PIC Software UART Routines

© 2005 by Datadog Systems   Author John Massa

UARTs

A UART (Universal Asynchronous Receive/Transmit) is a hardware or software mechanism which enables a microcontroller to receive/send serial communications, usually with RS232 hardware.

RS232

RS232 (Revised Standard 232) is a hardware standard specifying the connector pins and the voltage levels used by some serial communication devices, such as the ASR33 teletype machine first used circa 1941. This standard is still popular today, although the original 25-pin connector was replaced by IBM with a 9-pin miniature 'D' connector, which is today's 'de facto' standard. The voltage levels describe a 'MARK' (i.e.: A logic 1) and a 'SPACE' (i.e.: A logic 0). A MARK is the idle signal level and is any voltage between –3 Volts and –15 Volts. A SPACE is the high signal level and is any voltage between +3 Volts and +15 Volts.

RS232 Protocol

The RS232 protocol is a serial method of sending one single 'frame' at a time. A frame is defined as a 'start-bit', a 7-bit ASCII character, a 'parity-bit' and one or more 'stop-bits'. ASCII is the American Standard Code for Information Interchange. A frame may be sent immediately following the previous frame, or sent sometime later, as in the case of a slow typist. See the frame below containing the character 'A'. Note: The character 'A' has the same bits as the hexadecimal value h'41' and/or the binary value b'01000001':

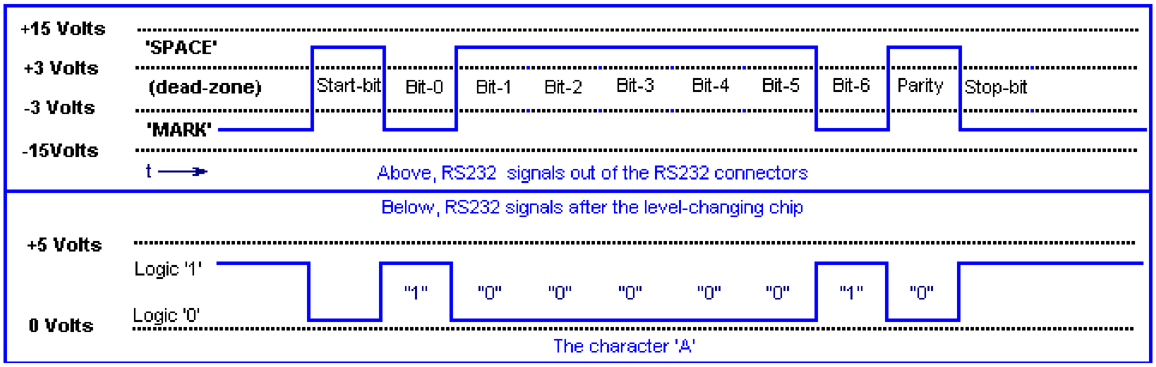


Figure 1 – The top drawing is the character 'A' as it is received from, or sent to, an RS232 connector. The bottom drawing is the same signal sent to, or received from, a level-changing inverter, such as the popular MAX232 chip. See the drawing below.

The Parity-Bit

A parity-bit is appended to the 7-bit ASCII character for error detection purposes. There are four types of parity: 1) Odd parity (O), where the parity-bit is added, or not, to make the total number of bits in the frame 'odd'. 2) Even parity (E), where the parity-bit is added, or not, to make the total number of bits even. 3) Parity stuck 'on', is where the parity bit is always a logic '1'. 4) Parity 'none' (N) is where the parity bit is always a logic '0'. This latter convention is popular today, and more modern methods of error detection replace parity, such as using a 'check-sum' character following 255 frames, or a Cyclic Redundancy Character (CRC) after a larger number of frames.

PIC Software UART Routines (Continued)

Listed below are five PIC UART software routines to use with PIC microprocessors that have no hardware UART:

1) INCH This routine inputs RS232 directly from the RS232 connector, through a 22K Ohm resistor. This method is used where economy is a higher priority then reliability. Any PIC I/O pin can be used.
2) INCH_N This routine inputs RS232 from a level-changing inverter such as the MAX232. This method is used with far (~10-Meter) distances where electrical noise may be problem. Any PIC I/O pin can be used.
3) OUTCH This routine outputs RS232 directly to the RS232 connector. This method is used where economy is a higher priority than reliability. Any PIC output pin can be used.
4) OUTCH_N This routine outputs RS232 to a level changing inverter such as the MAX232. This method is used with far (~10-Meter) distances where electrical noise may be a problem. Any PIC output pin may be used.
5) BAUD This routine sets the Baud-rate to standard speeds.

These five software routines are listed below. The I/O pins are named after the PIC10F names. You need to rename these I/O pins to match other models of PIC microcontrollers, such as to PORTA,1 and etc. Also, change the Baud-rate constant in the BAUD routine to suit the Baud-rate required by your specific application. Use a 'straight-thru' cable to connect to your PC, do not use a 'null-modem' cable. Short the RS232 connector pins 1, 7 & 8 and 6 & 4 as shown.

Remember that while transmitting or receiving a character, the software UART totally consume the PIC resources. And the only time you have to store a received character or to go get a character to transmit, is the time between the last frame's 'stop-bit' and the next frame's 'start-bit'. Work-arounds include: Using slow Baud-rates, placing a delay between frames,, or sending a group of frames as a 'packet'. With packets, you immediately store each incoming character in a separate RAM location until the entire packet is received, before attempting to process any of them.

Also remember to set the tristate register TRIS so that the PIC's RX pin is an input and that the PIC's TX pin is an output.

These routines have been successfully tested with Baud-rates as high as 38.4 Kbaud.

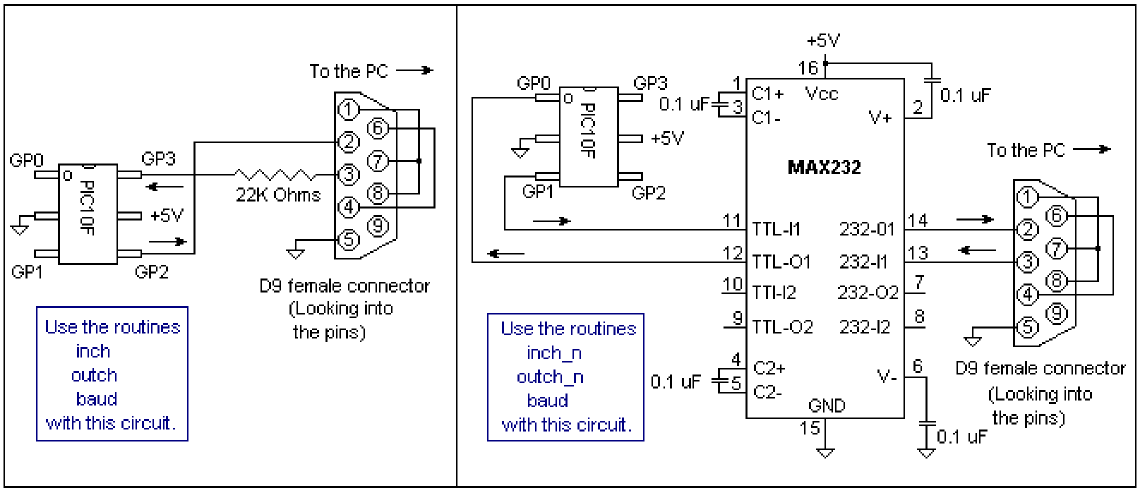
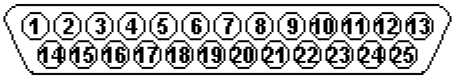


Figure 2 – Illustrating which PIC software UART routines to use with which circuits.

PIC Software UART Routines (Continued)

```
; *********************************************************************
;                            INCH ROUTINE
; THIS ROUTINE INPUTS RS232 DATA USING A 22K OHM RESISTOR, NO LEVEL-
; CHANGING INVERTER IS USED.  GPIO,3 = RX (MARK = 0, SPACE = 1).
; THIS ROUTINE USES A 8-DATA BIT PER CHARACTER PROTOCOL.
; TO RECIEVE A CHARACTER, CALL inch. THE RECEIVED CHARACTER IS PLACED
; IN THE REG 'W' AND IN THE REG 'SERBUF'.
; CHARACTER WILL ECHO IF 'retlw  0' IS REM-ED OUT.
; VARIABLES USED: REG 'TEMP' AND REG 'SERBUF' BOTH VARIABLES ARE
;                 SHARED WITH THE 'outch' ROUTINE
; ROUTINES CALLED: 'half_baud' AND 'baud' FOR THE BAUD-RATE TIMING.
; *********************************************************************
inch
        btfss   GPIO,3          ; SKIP ON START BIT = "SPACE" (+RS232)
        goto    inch            ; ELSE KEEP LOOKING FOR A START BIT
        movlw   d'08'           ; START SERIAL INPUT SEQUENCE
        movwf   TEMP            ; COLLECT 8 DATA BITS
        clrf    SERBUF          ; CLEAR SERIAL CHARACTER BUFFER
        call    half_baud       ; DELAY FOR ONE HALF BAUD TIME
        btfss   GPIO,3          ; FALL THRU IF START BIT STILL = "SPACE"
        goto    inch            ; ELSE IT WAS JUST A NOISE SPIKE, LOOP
inch1
        call    baud            ; DELAY ONE BAUD-BIT TIME ( = 1/BAUD-RATE)
        bcf     STATUS,0        ; CLEAR THE CARRY BIT
        rrf     SERBUF,F        ; ROTATE CRY -> MSB, ROTATE MSB RIGHT
        btfss   GPIO,3          ; IS INPUT = "SPACE" (+RS232) ?
        bsf     SERBUF,7        ; ...SKIP IF YES, ELSE SET BIT TO LOGIC '1'
        decfsz  TEMP,F          ; EIGHT COUNTS YET?
        goto    inch1           ; ...NO, GET ANOTHER BIT
        call    baud            ; DELAY FOR THE FIRST STOP BIT
        movf    SERBUF,W        ; Put the character in reg 'W'
        retlw   0               ; NOTE: REM THIS OUT IF YOU NEED AN "ECHO"
                                ; ...AND FALL THROUGH TO THE 'OUTCH' ROUTINE
```

Continued on the next page



Serial Port DB-25 Male

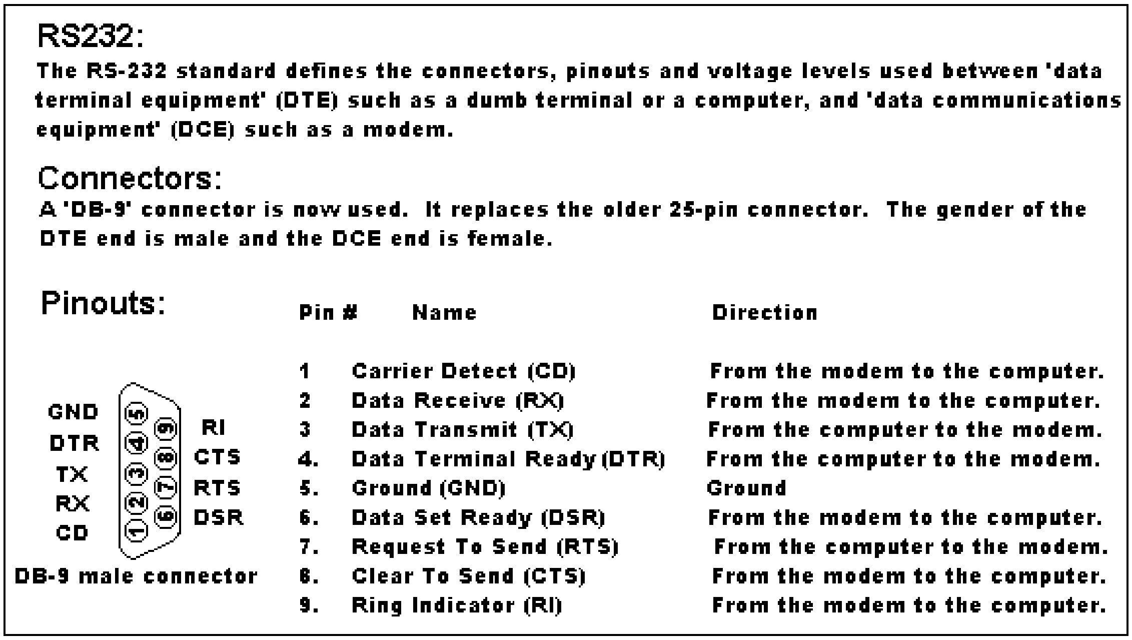
1. (Grounded Shield)
2. TxD Transmit Data
3. RxD Received Data
4. RTS Request to Send
5. CTS Clear to Send
6. DSR Data Set (Modem) Ready
7. GND Signal Ground
8. DCD Data Carrier Detect
9. to 19. Generally Unused
20. DTR Data Terminal Ready
21. Generally unused
22. RI Ring Indicator
23. to 25. Generally Unused

PIC Software UART Routines (Continued)

```
; ******************************************************************
;                                OUTCH ROUTINE
; THIS ROUTINE OUTPUTS RS232 DATA WITHOUT AN INVERTER
; THIS ROUTINE USES AN 8-DATA BIT PER CHARACTER PROTOCOL
; TO PRINT A CHARACTER, LOAD BYTE INTO REG 'W' and CALL OUTCH
; GPIO,2 = TX (MARK = 0, SPACE = 1)   ; USE NO INVERTER ON THE OUTPUT.
; VARIABLES USED: 'TEMP' AND SHARE REG 'SERBUF' WITH THE ROUTINE 'inch'
; CALLS THE ROUTINE 'baud' FOR THE BAUD-RATE TIMING.
; ******************************************************************
outch                          ; THIS ROUTINE USES 8 DATA BITS
        movwf   SERBUF          ; SERBUF CONTAINS CHARACTER TO XMT
        movlw   8               ; THE CHARACTER HAS 8 BITS
        movwf   TEMP
        bsf     GPIO,2          ; SET START-BIT TO A "SPACE"
        call    baud            ; WAIT ONE BAUD TIME
outch1
        rrf     SERBUF,F        ; ROTATE THE FIRST BIT INTO CARRY
        btfss   STATUS,0        ; TEST THE CARRY BIT
        bsf     GPIO,2          ; IF BIT IS 0 SET OUTPUT PIN TO A "1" (SPACE)
        btfsc   STATUS,0        ; TEST THE CARRY BIT AGAIN
        bcf     GPIO,2          ; IF BIT IS 1 SET OUTPUT PIN TO A "0" (MARK)
        call    baud            ; ONE BAUD-BIT DELAY
        decfsz  TEMP,F          ; IF COUNT IS ZERO THEN XMIT A STOP BIT
        goto    outch1          ; ...ELSE XMIT NEXT BIT

        rrf     SERBUF,F        ; ROTATE CARRY, GET THE MSB BACK INTO BIT 7
        bcf     GPIO,2          ; SET PIN TO A "MARK"(-RS232) FOR THE STOP BIT
        call    baud            ; FIRST BAUD-BIT DELAY
        call    baud            ; SECOND BAUD-BIT DELAY
        retlw   0               ; RETURN WITH THE CHARACTER IN SERBUF
```

Continued on the next page.



RS232:

The RS-232 standard defines the connectors, pinouts and voltage levels used between 'data terminal equipment' (DTE) such as a dumb terminal or a computer, and 'data communications equipment' (DCE) such as a modem.

Connectors:

A 'DB-9' connector is now used. It replaces the older 25-pin connector. The gender of the DTE end is male and the DCE end is female.

Pinouts:

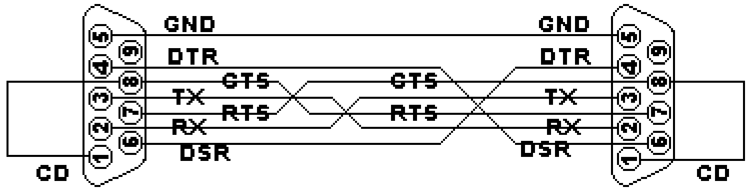
Pin #	Name	Direction
1	Carrier Detect (CD)	From the modem to the computer.
2	Data Receive (RX)	From the modem to the computer.
3	Data Transmit (TX)	From the computer to the modem.
4.	Data Terminal Ready (DTR)	From the computer to the modem.
5.	Ground (GND)	Ground
6.	Data Set Ready (DSR)	From the modem to the computer.
7.	Request To Send (RTS)	From the computer to the modem.
8.	Clear To Send (CTS)	From the modem to the computer.
9.	Ring Indicator (RI)	From the modem to the computer.

PIC Software UART Routines (Continued)

```
; ********************************************************************
;                       BAUD ROUTINE @ 4 MHz
; BAUD RATE:      CONSTANT:
;   1200 Baud     D'137'
;   2400 Baud     D'68'
;   4800 Baud     D'34'
;   9600 Baud     D'16'
;  19200 Baud     D'8'
;  38400 Baud and up - use 'NOP' delays
; VARIABLES USED: REG 'COUNT'
; ROUTINES CALLED: NONE
; ********************************************************************
baud                            ; AT 2400 BAUD THE PERIOD IS 416.6 US
                                ; CLK = 4MHz
        movlw   D'68'           ; 1 US   (BAUD RATE CONSTANT)
        movwf   COUNT           ; 1 US
baud1
        decfsz  COUNT,F         ; 1 US (+ 1 US MORE IF SKIP)
        goto    baud1           ; 2 US
                                ; FALL THRU...AFTER 1+1+3x68+1 = 207 US
half_baud
        movlw   D'68'           ; 1 US
        movwf   COUNT           ; 1 US
hbaud1
        decfsz  COUNT,F         ; 1 US (+ 1 US MORE IF SKIP)
        goto    hbaud1          ; 2 US
        retlw   0               ; ...AFTER 1+1+3x68+1 = 207 US  (X2=414 US)
```

Continued on the next page.

A 'Null' Modem: A connection between two DTE devices.



PIC Software UART Routines (Continued)

```
; ********************************************************************
;                            INCH_N
; THIS ROUTINE INPUTS RS232 DATA USING AN INVERTER, LIKE THE MAX232.
; THIS ROUTINE USES A 8-DATA BIT PER CHARACTER PROTOCOL
; GPIO,0 = RX (MARK = 1, SPACE = 0).
; TO RECIEVE A CHARACTER, CALL inch_n, THE RECEIVED CHARACTER IS PLACED
; IN THE REG 'W' AND IN THE REG 'SERBUF'.
; THE RECEIVED CHARACTER WILL ECHO IF 'RETLW   0' IS REM-ED OUT.
; VARIABLES USED: REG 'TEMP' AND REG 'SERBUF' BOTH VARIABLES ARE
;                 SHARED WITH THE 'outch_n' ROUTINE.
; ROUTINES CALLED: 'half_baud'AND 'baud' TO SET THE BAUD-RATE
; ********************************************************************
inch_n
        btfsc   GPIO,0          ; SKIP ON START BIT = 1 (A "MARK")
        goto    inch_n          ; ELSE KEEP LOOKING FOR A START BIT
        movlw   8               ; START SERIAL INPUT SEQUENCE
        movwf   TEMP            ; COLLECT 8 DATA BITS
        clrf    SERBUF          ; CLEAR SERIAL CHARACTER BUFFER
        call    half_baud       ; DELAY FOR ONE HALF BAUD TIME
        btfsc   GPIO,0          ; FALL THRU IF START BIT STILL = 1 (A "MARK")
        goto    inch_n          ; ELSE IT WAS JUST A NOISE SPIKE, KEEP LOOKING
inch_n1
        call    baud            ; DELAY ONE BAUD-BIT TIME ( = 1/BAUD-RATE)
        bcf     STATUS,0        ; CLEAR THE CARRY BIT
        rrf     SERBUF,F        ; ROTATE CRY -> MSB, ROTATE MSB RIGHT
        btfsc   GPIO,0          ; IS IT A "MARK" ?
        bsf     SERBUF,7        ; ...SKIP IF YES, ELSE SET BIT TO LOGIC '1'
        decfsz  TEMP,F          ; EIGHT COUNTS YET?
        goto    inch_n1         ; ...NO, GET ANOTHER BIT
        call    baud            ; DELAY FOR THE STOP BIT
        movf    SERBUF,W        ; PUT THE RECEIVED CHARACTER IN REG 'W'
        retlw   0               ; NOTE: REM THIS OUT IF YOU NEED AN "ECHO"
                                ; ...AND FALL THROUGH TO THE 'OUTCH' ROUTINE
```

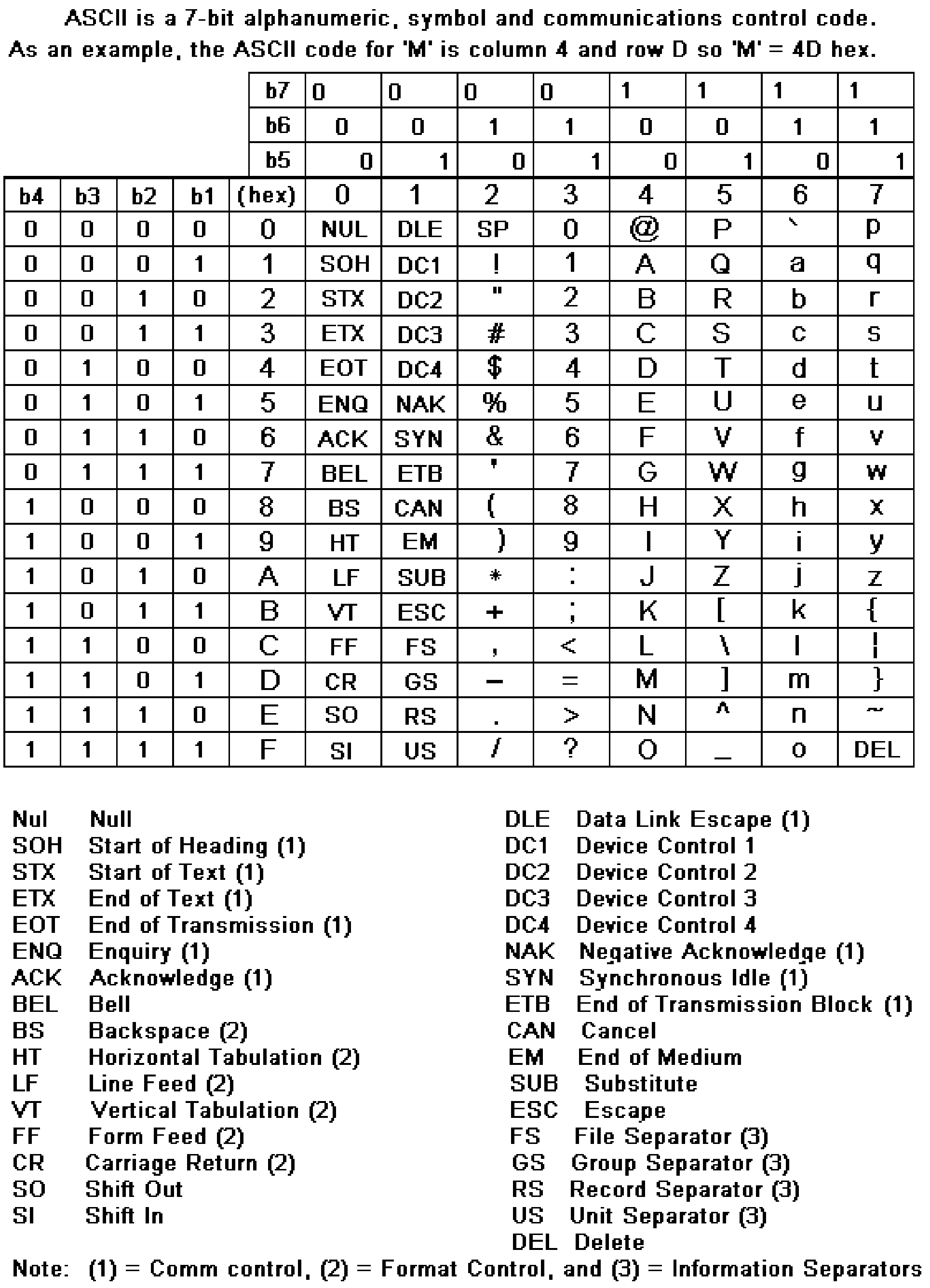
Continued on the next page.

PIC Software UART Routines (Continued)

```
;*********************************************************************
;                               OUTCH_N
; THIS ROUTINE OUTPUTS RS232 DATA THROUGH AN INVERTER.
; THIS ROUTINE USES AN 8-DATA BIT PER CHARACTER PROTOCOL.
; TO PRINT A CHARACTER, LOAD BYTE INTO REG 'W' AND CALL outch_n.
; GPIO,1 = TX (MARK = 1, SPACE = 0)   ; USE INVERTER ON THE OUTPUT
; VARIABLES USED: REG 'TEMP' AND SHARE REG 'SERBUF' WITH THE ROUTINE
;  'inch_n'
; CALLS THE ROUTINE 'baud' TO SET THE BAUD-RATE TIMING.
;*********************************************************************
outch_n                         ; THIS ROUTINE USES 8 DATA BITS
        movwf   SERBUF          ; SERBUF CONTAINS CHARACTER TO XMT
        movlw   8               ; THE CHARACTER HAS 8 BITS
        movwf   TEMP
        bcf     GPIO,1          ; SET START-BIT TO A "SPACE"
        call    baud            ; WAIT ONE BAUD TIME
outch_n1
        rrf     SERBUF,F        ; ROTATE THE FIRST BIT INTO CARRY
        btfss   STATUS,0        ; TEST THE CARRY BIT
        bcf     GPIO,1          ; IF BIT IS 0 SET OUTPUT PIN TO A "0" (SPACE)
        btfsc   STATUS,0        ; TEST THE CARRY BIT AGAIN
        bsf     GPIO,1          ; IF BIT IS 1 SET OUTPUT PIN TO A "1" (MARK)
        call    baud            ; ONE BAUD-BIT DELAY
        decfsz  TEMP,F          ; IF COUNT IS ZERO THEN XMIT A STOP BIT
        goto    outch_n1        ; ...ELSE XMIT NEXT BIT

        rrf     SERBUF,F        ; ROTATE CARRY, GET THE MSB BACK INTO BIT 7
        bsf     GPIO,1          ; SET PIN TO A 1 (A "MARK") FOR THE STOP BIT
        call    baud            ; FIRST BAUD-BIT DELAY
        call    baud            ; SECOND BAUD-BIT DELAY
        retlw   0               ; RETURN WITH THE CHARACTER IN SERBUF
```

Continued on the next page



ASCII is a 7-bit alphanumeric, symbol and communications control code.
As an example, the ASCII code for 'M' is column 4 and row D so 'M' = 4D hex.

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
b4	b3	b2	b1	(hex)	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	A	LF	SUB	*	:	J	Z	j	z
1	0	1	1	B	VT	ESC	+	;	K	[	k	{
1	1	0	0	C	FF	FS	,	<	L	\	l	¦
1	1	0	1	D	CR	GS	−	=	M	]	m	}
1	1	1	0	E	SO	RS	.	>	N	^	n	~
1	1	1	1	F	SI	US	/	?	O	_	o	DEL

Nul	Null
SOH	Start of Heading (1)
STX	Start of Text (1)
ETX	End of Text (1)
EOT	End of Transmission (1)
ENQ	Enquiry (1)
ACK	Acknowledge (1)
BEL	Bell
BS	Backspace (2)
HT	Horizontal Tabulation (2)
LF	Line Feed (2)
VT	Vertical Tabulation (2)
FF	Form Feed (2)
CR	Carriage Return (2)
SO	Shift Out
SI	Shift In
DLE	Data Link Escape (1)
DC1	Device Control 1
DC2	Device Control 2
DC3	Device Control 3
DC4	Device Control 4
NAK	Negative Acknowledge (1)
SYN	Synchronous Idle (1)
ETB	End of Transmission Block (1)
CAN	Cancel
EM	End of Medium
SUB	Substitute
ESC	Escape
FS	File Separator (3)
GS	Group Separator (3)
RS	Record Separator (3)
US	Unit Separator (3)
DEL	Delete

Note: (1) = Comm control, (2) = Format Control, and (3) = Information Separators

For sample source code and further information, please see the Nano-Lab 1002 project manual.

Datadog Systems
www.Datadog.com

END