         btfsc PORTA,3 ; Test PA3 and skip if low
         bsf  _B0,0    ; PA3 high, so set B0 bit 0
         goto  done    ; Return to PICBASIC
;
;***********************************************
; Set PA4
;
_Set_PA4 bsf   PORTA,4 ; Set PA4
         bsf   STATUS,5; Change to Bank1
         bcf   TRISA,4 ; Direction of PA4 is
output
         bcf   STATUS,5; Change to Bank0
         goto  done    ; Return to PICBASIC
;
;***********************************************
; Clear PA4
;
_Clr_PA4 bcf   PORTA,4 ; Clear PA4
         bsf   STATUS,5; Change to Bank1
         bcf   TRISA,4 ; Direction of PA4 is
output
         bcf   STATUS,5; Change to Bank0
         goto  done    ; Return to PICBASIC
;
;***********************************************
; Test PA4
;
_Tst_PA4 bsf   STATUS,5; Change to Bank1
         bsf   TRISA,4 ; Direction of PA4 is input
         bcf   STATUS,5; Change to Bank0
         bcf   _B0,1   ; Clear bit 1 of B0
         btfsc PORTA,4 ; Test PA4 and skip if low
         bsf   _B0,1   ; PA4 high, so set B0 bit 1
         goto  done    ; Return to PICBASIC
;
endasm
```

4.3 REAL-TIME CLOCK

PicStic 2 includes a JRC NJU6355 serial real-time clock chip. The following sections describe how to access the part. **Note: All compilers contain program routines for directly reading and setting the clock. The following clock information is primarily for those people programming in assembly language.**

4.3.1 TIMER DATA STRUCTURE

The NJU6355 uses BCD code consisting of four bits per digit (see Figure 4.3.1). The calendar function includes the last date of each month. The leap-year calculation executes automatically. The unused bit for the timer is "0".

4.3.2 TIMER DATA HANDLING

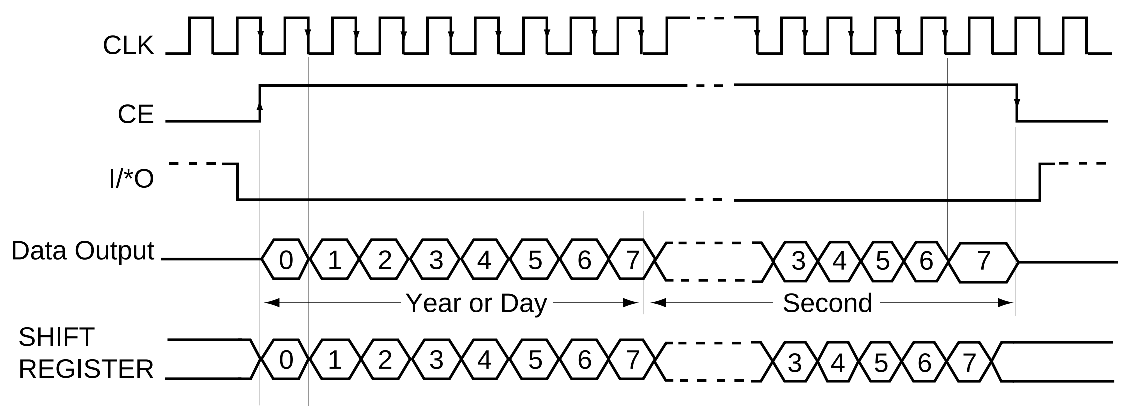
When the I/O terminal is "L" and the CE terminal is "H", the timer data can be read out. The output is LSB first, and the output data strings (depending on the version) are shown below. The timer data is transferred from the timer/counter to the shift register at the rising edge of the chip enable on the CE terminal. The output of the LSB of the timer data is from the DATA terminal. The timer data in the shift register shifts by synchronizing at the falling edge of the clock signal on the CLK terminal. Output is from the DATA terminal. If the timer data is updated in the data output, there is a 1-s difference between the timer data and the output data.

Year	Month	Date	Day	Hour	Minute	Second

The data is read out from LSB of Year, and first 52-bit is effective. If the low-voltage detector detects a low battery, (EE)x is written into each digit of the timer data and is read out. Code EE(x) warns that the data is broken (see Figure 4.3.2).

	MSB							LSB	Range
Second	0	S6	S5	S4	S3	S2	S1	S0	0–59
Minute	0	m6	m5	m4	m3	m2	m1	m0	0–59
Hour	0	0	H5	H4	H3	H2	H1	H0	0–23
Days of Week					0	W2	W1	W0	1–7
Date	0	0	D5	D4	D3	D2	D1	D0	1–31
Month	0	0	0	M4	M3	M2	M1	M0	1–12
Year	Y7	Y6	Y5	Y4	Y3	Y2	Y1	Y0	0–99

Figure 4.3.1: Timer data bit map



The timer data is transferred to the shift register at the rising edge of the CE and LSB of the timer data is output to the Data terminal. Afterward, the timer data in the shift register shifts by synchronizing at the falling edge of the CLK and then outputs to the Data terminal time-to-time.

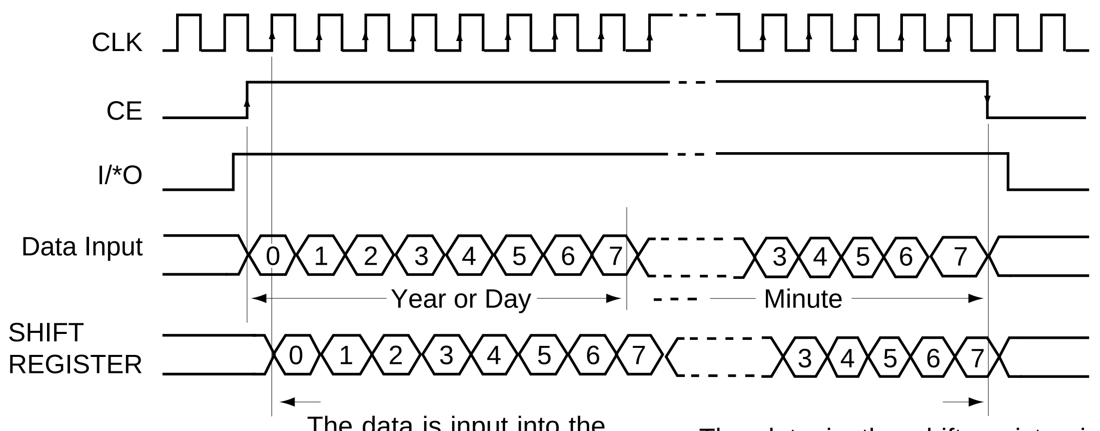
Figure 4.3.2: Read-out timing.

4.3.3 TIMER DATA WRITING

When both the I/O and CE terminals are "H", the update stops. The oscillator divider clears, and the timer data can be written to the NJU6355. The timer data is written into the shift register from the DATA terminal by synchronizing with the rising edge of the clock-signal input from the CLK terminal, and the data is transferred from the shift register to the timer/counter by synchronizing with the falling edge of the CE signal (see Figure 4.3.3). The second counter clears to "0", and the oscillator divider starts the operation. The input data strings are LSB first of each digit as shown below (the data format depends on the version).

Year	Month	Date	Day	Hour	Minute

The data is written from LSB of Year, and last 44-bit is effective.



The data is input into the shift register at the rising edge of the CLK.

The data in the shift register is transferred to the timer counter at this falling edge of the CE, then the oscillator divider starts the operation

Figure 4.3.3: Write-down timing.

4.3.6 SAMPLE CODE

The PICBASIC compiler (PBC) library contains several routines to support the real-time clock on PicStic 2. When writing PBC programs, it's best to simply use these routines when you want to access this peripheral. See the CLOCK.BAS sample program on the PBC disk for more information. If you're writing your own PIC assembly-language code, the following sample code can be used to access the clock.

```
;
;************************************************
; Real-time clock I/O (PicStic 2)
;************************************************
;
        data
;
year    ds      1       ; Year (00h-99h)
month   ds      1       ; Month (01h-12h)
day     ds      1       ; Day of Month (01h-31h)
dow     ds      1       ; Day of Week (01h-07h)
hour    ds      1       ; Hour (00h-23h)
min     ds      1       ; Minute (00h-59h)
sec     ds      1       ; Second (00h-59h)
T0      ds      1       ; Temporary storage
T1      ds      1       ; Temporary storage
;
        code
;
;================================================
;
; Write time/date to real-time clock
;
setclk
        movlw   08h     ; CE=low, IO=high
        movwf   PORTA
        movlw   10h     ; CE,IO,CLK,DATA=out
        tris    PORTA
        movlw   year    ; INDF=year
        movwf   FSR
        bsf     PORTA.2 ; Enable clock I/O
        call    CS3     ; Set year, month, and day
        movf    INDF,W  ; Set day of week
        call    CSNI
        call    CS2     ; Set hr, min (zero second)
        bcf     PORTA.2 ; Disable clock I/O
        goto    done    ; Done
;
CS3     call    CSB     ; Set next byte and advance
CS2     call    CSB     ; Set next byte and advance
;
CSB     movf    INDF,W  ; Shift out LSN
        call    CSN
        swapf   INDF,W  ; Shift out MSN
CSNI    incf    FSR     ; Bump storage pointer
;
CSN     movwf   T1      ; T1=data
        movlw   4       ; T0=bit count
        movwf   T0
:next   movb    PORTA.0,T1.0; Output next bit
        rrf     T1      ; Ready next bit
        bsf     PORTA.1 ; Clock bit out of clock
        bcf     PORTA.1
        decfsz  T0      ; Continue until byte done
        goto    :next
        return          ; Done
;
```

```
;================================================
;
; Read time/date from real-time clock
;
getclk
        movlw   00h     ; CE,IO=low
        movwf   PORTA
        movlw   11h     ; CE,IO,CLK=out, DATA=in
        tris    PORTA
        movlw   year    ; INDF=year
        movwf   FSR
        bsf     PORTA.2 ; Enable clock I/O
        call    CG3     ; Get year, month, and day
        call    CGNI    ; Get day of week
        swapf   dow
        call    CG3     ; Get hr, min, and sec
        bcf     PORTA.2 ; Disable clock I/O
        return          ; Done
;
CG3     call    CGB     ; Get next byte and advance
        call    CGB     ; Get next byte and advance
;
CGB     call    CGN     ; Shift LSN in
CGNI    call    CGN     ; Shift MSN in
        call    CGMask  ; Mask unused bits
        andwf   INDF
        incf    FSR     ; Bump storage pointer
        return          ; Done
;
CGN     movlw   4       ; T0=bit count
        movwf   T0
:next   rrf     INDF    ; Shift accumulator
        movb    INDF.7,PORTA.0; Store bit to acc.
        bsf     PORTA.1 ; Clock next bit
        bcf     PORTA.1
        decfsz  T0      ; Continue until done
        goto    :next
        return          ; Done
;
CGMask  movf    FSR,W   ; W=RTC reg (0=year, ...)
        addlw   -year
        addwf   PCL     ; W=bit mask for register
        retlw   0FFh    ; Year (8 bits)
        retlw   01Fh    ; Month (5 bits)
        retlw   03Fh    ; Day of month (6 bits)
        retlw   070h    ; Day of week (3 bits)
        retlw   03Fh    ; Hour (6 bits)
        retlw   07Fh    ; Minute (7 bits)
        retlw   07Fh    ; Second (7 bits)
```

4.4 A/D CONVERTER

PicStic 3 includes a Linear Technology LTC1298 serial A/D converter chip. The following sections describe how to access the part. **Note: All compilers contain program routines for directly reading the ADC. The following ADC information is primarily for those people programming in assembly language.**

4.4.1 SERIAL INTERFACE

The two-channel LTC1298 communicates with microprocessors and other external circuitry via a synchronous, half-duplex, four-wire serial interface.

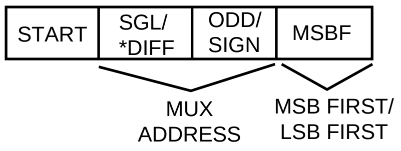
4.4.2 DATA TRANSFER

The CLK (RA0) synchronizes the data transfer with each bit being transmitted on the falling CLK edge and captured on the rising CLK edge in both transmitting and receiving systems (see figure 4.4.2). The LTC1298 first receives input data and then transmits back the A/D conversion result (half duplex).

Data transfer is initiated by a falling chip select (*CS) signal. After *CS falls, the LT1298 looks for a start bit. After the start bit is received, the three-bit input word shifts into the D_{IN} input which configures the LTC1298 and starts the conversion. After one null bit, the result of the conversion is output on the D_{OUT} line. At the end of the data exchange, *CS should be brought high. This resets the LTC1298 in preparation for the next data exchange.

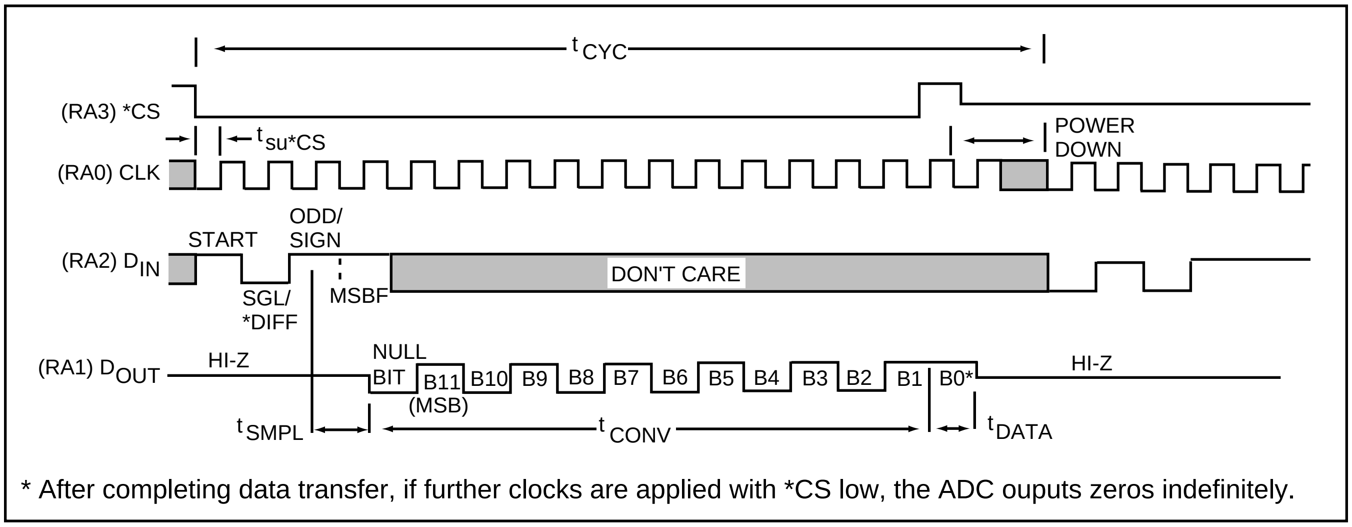
4.4.3 INPUT DATA WORD

The LTC1298requires no D_{IN} word. It is permanently configured to have a single differential input. The conversion result appears on the D_{OUT} line. The data format is MSB first, followed by the LSB sequence. This provides easy interface to MSB or LSB first serial ports. For MSB first data, the *CS signal can be taken high after B0 (see figure x). The LTC1298 clocks data in to the D_{IN} input on the rising edge of the clock. The input data words are defined as follows.



4.4.4 START BIT

The first "logical one" clocked into the D_{IN} input after *CS goes low is the start bit. The start bit initiates the data transfer. The LTC1298 ignores all leading zeros which precede this logical one. After the start bit is received, the remaining bits of the input word will be clocked in. Further inputs on the D_{IN} pin are ignored until the next CS cycle.



* After completing data transfer, if further clocks are applied with *CS low, the ADC ouputs zeros indefinitely.

Figure 4.4.2: MSB-first data (MSBF = 1).

4.4.5 MULTIPLEXER (MUX) ADDRESS

The bits of the input word following the start bit assign the MUX configuration for the requested conversion. For a given channel selection, the converter measures the voltage between the two channels indicated the + and – signs in the selected row of the following tables. In single-ended mode, all input channels are measured with respect to GND.

	MUX ADDRESS		CHANNEL #		GND
	SGL/*DIFF	ODD/SIGN	1	0	
SINGLE-ENDED MUX MODE	1	0	+		–
	1	1		+	–
DIFFERENTIAL MUX MODE	0	0	+	–	
	0	1	–	+	

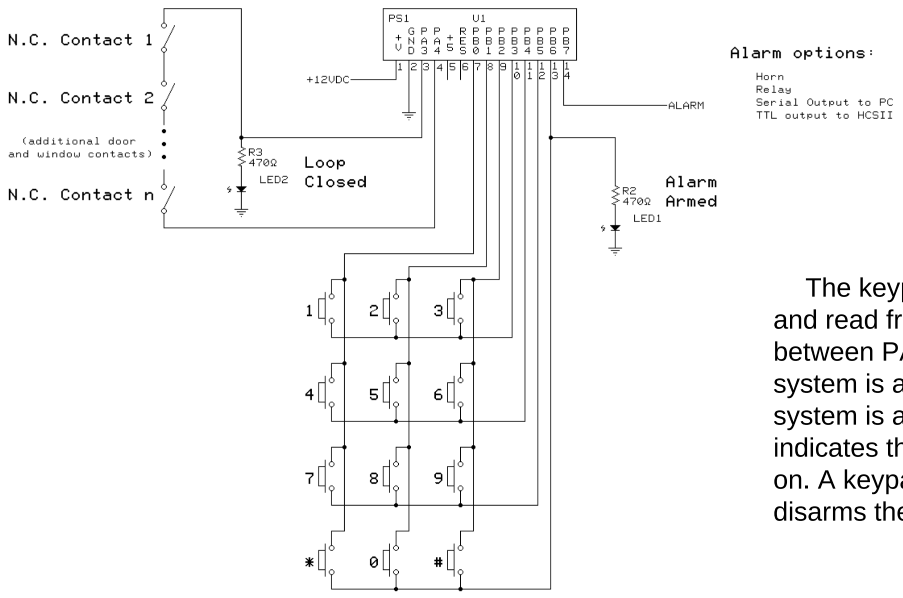
4.4.6 SAMPLE CODE

The PICBASIC compiler (PBC) library contains several routines to support the A/D converter on PicStic 3. When writing PBC programs, it's best to simply use these routines when you want to access this peripheral. See the ADC.BAS sample program on the PBC disk for more information. If you're writing your own PIC assembly language code, the following sample code can be used to access the ADC.

```
;
;**************************************************
; Analog-to-digital converter I/O (PicStic 3)
;**************************************************
;
        data
;
result  ds   2   ; A/D conv. result (low,high)
T0      ds   1   ; Temporary storage
T1      ds   1   ; Temporary storage
;
        code
;
;==================================================
;
; Read differential input (CH0=+, CH1=-)
;
addiff
        movlw   12h      ; SGL/DIFF=0, ODD/SIGN=0
        goto    adcom    ; Perform conversion
;
;==================================================
;
; Read channel 1 (single ended)
;
adch1
        movlw   1Eh      ; SGL/DIFF=1, ODD/SIGN=1
        goto    adcom    ; Perform conversion
;
;==================================================
;
; Read channel 0 (single ended)
;
adch0   movlw   1Ah      ; SGL/DIFF=1, ODD/SIGN=0
        goto    adcom    ; Perform conversion
;
```

```
;==================================================
;
; Common A/D converter code
;
;  Enter with mode in W
;  (W.3=SGL/DIFF, W.2=ODD/SIGN)
;
adcom
        movwf   T1       ; T1=mode register
        movlw   08h      ; *CS=high, CLK=low
        movwf   PORTA
        movlw   12h      ; DOUT,CLK,*CS=out, DIN=in
        tris    PORTA
        movlw   4        ; T0=mode bit counter
        movwf   T0
        bcf     PORTA.3  ; *CS=low
:mode   movb    PORTA.2,T1.4; DOUT=next mode bit
        rlf     T1       ; Ready next mode bit
        bsf     PORTA.0  ; Clock out mode bit
        bcf     PORTA.0
        decfsz  T0       ; Continue until mode done
      goto    :mode
      clrf    result     ; Clear result[15..12]
      movlw   12         ; T0=data bit count
      movwf   T0
:data bsf     PORTA.0    ; Clock in data bit
      bcf     PORTA.0
      movb    C,PORTA.1  ; C=next data bit
      rlf     result     ; Shift in data
      rlf     result+1
      decfsz  T0         ; Continue until done
      goto    :data
      bsf     PORTA.3    ; *CS=high
      return             ; Done
```

5.0 TYPICAL PICSTIC APPLICATION



The keypad is scanned by PB4–6 and read from PB0–2. The closed loop between PA3–4 is monitored when the system is armed. LED1 indicates the system is armed when on and LED2 indicates the loop is closed when it is on. A keypad security code arms and disarms the system.

Micromint Chips

Micromint, Inc.
902 Waterway Place
Longwood, FL 32750
(800) 635-3355 / (407) 262-0066
Fax: (407) 262-0069
www.micromint.com

Version 1.2 Addendum
PicStic 2k Version

The PicStic is now available with double the program space, double the data EEPROM space, and over three times the RAM space. The 2k version is available for the PicStic 1, 2, 3, and 4. Micromint's part numbers are PicStic 1-2k, PicStic 2-2k, PicStic 3-2k, PicStic 4P-2k, PicStic 4Q-2k and PicStic 4X-2k.

1.0 PIC16F628 Processor

The PicStic 2k versions use the PIC16F628 processor instead of the PIC16F84 processor. The PIC16F628 is a high-performance, low-cost, CMOS, fully-static 8-bit microcontroller with 2kB x 14 FLASH program memory, 128 bytes of EEPROM data memory, and 224 RAM bytes for data. It executes an instruction in a single cycle (1µS at 4-MHz clock), except for program branches which take two cycles.

The 2K PicStic still has an 8-level hardware stack but now has 10 interrupt sources. The peripherals include an 8-bit timer/counter with an 8-bit prescaler, an 8-bit timer with a prescaler and a postscaler, a 16-bit timer/counter consisting of 2 8-bit registers, a 10-bit capture and compare PWM and a hardware USART. The 13 bi-directional digital I/O pins can sink and source 25mA. All of the electrical specifications are the same as the Pic16F84's.

2.0 Compiling Programs with earlier versions of PicBasic and PicBasic Pro

Because the PIC16F628 is a relatively new processor, special include files were added to the PicBasic and PicBasic Pro Compilers. Without these files the PicBasic family of compilers cannot be used to program the PicStic 2K. Versions of the PicBasic Compiler earlier than version 1.42 cannot be used to program the 2K versions of the PicStic. Also, versions of the PicBasic Pro Compiler earlier than 2.21 cannot be used to program the 2K version of the PicStic.

When compiling with version 1.42 of the PicBasic compiler or versions 2.21 through 2.32 of the PicBasic Pro compiler the following line needs to be added at the beginning of the users program:

Compiling with the PicBasic Compiler
```
Symbol CMCON = $1F
Poke CMCON,7
```

Compiling with the PicBasic Pro Compiler
```
CMCON = 7
```

By adding this line to the users code, the analog comparators on PortA are set as Digital I/O. If this line is not added to the code than the processor will not be able to access any of the peripherals (I.E. the co-processor on the PicStic 4, the clock on the PicStic 2, or the A/D on the PicStic 3).

When compiling programs, the appropriate processor needs to be selected. At the command prompt the users command line should follow this format:

Compiling for the PicBasic Compiler
```
C:\PBC\PBC –p16f628 Filename.bas
```

Compiling for the PicBasic Pro Compiler
```
C:\PBP\PBP –p16f628 Filename.bas
```

3.0 Compiling for PicBasic Versions 1.43 and PicBasic Pro Version 2.33

PicBasic programs need to have -pps2k placed in the command line in order to compile a program for the PicStic 2k versions. A sample command line is **PBC -pps2k filename**. This one option will compile programs for the PicStic 1-2k, PicStic 2-2k, PicStic 3-2k, and PicStic 4-2k.

For existing PicBasic Pro programs there is a command line option for each PicStic 2k version. For the PicStic 1-2k, it is -pps1_2k. For the PicStic 2-2k, it is -pps2_2k. For the PicStic 3-2k, it is -pps3_2k. For the PicStic 4-2k, it is -pps4_2k. The following are examples of command lines for compiling programs for PicBasic Pro.

Sample PicBasic Programs	**Sample PicBasic Pro Programs**
C:\PBC\PBC -pps2k picstic1.bas	**C:\PBP\PBP -pps1_2k picstic1.bas**
C:\PBC\PBC -pps2k picstic2.bas	**C:\PBP\PBP -pps2_2k picstic2.bas**
C:\PBC\PBC -pps2k picstic3.bas	**C:\PBP\PBP -pps3_2k picstic3.bas**
C:\PBC\PBC -pps2k picstic4.bas	**C:\PBP\PBP -pps4_2k picstic4.bas**

Existing PicBasic and PicBasic Pro programs for the original PicStic will be able to be compiled for the PicStic 2k versions without altering the program. This is done by recompiling the existing PicBasic and PicBasic Pro programs with the above mentioned command line options.

4.0 Programming PicStic 2k Versions

PicStic 2k versions can not be programmed using EPIC DOS. When using EPIC Windows version 2.30 or earlier, be sure to set the device for 16F628. For EPIC Windows version 2.31 or greater be sure, to select the appropriate PicStic 2k device. When programming the PicStic 2k versions with the PicStic programmer, JP1 must be connected to force power to the PicStic. If JP1 is not connected, a programming code error will occur. The original PicStic can be programmed with JP1 connected without damaging it. **Warning: When inserting and removing a PicStic with JP1 installed the power to the PicStic programmer should be removed. Note:** Make sure your configuration is set properly. If you want to be able to reset the PicStic 2k then Master Clear Reset needs to be checked. To keep from having to check or change your configuration make sure that “Update Configuration” is checked under the Options menu.

Because the PIC16F628 has a low voltage-programming mode, an error can occur when trying to program a 2K version of the PicStic. The error comes from the PORTB.4 pin in a floating state during programming. If a noise spike great enough to generate a high on this pin occurs during programming Epic will generate a error (Programming Error at code 0000) and the device will not be programmed. This can be avoided by programming the PicStic 2Ks in a low noise environment. Also, if PORTB.4 is being used as an output and the Master Clear Reset (in the configuration) is not selected, then the device must be erased before it can be programmed. If this is not done then epic will generate an error and the device will not be programmed. If the Master Clear Reset is selected then the device will program normally.

Software Requirements for Programming a PicStic

The following software is required for programming the 1K versions of the PicStic:

Using the PicBasic Compiler	**Using the PicBasic Pro Compiler**	**Programming with Epic**
Version 1.11 or higher	Version 2.00 or higher	Any Windows version can be used. DOS versions will not work properly with any current versions of the PicStic

The following software is required for programming the 2K versions of the PicStic:

Using the PicBasic Compiler	**Using the PicBasic Pro Compiler**	**Programming with Epic**
Version 1.42 or higher Version 1.43 is recommended	Version 2.21 or higher Version 2.33 is recommended	Any Windows version 2.11 or Higher. Version 2.31 is recom-mended. DOS versions cannot be used.