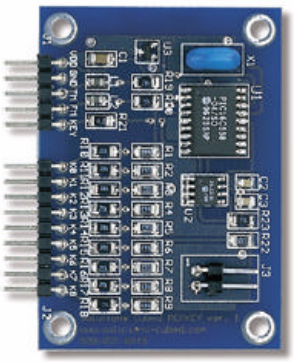


MEMKey

Keypad Encoder
*4X5 Matrix
*Programmable Rows & Columns
*PC/AT or Serial Output

SOLUTIONS CUBED
256 East 1st Street
Chico, CA 95928
Phone (530) 891-8045
Fax (530) 891-1643
www.solutions-cubed.com

MINIATURE ENGINEERING MODULES

MEMKey
Serial and PC/AT Keypad Encoder Module

FEATURES

- 4X5 Matrix
- Programmable rows and columns
- Programmable key values
- PC/AT or serial output
- Automatic or polled key reporting
- Programmable debounce and typematic times
- 64 Bytes of user-accessible EEPROM
- Standard TTL levels
- No external components
- Easy-to-use SIP package

DESCRIPTION

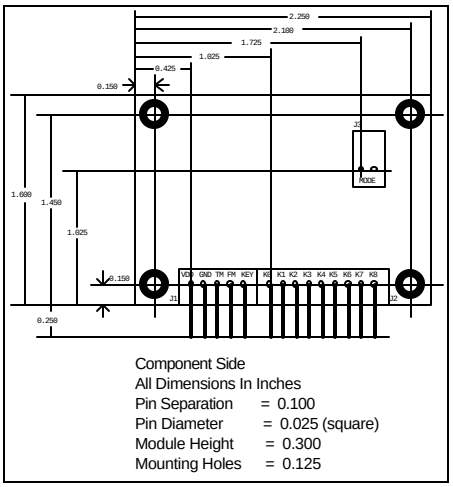
The MEMKey is a fully programmable keypad encoder. Using a jumper, the MEMKey supports either a simple serial communication protocol or the standard PC/AT communication protocol. In either communication mode, it can decode key matrixes of up to 4 columns by 5 rows. The rows and columns can be programmed to match the row-column configuration of any off the shelf keypad.

The value returned by the MEMKey can be programmed to any standard value. In addition, the debounce time and typematic rate are fully programmable. All programmable values are stored in non-volatile memory so they are saved when power is off.

When operating with the serial protocol, the MEMKey communicates at 2400 baud, 8N1, LSB first, asynchronous. This can be either a 1 wire or a 2 wire interface.

In addition, there are 64 bytes of EEPROM memory which is made available to the user as scratch pad space.

PIN CONFIGURATION AND MECHANICAL SPECS



Component Side
All Dimensions In Inches
Pin Separation = 0.100
Pin Diameter = 0.025 (square)
Module Height = 0.300
Mounting Holes = 0.125

VDD	Power supply pin
GND	Ground pin
TM	Serial Communication to master from MEMKey (open collector). CLOCK line in PC/AT communication.
FM	Serial Communication from master to MEMKey (open collector). DATA line in PC/AT communication.
KEY	Keypress output, active high, current limited, serial mode only.
K0 → K8	Key matrix connections.

SPECIFICATIONS

ABSOLUTE MAXIMUM RATINGS

note: These are stress ratings only. Stresses above those listed below may cause permanent damage and/or affect device reliability. The operational ratings should be used to determine applicable ranges of operation.

Storage Temperature	-65°C to +125°C
Operating Temperature	-20°C to +80°C
Supply Voltage	0 to 7.0V
Voltage on any pin	-0.6V to (VDD+0.6V)

DC ELECTRICAL CHARACTERISTICS

At T_A = 25°C and VDD = 5.0V unless otherwise noted.

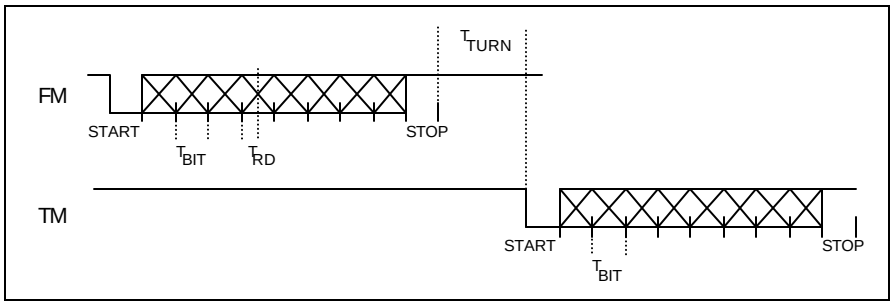
Characteristic	Symbol	Min	Typ	Max	Unit	Notes
Supply Voltage	VDD	3.0		5.5	V	
VDD rise time to ensure good reset	SVdd	0.05			V/ms	If this is not met, the MEMKey may start up in an unknown state.
Supply Current	Icc		3.5	6.0	mA	KEY inactive, no communication. Writing EEPROM adds 3mA.
FM Input Low Voltage	VIL	GND		0.2VDD	V	
FM Input High Voltage	VIH	2.0 0.2VDD+1 V		VDD	V	4.0<VDD<5.0 Full VDD range User may use better of two specs.
FM Input Weak Pull Up current	IFMPU	50	250	400	µA	VFM = GND Min value is at VDD Min, while Max value is at VDD max.
TM Output Low Voltage	VOLTM			0.6	V	
TM Output High Voltage	VIHTM	VDD			V	TM is open collector
TM Output Pull Up current	ITMPU	2.5	5.0	5.5	mA	TM open collector is tied to VDD with a 5% 1kΩ resistor.
KEYpin Output Low Voltage	VOLA			0.6	V	KEY pin has a 270Ω output impedance
KEY pin Output High Voltage	VOHA	VDD-0.7			V	KEY pin has a 270Ω output impedance
KEY pin Output current	IA			18.5	mA	KEY shorted to ground
KEY pin impedance	Ar		270		Ω	There is a series 5% 270 ohm resistor in line with the KEY output.

note: "Typ" values are for design guidance only and are not guaranteed.

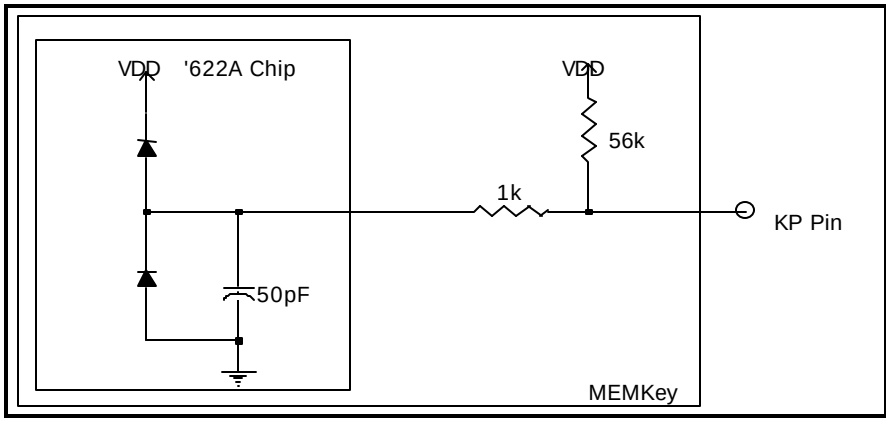
AC ELECTRICAL CHARACTERISTICS

At T_A = 25°C and VDD = 5.0V unless otherwise noted.

Characteristic	Symbol	Min	Typ	Max	Unit	Notes
Serial communication bit period	TBIT	414	416.7	419	µS	The bit period is determined by an on-board resonator, and is temperature sensitive.
Serial Offset when a bit is read	TRD	180	200	220	µS	This is used to ensure a bit is valid when read. A bit must be valid for at least this long in order for the communication to not be erroneous.
Time for a command from master to be responded to in serial mode	TTURN	2.0	2.5	3.0	mS	This time is used to allow for a master to change from transmission mode to reception mode.
Baud generator error	BGE		0.5	2	%	The baud generator has a 0.3% temperature stability from -20°C to +80°C



Communication Timing



KX Pin Input Structure

OPERATION

The MEMKey is a keypad encoder which can communicate via a serial interface or the standard PC/AT interface. In addition, it is fully programmable. Access to the programmable features is through the serial protocol. If the device is to be used in the PC/AT mode, it must first be programmed using the serial mode and then put into PC/AT mode. Details for using the MEMKey in the serial mode are given in the *Serial Operation* subsection. For further information on using the PC/AT mode see the subsection *PC/AT Operation.*

Key Map

The MEMKey maps the rows and columns of the keypad into the following key map. The map shows the physical key that corresponds to a particular row/column intersection. For example, if the key at row 3, column 2 was pressed down, the MEMKey would decode that as physical key 14 being pressed down.

Column/ Row	0	1	2	3
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19

Physical MEMKey key map

Serial Operation

In serial mode the MEMKey is a fully configurable keypad encoder. The value that the MEMKey sends to the host when a key is pressed can be individually programmed for each key. The KX pins can be configured in any row and column configuration. This allows for easy connections between the key matrix and the encoder. The amount of time that keys are debounced can be set. Typematic action can be enabled or disabled. In addition, the typematic delay and rate can be programmed. Also, the MEMKey can be configured to automatically report key presses. Or it can store the key values in an 8 key buffer and await a polling command from the host before it sends its key data. The KEY pin toggles high whenever there is key data in the buffer. Using this KEY pin allows for synchronization between the host and MEMKEY during poll mode. In addition, the serial mode is used to program the PC/AT key values, if they need to be different than the pre-configured values.

Serial Default Settings

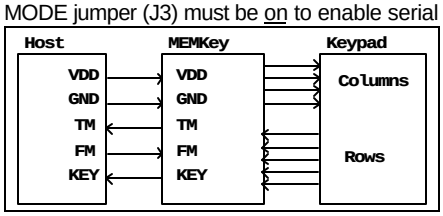
The MEMKey is pre-configured at the factory so it will work with no user setup necessary. The table below shows the default values for the MEMKey.

Item	Value
Key 0	‘00’H
Key 1	‘01’H
Key 2	‘02’H
Key 3	‘03’H
Key 4	‘04’H
Key 5	‘05’H
Key 6	‘06’H
Key 7	‘07’H
Key 8	‘08’H
Key 9	‘09’H
Key 10	‘0A’H
Key 11	‘0B’H
Key 12	‘0C’H
Key 13	‘0D’H
Key 14	‘0E’H
Key 15	‘0F’H
Key 16	‘10’H
Key 17	‘11’H
Key 18	‘12’H
Key 19	‘13’H
Pin K0	Row 0
Pin K1	Row 1
Pin K2	Row 2
Pin K3	Row 3
Pin K4	Column 0
Pin K5	Column 1
Pin K6	Column 2
Pin K7	Column 3
Pin K8	Row 4
Debounce Delay	10mS
Typematic Delay	500mS
Typematic Rate	1 per 100mS
Typematic Operation	On
Automatic Operation	On

Default settings

Hardware Hook up

The connection diagram below shows the basic setup for using the MEMKey. This is the easiest and simplest way to use the MEMKey. In addition, the MODE jumper (J3) must be on to enable serial mode.



Basic connection diagram

Power(VDD) must be supplied to the MEMKey from either a master processor or an external supply. When communication is taking place between the master and the MEMKey, both the master's ground and the MEMKey's GND pin must be at the same potential. The KEY pin can either be read by the master or it can drive other circuitry. As the diagram shows, the TM pin on the MEMKey provides the communication path to the master from the MEMKey; while the FM pin on the MEMKey provides the communication path from the master to the MEMKey. The keypad attached to the MEMKey can be of any type that fits within the 5 row by 4 column format. With the default settings, the MEMKey will match a Grayhill 96 series 4X4 matrix.

Communication Protocol

In the serial mode, communication with the MEMKey is accomplished with a two-wire (labeled TM and FM), asynchronous, serial communication channel. The FM pin carries data and commands from the master device to the MEMKey. The TM pin carries data and commands to the master device from the MEMKey.

All communication is 8N1, least significant bit first, 1 start bit, and 1 stop bit. The MEMKey communicates at 2400 baud. After receiving a command from the master, the MEMKey will wait approximately 2.5mS before responding (if a response is necessary). This time enables the master to prepare for the response.

Some commands write to the EEPROM on the MEMKey. After sending these commands, further commands that access the EEPROM must be withheld for at least 10mS. Commands that access EEPROM are marked with an * in the table below.

There is a frozen communication catch in the MEMKey. This is to enable the MEMKey to gracefully recover from communication that is interrupted. If a multibyte command is being sent to the MEMKey and more than 625mS elapses between bytes, the MEMKey ignores the communication that has occured thus far and resets.

Commands larger than '11'H are ignored.

Examples of the communication protocol and some of the commands are shown in the 'Command Set and Communication Examples' diagram.

Command Structure and Description

note: all values are given in decimal unless otherwise noted.

Command	Byte Sent
Read Key Buffer	'00'H
Read Number of Keys	'01'H
Program Typematic*	'02'H
Read Typematic	'03'H
Program Debounce*	'04'H
Read Debounce	'05'H
Program Rows & Columns*	'06'H
Read Rows & Columns	'07'H
Program User EEPROM*	'08'H
Read User EEPROM	'09'H
Program Serial Key*	'0A'H
Read Serial Key	'0B'H
Program PC/AT Key*	'0C'H
Read PC/AT Key	'0D'H
Program Configuration Byte*	'0E'H
Read Configuration Byte	'0F'H
Test Mode	'10'H
Default Reset*	'11'H

* = EEPROM access

Table of commands

Read Key Buffer ('00'H) After receiving this command the MEMKey sends all key press information in its buffer to the master. If there is no information in the key buffer, the MEMKey returns an 'FF'H. The oldest key presses are sent first (FIFO buffer). In automatic mode this command is most-likely never used.

Read Number of Keys ('01'H) After receiving this command the MEMKey sends the master the number of key presses it has in its buffer. This number will range between 0 and 8.

Program Typematic ('02'H) Typematic key action is defined as sending key press bytes for as long as a key is pressed. It consists of two variables: delay – the time before the repeat happens; and rate – the time between repeats after the first repeat. After sending this command, the master sends two more bytes: the delay and the rate. The bytes represent the time in 2.5mS increments. A value of 1 is the shortest delay and 255 is the largest (0 is defaulted to 255). For example, to set the typematic rate for a 500mS delay and a 100mS rate the following bytes would be sent to the MEMKey: '02'H, 'C8'H, '28'H.

Read Typematic ('03'H) After receiving this command, the MEMKey returns two bytes: the typematic delay and typematic rate.

Program Debounce ('04'H) When keys are pressed they do not immediately make a solid electrical connection. They "bounce" open and closed. If care is not taken, the device reading the key could read these bounces as separate key presses. The MEMKey has built in bounce protection. A key must be down continuously for as long as the debounce period before the key press is registered. After sending the command, the master sends one byte specifying the amount of debounce time is 2.5mS increments. 1 is the shortest delay and 255 is the largest (0 is defaulted to 255). For example, to set the debounce time to 50mS the following bytes would be sent to the MEMKey: '04'H, '14'H.

Read Debounce ('05'H) After receiving this command, the MEMKey returns the debounce delay byte.

Program Rows and Columns ('06'H) The KX pins on the MEMKey can be individually configured to be any column or row in the matrix. After sending the command, the master sends 9 more bytes. Each byte indicates which position the rows and columns are to take. The byte order is: row0, row1, row2, row3, row4, col0, col1, col2, col3. The value in each byte tells the MEMKey where to put the row or column. The following values represent the physical pins on the encoder

0 = K0
1 = K1
2 = K2
3 = K3
4 = K4
5 = K5
6 = K6
7 = K7
8 = K8

For example, to set the MEMKey's KX pins in the following order:: K0 = row0, K1 = row1, K2 = row2, K3 = row3, K4 = row4, K5 = col0, K6 = col1, K7 = col2, K8 = col3; the master would send the following bytes to the MEMKey: '06'H, '00'H, '01'H, '02'H, '03'H, '04'H, '05'H, '06'H, '07'H, '08'H.

There is no built in protection to ensure that one pin is not defined as both a column and a row; the user must make sure this unfortunate set of circumstances does not occur. If a value larger than 8 is received for any KX assignment, the MEMKey receives the rest of the bytes and then ignores the command.

Read Rows and Columns ('07'H) After receiving this command, the MEMKey returns the row and column assignments to the master. It uses the same format as the 'Program Rows and Columns' command.

Program User EEPROM ('08'H) There are 64 bytes of user accessible EEPROM, addressed '00'H through '3F'H. After sending the command byte, the master sends the address byte and then the data byte. Any address larger than '3F'H is simply wrapped back around to the beginning of the address space. For example, to program space '20'H with 'AA'H, the following bytes would be sent to the MEMKey: '08'H, '20'H, 'AA'H.

Read User EEPROM ('09'H) After sending the command, the master must send an address byte in the user EEPROM address space. After receiving the address byte, the MEMKey reads the EEPROM and then sends the value to the master.

Program Serial Key ('0A'H) This command is followed by two bytes: the physical key to program (see **Key Map**) and then the value that the master wants the MEMKey to send when that key is pressed. The key address must be between '00'H and '13'H or the command is ignored. The key value may be any value; however, because the MEMKey uses 'FF'H to indicate no key in the buffer, this value should not be used to avoid ambiguous results. For example to program key 3 to return 'EF'H when pressed, the following bytes would be sent: '0A'H, '03'H, 'EF'H.

Read Serial Key ('0B'H) The master follows this command with the physical key address (see **Key Map**) to read. After receiving the address the MEMKey sends the value of the serial key to the master.

Program PC/AT Key ('0C'H) This command is followed by two bytes: the physical key to program (see **Key Map**) and then the value the master wants the MEMKey to send when the key is pressed. The key address must be between '00'H and '13'H or the command is ignored. The value to send to the master is a look up value which corresponds to keys on a keyboard. This look up value is in the **PC/AT Operation** section. If a key look up value larger than 105 is sent, the value reverts to 0 which would return an A to the host if the key was pressed. For example, to program key 15 to return an X when pressed the following bytes would be sent: '0C'H, '0F', '17'H.

Read PC/AT Key ('0D'H) The master follows this command with the physical key address (see **Key Map**) to read. After receiving the address the MEMKey sends the look up value of the PC/AT key to the master.

Program Configuration Byte ('0E'H) The configuration byte tells the MEMKey if it should have typematic action and/or if it should report keys to the host automatically. These actions are controlled by bits in the configuration byte. The following figure shows the position of the bits. Bit 0 will always be '0' and is read only. Bits 3 through 8 are not used and may be used for general purpose storage, however this is not recommended for future compatibility. A '1' in the bit position turns on that particular action. For example, to enable automatic key reporting, and disable typematic mode, the master would send the following to the MEMKey: '0E', '02'.

7	6	5	4	3	2	1	0
-	-	-	-	-	T	A	0

Configuration byte
- \- - unused
- T - typematic
- A - automatic

Read Configuration Byte ('0F'H) After receiving this command the MEMKey sends the current configuration byte back to the host.

Test Mode ('10'H) After receiving this command the MEMKey performs two actions. It first toggles the KEY line high for 5µS and then low. It then sends the current MEMKey firmware version to the host.

Default Reset ('11'H) After receiving this command the MEMKey reverts to the default states for all programmable features. It reprograms the EEPROM with the default values and then resets. More information about the default states can be found in the Serial and PC/AT Default Settings sections.

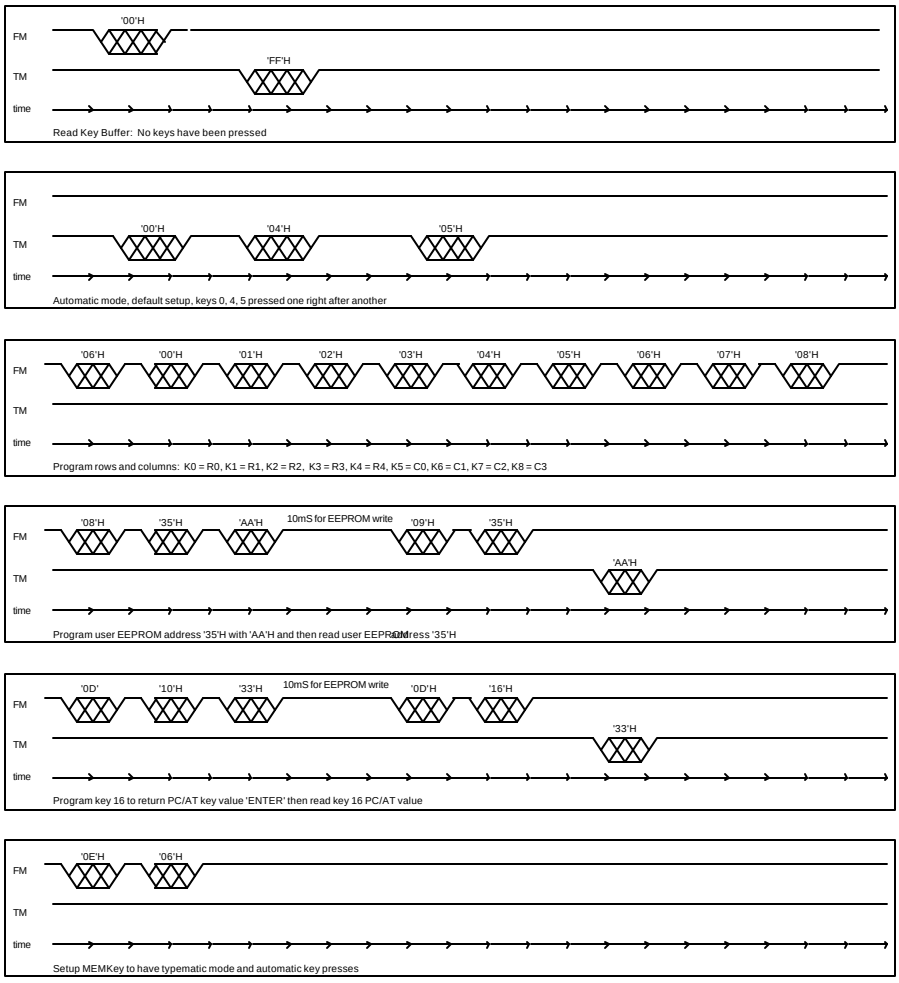
KEY Operation

In the serial mode, the KEY pin toggles high whenever there is key information in the buffer. The line does not go back low until all keys in the buffer have been released. For example, if the MEMKey was in poll mode and a key was pressed, the KEY pin would go high but the information would not be sent. The master would then send the 'Read Key Buffer Command' and then the key information would be sent. If the key still had not been released the KEY pin would still stay high. In this manner the master can track when no key is actually pressed down.

Typematic Action

Typematic only works on the last key pressed. If a key was pressed then another, the 2nd key would be typematic. If the 2nd key was then released, there would be no typematic on the 1st key.

In polled mode, typematic works only as fast as the buffer is actually read.



Command Set and Communication Examples

Ghost Key Protection

A ghost key situation can arise when three keys are pressed simultaneously and they form the vertices of a rectangle in relation to the row and column assignment. In this case the fourth corner of the rectangle will also be read as down.

The MEMKey combats this situation by implementing a simple key counter and allowing no more than 2 keys to be registered at one time. If more than 2 keys are pressed simultaneously, then no keys are registered as pressed. This ensures that no ghost key is decoded as pressed. Because of this method, system designers must ensure that three-key combinations are not implemented with the MEMKey, because the MEMKey will disallow any three-key combinations.

PC/AT Operation

In PC/AT mode the MEMKey operates as a standard keyboard that can be attached to any PC or single board computer that supports the PC/AT keyboard protocol.

If attributes of the MEMKey such as the row/column configuration or the key values returned for a key press need to be changed from the default values, the MEMKey needs to be programmed in the serial mode. See **Serial Operation** for more details.

The MEMKey supports a subset of the PC/AT protocol. It supports the mode II interface only. As such, mode III commands such as set make/break are emulated to ensure no communication error but they are not implemented. The table below gives all of the commands that are implemented and emulated. Because modes I and III are used very rarely, the MEMKey can interface to all desktop and laptop PCs and single board computers with no problems. In addition, the Set/Reset Indicators command is emulated because there are no indicator LEDs on the MEMKey.

Command	Action
Set/Reset Indicators	Emulated
Echo	Implemented
Select Alternate Scan Code	Emulated
Read ID	Implemented
Set Typematic	Implemented
Enable	Implemented
Default Disable	Implemented
Set Default	Implemented
Set All Keys	Emulated
Set Key Type	Emulated
Resend	Implemented
Reset	Implemented

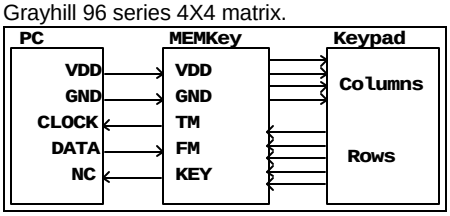
Supported commands

Hardware Hook up

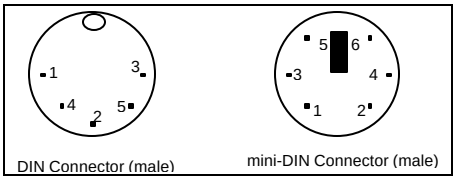
The connection diagram below shows the basic setup for using the MEMKey in PC/AT mode. In addition, the MODE jumper (J3) must be off to enable the PC/AT mode.

Power(VDD) is supplied by the standard PC/AT connector as is GND. The CLK and DATA lines are also supplied by the connector. The diagram below gives the standard pin outs for both the DIN and mini-DIN connectors that will mate with the connectors found on most PCs. Note the KEY pin on the MEMKey is not used. It is configured as an output low when in PC/AT mode.

The keypad attached to the MEMKey can be of any type that fits within the 5 row by 4 column format. With the default settings, the MEMKey will match a Grayhill 96 series 4X4 matrix.



Basic connection diagram



DIN Connectors (looking into cable)

Pin	DIN	mini-DIN
1	Clock	Data
2	Data	NC
3	NC	Ground
4	Ground	VDD
5	VDD	Clock
6	-	NC

DIN connector pin outs

For OEM customers that need the MEMKey customized, there are two options. Solutions Cubed can program the MEMKeys before shipping. Alternately, there is a serial MEMKey programming board which runs off of a PC serial port available from Solutions Cubed.

PC/AT Default Settings

The MEMKey is pre-configured at the factory so it will work with no user intervention. The table below shows the default values for the MEMKey.

Item	Value
Key 0	'00'H = A
Key 1	'01'H = B
Key 2	'02'H = C
Key 3	'03'H = D
Key 4	'04'H = E
Key 5	'05'H = F
Key 6	'06'H = G
Key 7	'07'H = H
Key 8	'08'H = I
Key 9	'09'H = J
Key 10	'0A'H = K
Key 11	'0B'H = L
Key 12	'0C'H = M
Key 13	'0D'H = N
Key 14	'0E'H = O
Key 15	'0F'H = P
Key 16	'10'H = Q
Key 17	'11'H = R
Key 18	'12'H = S
Key 19	'13'H = T
Pin K0	Row 0
Pin K1	Row 1
Pin K2	Row 2
Pin K3	Row 3
Pin K4	Column 0
Pin K5	Column 1
Pin K6	Column 2
Pin K7	Column 3
Pin K8	Row 4
Debounce Delay	10mS
Typematic Operation	On
Automatic Operation	On

Default settings

PC/AT Key Look Up

Because the values sent for a key in the PC/AT protocol often do not fit into one byte, a look up table is used to program which value needs to be sent for each key press. The following table gives all of the keys and their look up values for programming the MEMKey. When reading these values out in the serial mode, the MEMKey gives the look up value.

Key	Look Up	Key	Look Up
A	0	**F1**	53
B	1	**F2**	54
C	2	**F3**	55
D	3	**F4**	56
E	4	**F5**	57
F	5	**F6**	58
G	6	**F7**	59
H	7	**F8**	60
I	8	**F9**	61
J	9	**F10**	62
K	10	**F11**	63
L	11	**F12**	64
M	12	**Insert**	65
N	13	**Delete**	66
O	14	**Home**	67
P	15	**End**	68
Q	16	**Page Up**	69
R	17	**Page Dn**	70
S	18	**←**	71
T	19	**-**	72
U	20	**–**	73
V	21	**●**	74
W	22	**0**	75
X	23	**1**	76
Y	24	**2**	77
Z	25	**3**	78
0)	26	**4**	79
1!	27	**5**	80
2@	28	**6**	81
3#	29	**7**	82
4$	30	**8**	83
5%	31	**9**	84
6^	32	**.Del**	85
7&	33	**+**	86
8*	34	**-**	87
9(	35	*****	88
`~	36	**/**	89
-_	37	**Num Enter**	90
=+	38	**Num Lock**	91
[{	39	**Print Scrn**	92
]}	40	**Scrl Lock**	93
|	41	**Pause**	94
;:	42	**L Shift**	95
'"	43	**R Shift**	96
,<	44	**L Ctrl**	97
.>	45	**R Ctrl**	98
/?	46	**L Alt**	99
Escape	47	**R Alt**	100
Backspace	48	**key a**	101
Tab	49	**key b**	102
Caps	50	**L Window**	103
Enter	51	**R Window**	104
Space	52	**Menu**	105

PC/AT key map

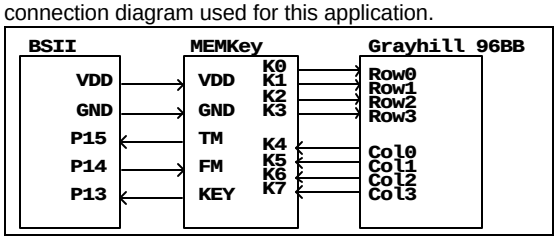
APPLICATIONS

The following examples show how to interface the MEMKey to various master processors in various configurations. These examples should make it fairly easy for an end user to custom design their own programs and uses for the MEMKey.

AN-300 'Interfacing to a Parallax BASIC Stamp II', AN-301 'Interfacing to a PC serial port', and AN-302 'Interfacing with a single wire', are all included here.

AN-300 Interfacing to a Parallax Basic Stamp II

The Parallax BASIC Stamp II makes an ideal choice for the master processor for the MEMKey for one main reason: ease of developing code. The SERIN and SEROUT commands which the BASIC Stamp interpreter supports, provide the end user with an effective and simple method to interface to the MEMKey. The diagram below shows the connection diagram used for this application.



The MEMKey's KEY pin is attached to the Basic Stamp II so that the BSII can tell when a key is pressed. The program below simply sets up the programmable aspects of the MEMKey, sets it up for poll mode and then scans for keys. The in-line comments should make it easy to understand.

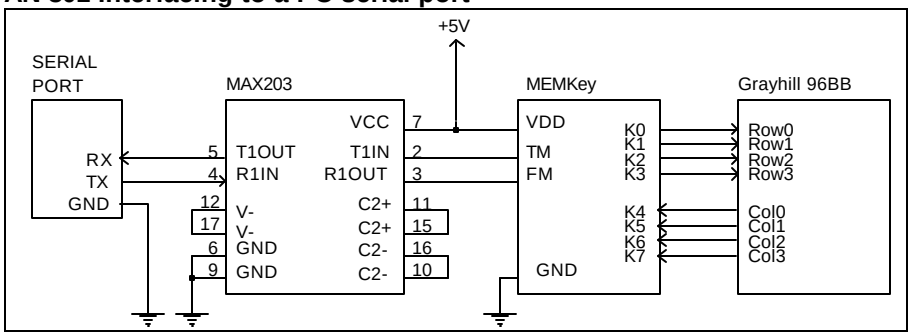
AN-300 Interfacing to a BSII code listing

```
input 15                                    'TM
output 14                                   'FM
input 13                                    'KEY
'
BEGIN:
'
HIGH   14                                   'let noise settle
PAUSE  1000

SEROUT 14,396,[$02,$F0,$0F]                 'program typematic:  delay = 600mS, rate = 37.5mS
PAUSE 10                                    'allow EEPROM to write
SEROUT 14,396,[$03]                         'read typematic
SERIN 15,396,[b0,b1]
DEBUG "delay is:  ",dec b0, cr, "rate is:  ",dec b1,cr,cr
'
SEROUT 14,396,[$04,$10]                     'program debounce:  time = 40mS
PAUSE 10                                    'allow EEPROM to write
SEROUT 14,396,[$05]                         'read debounce
SERIN 15,396,[b0]
DEBUG "debounce is:  ",dec b0,cr,cr
'
SEROUT 14,396,[$08,$18,$AA]                 'program user EEPROM address $18 with $AA
PAUSE 10                                    'allow EEPROM to write
SEROUT 14,396,[$09,$18]                     'read user EEPROM to ensure wrote correctly
SERIN 15,396,[b0]
DEBUG "EEPROM value is:  ",dec b0,cr,cr
'
SEROUT 14,396,[$0A,$00,$40]                 'program serial key 0 to return $40
PAUSE 10                                    'allow EEPROM to write
SEROUT 14,396,[$0A,$01,$41]                 'program serial key 1 to return $41
PAUSE 10                                    'allow EEPROM to write
SEROUT 14,396,[$0A,$02,$42]                 'program serial key 2 to return $42
PAUSE 10                                    'allow EEPROM to write
SEROUT 14,396,[$0A,$03,$43]                 'program serial key 3 to return $43
PAUSE 10                                    'allow EEPROM to write
SEROUT 14,396,[$0A,$04,$44]                 'program serial key 4 to return $44
PAUSE 10                                    'allow EEPROM to write
```

```
SEROUT 14,396,[$0A,$05,$45]          'program serial key 5 to return $45
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$06,$46]          'program serial key 6 to return $46
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$07,$47]          'program serial key 7 to return $47
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$08,$48]          'program serial key 8 to return $48
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$09,$49]          'program serial key 9 to return $49
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$0A,$4A]          'program serial key 10 to return $4A
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$0B,$4B]          'program serial key 11 to return $4B
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$0C,$4C]          'program serial key 12 to return $4C
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$0D,$4D]          'program serial key 13 to return $4D
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$0E,$4E]          'program serial key 14 to return $4E
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$0F,$4F]          'program serial key 15 to return $4F
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$10,$50]          'program serial key 16 to return $50
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$11,$51]          'program serial key 17 to return $51
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$12,$52]          'program serial key 18 to return $52
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0A,$13,$53]          'program serial key 19 to return $53
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0B,$13]              'read serial key 19
SERIN 15,396,[b0]
DEBUG "Key 19 is:  ",dec b0,cr,cr
'
SEROUT 14,396,[$0E,$00]              'program config byte: poll mode, typematic off
PAUSE 10                             'allow EEPROM to write
SEROUT 14,396,[$0F]                  'read configuration byte
SERIN 15,396,[b0]
DEBUG "Config byte is:  ",bin b0,cr,cr
'
SEROUT 14,396,[$10]                  'test mode
SERIN 15,396,[b0]
DEBUG "Firmare version is:  ",dec b0,cr,cr
'
KeyScanStart
DEBUG "Press any key",cr,cr
KeyScanLoop:
SEROUT 14,396,[$01]                  'read # of keys
SERIN 15,396,[b0]
If b0 = 0 Then KeyScanLoop           'if no keys in buffer keep looking
'
ReadKeys:                            'keys in buffer so read buffer
SEROUT 14,396,[$00]
SERIN 15,396,[b0]                    'get key value
DEBUG "Keypress is:  ",dec b0,cr,cr
GOTO KeyScanStart
'
END
```

AN-301 Interfacing to a PC serial port



Serial Port Level Conversion

The MEMKey can be interfaced to a PC serial port with a minimum of effort and difficulty. Perhaps the biggest obstacle is to convert from the TTL level true data format of the MEMKey to the RS-232 level inverted data of the PC serial port. The easiest way to do this is with a level conversion chip, such as those produced by Maxim, Linear Technologies, and others. The diagram above shows a circuit to do the level conversion.

For a standard PC serial port use the following pin out for the serial connector. While these connections should work with most computers, you should verify this for your PC.

Signal Name	DB-25	DB-9
TX	2	3
RX	3	2
GND	7	5

Serial Port Pin Outs

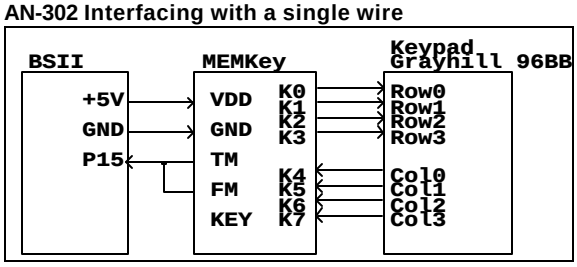
A short Q-BASIC program is provided below which resets the MEMKey to its default state, turns off typematic and automatic mode, and then queries the MEMKey for key presses. A complete menu-driven Q-BASIC program which demonstrates all of the MEMKey commands can be downloaded from the Solutions Cubed web site (www.solutions-cubed.com).

AN-301 Interfacing to a PC serial port code listing

```
OPEN "COM1:2400,N,8,1,CD0,CS0,DS0,OP0,TB2048,RB2048" FOR RANDOM AS #1
    CLS
'Reset MEMKey to Default Values...
    PRINT #1, CHR$(17);                    'Send Default Reset Command
    PRINT "The Default Reset Command Has Been Sent...": PRINT
'Provide delay for write to EEPROM
    FOR i = 1 TO 100000: NEXT
'Disable Typematic and auto-return of keys
    PRINT #1, CHR$(14) + CHR$(0);                    'Program Config Byte
    PRINT "Typematic And Auto-Return have been disabled...": PRINT
    FOR i = 1 TO 100000: NEXT                        'Provide delay for write to EEPROM
'Determine Function
Determine:
    INPUT "Read the key buffer [R] or program a key value [P] or Quit [Q]"; Funct$
      IF Funct$ = "R" OR Funct$ = "r" THEN GOTO ReadKeyBuffer
      IF Funct$ = "P" OR Funct$ = "p" THEN GOTO RetryProgram
      IF Funct$ = "Q" OR Funct$ = "q" THEN GOTO ExitProgram
    GOTO Determine
'Read The Key Buffer
ReadKeyBuffer:
    PRINT #1, CHR$(0);                               'Read Key Buffer
    PRINT : PRINT "The Key Buffer Reads: ";
    FOR delay = 1 TO 20000: NEXT                     'Allow data to be received
```

```
NextBufferByte:                                    'Get Serial Data
        Valid$ = "No"                              'Reset Flags
        MoreData$ = "No"
        IF LOC(1) = 0 THEN GOTO EndGetByte
        A$ = INPUT$(1, #1)                         'Read Data From Buffer
        Byte$ = A$
        Valid$ = "Yes"
        IF LOC(1) <> 0 THEN MoreData$ = "Yes"
EndGetByte:
          IF Valid$ = "Yes" THEN PRINT ASC(Byte$);
          IF Valid$ = "No" THEN PRINT "(No Data Returned)"
    IF MoreData$ = "Yes" THEN GOTO NextBufferByte
    PRINT
    GOTO PressReturn
'Program Serial Key Value
RetryProgram:
INPUT "Enter the Serial Key to Program. (0 - 19) ", SerKey
     IF SerKey < 0 OR SerKey > 19 THEN GOTO RetryProgram
    INPUT "Enter the Data Value to Send. (0 - 255) ", Value
     IF Value < 0 OR Value > 255 THEN GOTO RetryProgram
    PRINT #1, CHR$(10) + CHR$(SerKey) + CHR$(Value);
    FOR i = 1 TO 100000: NEXT                      'Provide delay for write to EEPROM
PressReturn:
    PRINT : PRINT "******** PRESS ANY KEY TO CONTINUE ********"
    DO: LOOP WHILE INKEY$ = "": CLS
    GOTO Determine
ExitProgram:
 CLOSE #1
END
```

AN-302 Interfacing with a single wire



The MEMKey uses a two wire serial interface for communication. However, its TM and FM pins are pseudo open-collector. This coupled with the fact that the FM pin and TM pin never communicate simultaneously (half-duplex) allows the MEMKey to communicate via 1 wire if the master processor is able to support this mode.

In order for 1 wire communication to work, the master processor must be able to both send and receive on one I/O line. Because of this, a standard PC serial port is not able to function in this mode. However, most microcontrollers are able to support this functionality. this example shows how to interface the MEMKey to a Parallax BASIC Stamp II with only one wire.

The diagram shows the connections necessary and the code following uses the default MEMKey settings and merely reads the key presses in a continuous loop.

AN-302 Interfacing with a single wire code listing

```
input 15                                'TO/FROM  Memkey
key         var         byte
'
Begin:
HIGH    15                              'let noise settle
PAUSE   1000
'
SEROUT 15,396,[$11]                     'default reset command
GetKey:
SERIN 15,396,[key]                      'read the key buffer
DEBUG "Keypress is:  ",dec2 key, cr,cr
GOTO    GetKey
'
END
```

Distributors

Digi Key
701 Brooks Ave. South
Thief River Falls, MN 56701-0677
800-344-4539
www.digikey.com

Jameco
1355 Shoreway Road
Belmont, CA 94002-4100
800-831-4242
www.jameco.com

JDR Microdevices
1850 South 10th Street
San Jose, CA 95112-4108
800-538-5000
www.jdr.com

Mondo-tronics
4286 Redwood Hwy #226
San Rafael, CA 94960
415-491-4600
www.mondo.com

Parallax
599 Menlo Drive Suite 100
Rocklin, CA 95765
888-512-1024
www.parallaxinc.com

Disclaimer of Liability and Accuracy: Information provided by Solutions Cubed is believed to be accurate and reliable. However, Solutions Cubed assumes no responsibility for inaccuracies or omissions. Solutions Cubed assumes no responsibility for the use of this information and all use of such information shall be entirely at the user's own risk.

Life Support Policy: Solutions Cubed does not authorize any Solutions Cubed product for use in life support devices and/or systems without express written approval from Solutions Cubed.

Warrantee: Solutions Cubed warrants all Miniature Engineering Modules against defects in materials and workmanship for a period of 90 days. If you discover a defect, we will, at our option, repair or replace your product or refund your purchase price. This warrantee does not cover Miniature Engineering Modules which have been physically abused or misused in any way.