

CP-SAR Serial/Analog/RAM Co-Processor User Manual

DOCUMENTATION

Overview	2
Pinout & Available Commands	3
Available Examples	4
Command Details:	
Serial Port	5
Analog to Digital	6
Memory	7
Scratch Pad	8
PWM	9
Miscellaneous	11
Wiring/Updating Firmware	12

MAIN FEATURES:

- One Pin Bi-Directional Serial Communication with Host
- Large Serial Buffer with Commands for Searching & Handling Data
- Analog Inputs down to 57uV resolution
- 3 Analog Ranges of 1.1v / 2.5v / 5.0v
- On-board Temperature Sensor
- PWM for Analog Out, Control, or Timing Ticks
- Up to 240 Bytes of RAM Storage
- Scratch Pad for fast Single Byte Transfers
- User Loadable Updates/Revisions

OBJECTIVE:

To supplement the functionality of Stamps with a CoProcessor that includes the most commonly needed extra functions, namely a Buffered Serial Port, Analog In, PWM Out, and RAM.

The CP-SAR also recognises the restraints under which a Stamp often operates. For example, capturing and buffering serial port data is only part of a solution. Often the internal RAM space does not exist to then import and extract relevant portions.

But with the addition of simple Commands to remotely 'Find', 'Move' and provide easy access to that data, then a complete solution is created.

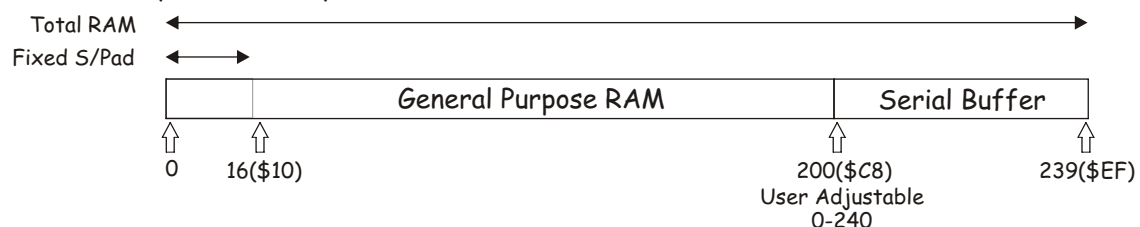
TASKS IN PARALLEL:

The CP-SAR itself can have an active Serial Buffer (monitoring for, or receiving Data), yet still accept Commands to Read & Write RAM, Read Analog, Write PWM without any interaction.

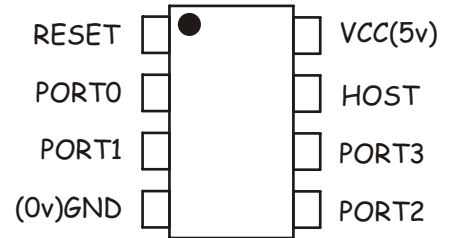
Example #5 in 'RAM.bs2' demonstrates this parallel operation with a 38.4k Serial Receive whilst the Host is running at 4.8k and continuously Reading & Writing RAM.

RAM ALLOCATION:

The General Purpose RAM and Serial Buffer occupy opposite ends of 240 Bytes of RAM. The first 16 bytes of the GP RAM are allocated to a Scratch Pad. That allows their Addresses of 0-15 to be included within the single-byte Command for fast simple transfers. The allocation of bytes between GP and S/Buffer is by default 200/40 but can be determined by the User - see Rd_Setup & Wr_Setup.bs2.



	Serial Port	Single A/D	Diff A/D	PWM
PORT0		X	X	
PORT1		X	X	
PORT2				X
PORT3	X			X



Cmd#	Name	Description	Page
\$00	Rd_Stat	Read Status of Serial Buffer	5
\$10	Sbuf	Specify Data to be Collected	5
\$20	SBuf_S	Specify Data to be Collected	5
\$30	A2D	Analog to Digital Conversion	6
\$40	Diff_A2D	Differential A/D Conversion	6
\$50	UpDte_PWM	Update PWM Pulse Width	10
\$60	Wr_SPad	Write to Scratch Pad	8
\$70	Rd_SPad	Read from Scratch Pad	8
\$80	Wr_RAM	Write to RAM	7
\$90	Rd_RAM	Read from RAM	7
\$A0	Find	Find Consecutive Characters	7
\$B0	Move	Move a Block within RAM	7
-	-	-	
\$FD	Init_PWM	Initialise PWM Modes	9
\$FE	Wr_Setup	Write Setting to EE Memory	11
\$FF	Rd_Setup	Read Current Settings	11

The Ready-to-Run Examples are intended to be the most valuable part of the information provided.

It is suggested that the User runs most of the Examples to gain a feeling of what is possible - there are over 30 Examples/Modes of operation contained in the 9 Files and all written for the Parallax Basic Stamp.

They have been tested with the Bs2, and because they use standard SerIn/Out they should be compatible with all other versions of the Stamp.

Two LEDs and a Potentiometer (5-10k) would be useful for PWM and A/D Examples respectively.

Rd_Setup.bs2 - Returns current settings of all User Options

SBuf.bs2 - 7 Examples of collecting Data using Qualifier/Terminator Characters, Packet Data with Header Length, Addressable Nodes etc.

SBuf_S.bs2 - 7 Examples of collecting data but placing the Status directly on the communication line itself

A2D.bs2 - 5 Cases of Analog to Digital conversion using 1.1/2.5/5V References plus reading the Internal Temperature Sensor

D_A2D.bs2 - 6 Cases of Differential Analog to Digital conversion optionally using 20x Gain for 57uV Resolution

PWM.bs2 - 3 Modes of operation including the generation of Timing Ticks

RAM.bs2 - 5 Examples covering from Read/Write to Find, Move and Scratch Pad operation

Wr_Setup.bs2 - Details all the User adjustable Options

SBuf - Collect & Buffer Serial Data

\$10+%----, 2nd Byte, 3rd Byte, etc.

The Command LS Nibble determines how many Bytes will follow, thus:

%xx-- 00/01/10/11 0-3 Count of Start Chars (needed to qualify Serial Data)

%--1- Stop Char (to terminate Serial Data)

%---1 Length Byte (to terminate Serial Data)

e.g. \$16 says 1 Start, 1 Stop, therefore 2 Bytes will follow the Command Byte.

See **SBuf.bs2 Examples**.

Rd_Stat - Read Sbuf Status

\$00 Single Byte Command returns a False/True (0/1) Byte indicating whether the required data has been collected. All other Commands for A/D, PWM, and RAM can still be used whilst the 'Sbuf' Command is active.

See **SBuf.bs2 Examples**.

SBuf_S - Collect & Buffer Serial Data - Display Status on the Communication Wire

\$20+%----, 2nd Byte, 3rd Byte, etc.

The Command LS Nibble determines how many Bytes will follow, thus:

%xx-- 00/01/10/11 0-3 Count of Start Chars (needed to qualify Serial Data)

%--1- Stop Char (to terminate Serial Data)

%---1 Length Byte (to terminate Serial Data)

e.g. \$16 says 1 Start, 1 Stop, therefore 2 Bytes will follow the Command Byte.

See **SBuf_S.bs2 Examples**.

If other CP Functions are not needed in parallel to Sbuf, then this mode has the advantage that the Comm. Wire can be simply tested to see the Status of False/True (0/1) for Buffer Ready. Other Commands are not possible because the CP is continuously driving that single wire to show the Status.

A2D - Read the Analog Value on Port0 or Port1

\$30+%- Single Byte Command - Returns 1 or 2 Bytes as specified below

The Command LS Nibble determines

%xx-- RefV where: 00,01,10,11 = 5v,1.1v,2.56v,'1.1v with TempSensor as Input'

%--x- 0/1 returns 2Bytes/'1Byte of 8 MS Bits' as the result

%---x 0/1 Selects whether Port0 or Port1 is read

Performs a 10 Bit A/D Conversion on Port0 or 1 using a the choice of 3 Reference voltages. The 10 bits are returned as 2 Bytes or a single Byte of the 8 MS Bits. When using 1.1v Reference the resolution is 1.07mV.

The on-board Temperature can also be read using a fixed 1.1v Reference.

See A2D.bs2 Examples.

Diff_A2D - Read the Analog Difference between Port0/Port1 (as -ve/+ve)

\$40+%- Single Byte Command

The Command LS Nibble determines

%xx-- RefV where: 00,01,10,11 = 5v,1.1v,2.56v,User RefV on Port2/Pin5

%--x- If = 1 Expects 0.1mFd Cap to Port2 for Decoupling Int VRef (2.5v only)

%---x If = 1 the Internal Gain is increased from 1x to 20x

Performs a 10 Bit A/D Conversion on the difference between Port1 minus Port0 (must be a +ve result). There is an optional on-board Gain of 20 which allows a resolution of 52.7uV when using the 1.1v Ref..

There is also provision for a User Ref Voltage to be applied to Port2, if that pin is not required for PWM. The same pin can also be used for de-coupling the 2.56v Ref, again if not required for PWM.

See D_A2D.bs2 Examples.

Wr_RAM Writes the Count of Bytes starting at the specified Address

\$80+0-15, Start_Addr, Data,,,

- The Command LS Nibble is a Count of 1-15 to be written, and 0 represents a Count of 16
- Start_Addr is a single Byte, range 0-239 (\$EF)
- Data are Bytes matching the specified Count

See RAM.bs2, SBuf.bs2, SBuf_S.bs2 Examples.

Rd_RAM Reads the Count of Bytes starting at the specified Address

\$90+0-15, Start_Addr

- The Command LS Nibble is a Count of 1-15 to be read, and 0 represents a Count of 16
- Start_Addr is a single Byte, range 0-239 (\$EF)

See RAM.bs2, SBuf.bs2, SBuf_S.bs2 Examples.

Find Returns the Address immediately following the located set of Bytes

\$A0+%-----, 2nd Byte, 3rd Byte, etc.

The Command LS Nibble determines how many Bytes will follow, thus:

- ' %1--- Avoids need for Address Byte & uses \$10 (immediately after S/Pad)
- ' %-1-- Avoids need for Address Byte & uses the S/Buffer Start Address
- ' %--xx Count of 1-3 Bytes to be found - 0 will fail & return 0 as Address

e.g. \$A7 says starting at the S/Buffer, locate the 3 attached and consecutive Bytes, then return the address that follows - if the search fails '0' is returned.

See RAM.bs2, SBuf.bs2, SBuf_S.bs2 Examples.

Move Moves the Count of Bytes from Source Address to Destination Address

\$B0+0-15, Srce_Addr, Dest_Addr

- The Command LS Nibble is a Count of 1-15 to be moved, and 0 represents a Count of 16
- Srce_Addr Range of 0-239(\$EF)
- Dest_Addr Range of 0-239(\$EF)

See RAM.bs2, SBuf.bs2, SBuf_S.bs2 Examples.

Wr_SPAD Writes one Byte to the Scratch Pad (S/Pad = RAM Addresses 0-15)

\$60+0-15, Data

- The Command LS Nibble is the Address 0-15 where the Byte will be written
- Data a single Byte

See **RAM.bs2**, **SBuf.bs2**, **SBuf_S.bs2** Examples.

Rd_SPAD Reads one Byte from the Scratch Pad (S/Pad = RAM Addresses 0-15)

\$70+0-15 Single Byte Command

- The Command LS Nibble is the Address 0-15 where the Byte will be read from

See **RAM.bs2**, **SBuf.bs2**, **SBuf_S.bs2** Examples.

Init_PWM Setup the Waveform Generator common to both Port 2 & 3

\$FD, 2nd Byte (, 3rd Byte)

The 2nd Byte specifies the following:

%x--- If = 1 Connect Port3 to Waveform Generator (Ignored in Tick Mode)

%-x-- If = 1 Connect Port2 to Waveform Generator

%--xx Modes 00,01,10,11 = Stop, Ticks, Edge PWM, Centered PWM

3rd Byte is not required when Mode0(00) - Stop

Note that only the Waveform Generator is Stopped. If the current connection status (those last established) for Port2 & 3 will remain unchanged, and if 'Connected' they will continue to Output unless the Connect Bits are at 0 in the Stop Command. If not Disconnected, the Output level at the time of Stop will be unknown.

However if just a known Static level is needed, then simply use the UpDte_PWM Command using 0 or 255 as appropriate for each Port - if a Stop is subsequently issued it will not affect the Output level of those still connected.

3rd Byte if Mode1(01) - Ticks

1-255 in units of 0.512mSecs for the Period between Ticks (same edges) Examples:

125mS (8 Tics/sec) $125/0.512 = 244$

62.5mS (16 Tics/sec) $62.5/.512 = 122$

25mS (40 Tics/sec) $25/0.512 = 49$

20mS (50 Tics/Sec) $20/0.512 = 39$

3rd Byte if Mode2(10) - Edge Aligned PWM

Establishes any Inversions & Sets the Frequency of the 0-255 Counter, which in turn sets the Period within which the PWM On/Off will operate.

%--x//---- If = 1 then Port3 Waveform will be Inverted

%---x//---- If = 1 then Port2 Waveform will be Inverted

%----//-xxx For 000 Freq = 31,250Hz Period 32 uSecs

001 = 3,906.6 256

010 = 488.3 2,048

011 = 120.1 8,192

100 = 30.5 32,768

Init_PWM continued:

3rd Byte if Mode3(11) - Center Aligned PWM

Establishes any Inversions & Sets the Frequency of the $2^{*(0-255)-2}$ Up/Down Counter, which in turn sets the Period within which the PWM On/Off will operate.

%--x//----	If = 1 then Port3 Waveform will be Inverted
%---x//----	If = 1 then Port2 Waveform will be Inverted
%----// -xxx	For 000 Freq = 15,686.6Hz Period 64 uSecs
001	= 1,860.8 537
010	= 245.1 4,080
011	= 61.3 16,313
100	= 15.3 65,359

See many Examples in PWM.bs2

UpDte_PWM Update the Pulse Width of a specific Port

\$50+ 2/3, 2nd Byte

The Command LS Nibble selects Port 2 or 3 to receive the 2nd Byte as 'Pulse Width'

- 2nd Byte holds the Value of 0-255 to define its new Pulse Width

Note that after power up, the Pulse Width values will be zero but Port2 & 3 will not be Outputting until the Init_PWM Command is issued. If the first values need to be other than zero, then the UpDte_PWM can be used before Init_PWM to achieve that.

See detailed Examples of all Modes, with Waveforms, in PWM.bs2

Rd_Setup Returns Device ID and current settings from EE Memory

\$FF

The code Rd_Setup.bs2 makes a good first program to run, and will return the following:

- Model/Version
- Calibration 1 and 2
- Transmit to Host Delay
- GP RAM Size/SerBuf Size
- Baud Index for Host/External
- Serial Polarity Host/External

The Calibration bytes 1 & 2 should always be equal - in the unlikely they are not, run the User Calibration program Calib.bs2.

Wr_Setup Writes User settings into EE Memory

\$FE

Note - It is strongly recommended that the Default Settings be retained until the Examples are no longer needed. The program comments fully explain all settings.

.

Recommended Wiring with Decoupling Capacitor:

The wiring is shown as a layout in order to emphasise the optimum positioning of the 0.1mFd decoupling capacitor supplied with the CP.

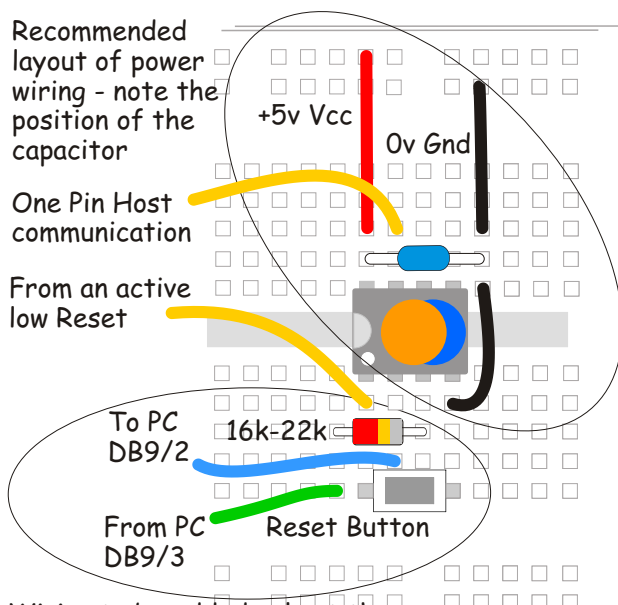
Unlike PICs, which spread the processing of each instruction over 4 oscillator cycles, the AVR uses only 1. That creates a heavy momentary demand on the two edges of the single cycle, but still a very low average.

The use and positioning of the capacitor is a small price to pay for 4x computing speed plus another 30% from its advanced internal architecture.

Updating Firmware using the 15v PC Serial signal:

The 16k to 22k resistor shown in series with the Green wire from a PC Serial Port or a USB to Serial Adapter, is not optional - without the resistor the CP will be damaged.

Equally important is that the Blue wire connection does go to the Db9 Pin2 which is voltage free - note that other DB9 Pins also have 15v present, and just momentarily making contact with one of them will damage the CP.



Updating from the PC:

The latest version e.g. SAR_vXX.bin and the associated PC Loader AVRO-Ldr15.exe will be available here:

www.rhombus-tek.com/Firmware.html

The Loader is self extracting and defaults to load within the Rhombus folder.

With the Loader Hex option unchecked, & the downloaded SAR_vXX.bin file selected, then simply hit 'Write Flash', and when Prompted operate the Reset Button.

As a part of downloading, the Loader also re-calibrates the CP's Internal Oscillator if needed. Disconnect the PC before further use.

Wiring to be added only at the time of an Optional Update.

Alternatively, the Stamp's Db9 and Reset Button could be used by removing the Stamp & jumpering its socket Pins1 & 2 as DB9/2 & DB9/3 respectively - the 22k resistor must still be used. Note there is no 2nd chance if the 15v signals are reversed.

