

Keypad A Module

User's Guide

Version: V1.0



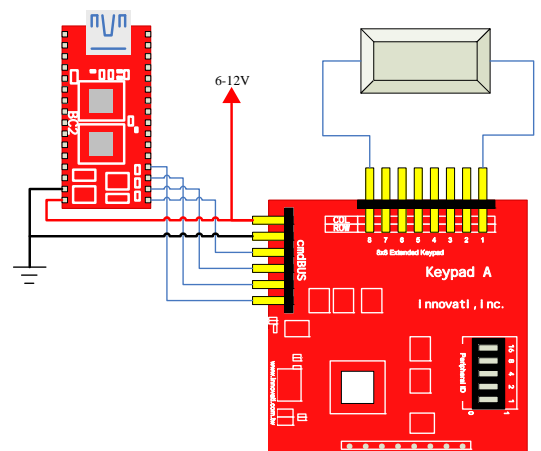
Product Overview: Innovati's 16-key Keypad A Module is designed to provide a range of versatile input functions. It can be directly controlled by Innovati's BASIC Commander for various applications using simple connections. By setting different input modes, it can be converted rapidly to emulate various commonly used input interfaces, such as the common numeric inputs for calculators, English character inputs for mobile phones, hexadecimal inputs for engineering, or even a user-defined returned value for each key. In addition to the input options, the debounce time of the keys can be setup to avoid problems due to mechanical key bounce when the keys are pressed. The automatic repeat (auto-repeat) input can be set to generate a repeat input when a key is pressed and held down. Please use "**KeypadA**" as the module object name in program.

Application:

- When used with an LCD display, it can rapidly emulate the function of a calculator by setting the correct mode.
- Can be used for password input complete with case-sensitive capability.
- The user defined input function allows the user to set and detect different keys through the software to setup the module for versatile operations.
- By setting the hold mode, the keypad can be used as 8-direction keys for a wired remote control.

Product Features:

- 4x4 input keypad. Can be operated in 9 different input modes.
 - Key value mode - default
 - Hexadecimal mode
 - Numeric mode
 - Upper case character input
 - Lower case character input
 - Symbol mode
 - Calculator mode
 - User defined mode
 - Extended keypad input mode
- The user can set the debounce value according to their personal preference to avoid repeat inputs.
- By using the extension pins, the product can be expanded with additional extended keypads.
- Determining the pressed keys can be performed either in the event mode or in the polling

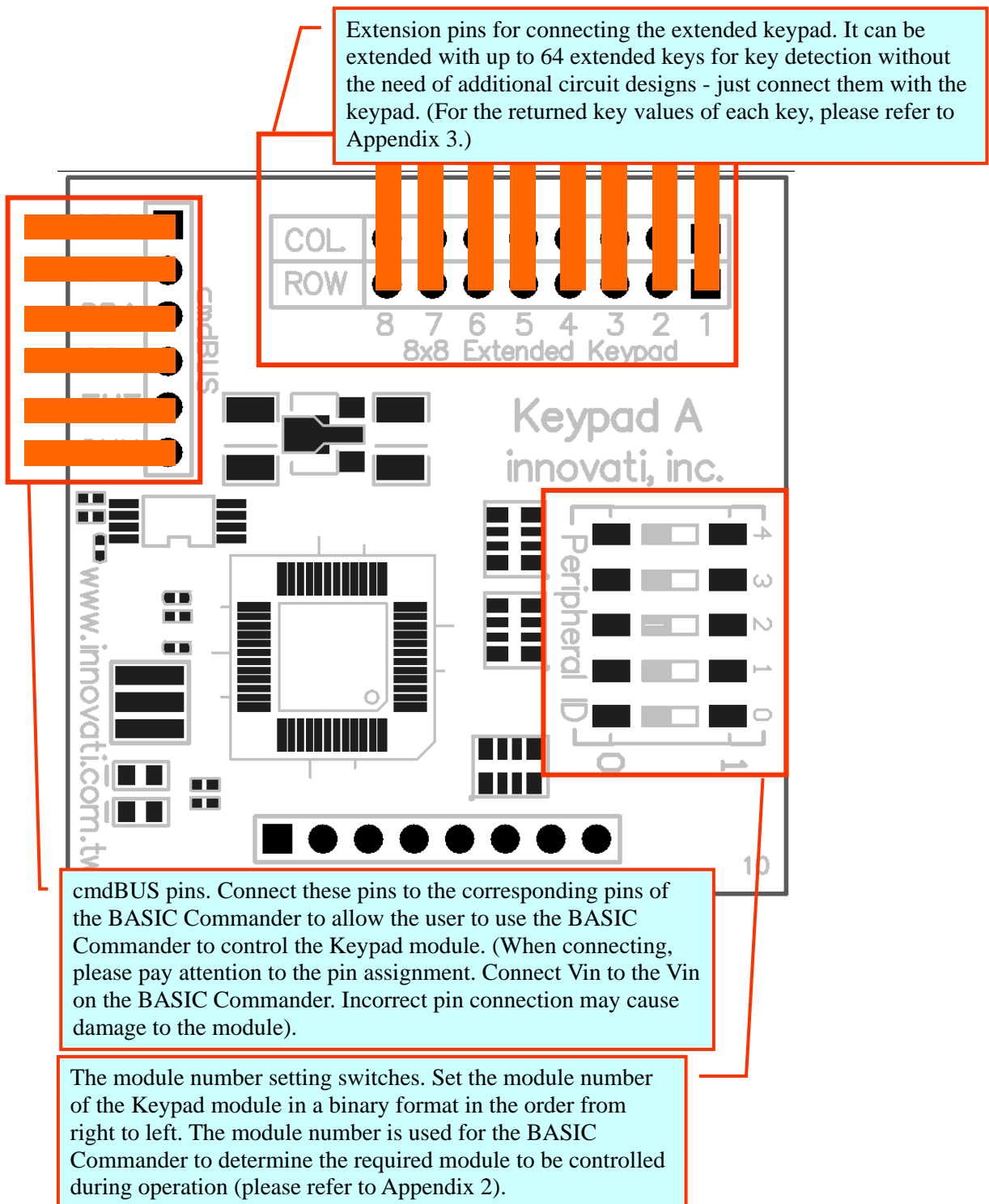


mode though different program designs.

- Pressing and holding the keys can implement a repetition key input.

Connection: Directly setup the ID switches to the required number, and then connect the cmdBUS cable to the corresponding pins on the BASIC Commander as shown in the following figure. Then the required operations can be performed through the BASIC Commander. DC power (6~12V) and ground should be connected to VIN and GND pin. If it is required to expand the system with additional keypads, it can be expanded with up to 64 additional keys just by connecting it to the 8x8 Extended Keypad pins.

Product Specifications:



Operating Temperature : 0 °C~ 70 °C

Storage Temperature : -20 °C~ 80 °C

Commands and Events:

The following tables list all the unique commands and events provided with the Keypad A module. Note that essential words in the commands will be written in **bold** type and *italics* in bold type. The bold type word must be written exactly as shown, whereas the italic bold type words must be replaced with the user values. Note that the innoBASIC language is case-insensitive.

Command Format	Description
ClearKeyBuffer()	Clear all the key press values in the buffer.
DisableBufferFullEvent()	Disable the key buffer full event.
DisableKeypressedEvent()	Disable the key pressed event.
EnableBufferFullEvent()	Enable the key buffer full event.
EnableKeypressedEvent()	Enable the key pressed event
GetCustomTable(<i>CustomTable</i>)	Get the custom key values and store them in the array <i>CustomTable</i> , which should be able to store 16 bytes.
GetCustomTableIndex(<i>Index</i>)	Get the number of the currently used CustomTable and store it in the byte variable <i>Index</i> . (See Note 1.)
GetDebounceTime(<i>Time</i>)	Get the debounce time setting and store it in the byte variable <i>Time</i> .
<i>Status</i> = GetKeyID(<i>KeyID</i>)	Return the keypressed status in the byte variable <i>Status</i> and store the key press value and store it in the byte variable <i>KeyID</i> . If the keypad is not pressed, value 0 will be returned, otherwise value 1 will be returned. When keypad is working in mode 3, 4 or 5 for multiple-character input, if the returned value is 1, it indicates the character stored in the <i>KeyID</i> is the final confirmed character. If the returned value is 2, it indicates the character stored in the <i>KeyID</i> is a transient character. (See Note 1.)
GetKeypadMode(<i>Mode</i>)	Get the current keypad working mode and store it in the byte variable <i>Mode</i> .
GetRepeatRate(<i>Rate</i>)	Get the current repeat rate setting and store it in the byte variable <i>Rate</i> .
GetRepeatTime(<i>Time</i>)	Get the current repeat time setting and store it in the byte variable <i>Time</i> .
LoadCustomTable(<i>Index</i>)	According to the value of the byte variable <i>Index</i> , load the user-defined custom table stored in the EEPROM.
SaveCustomTable(<i>Index</i>)	Store the currently user-defined custom table to the EEPROM at the address specified by the byte variable <i>Index</i> ranging from 0 to 15.
SetCustomTable(<i>KeyID</i>)	Set the elements of array <i>KeyID</i> as the user defined key press values mapped to key 1, 2, 3, A, 4, 5, 6, B, 7, 8, 9, C,

	*, 0, # and D. A total of 16 user-defined key press values should be given in the KeyID array.																																																																								
SetDebounceTime(<i>Time</i>)	Set the debounce time specified by the byte variable Time , the units are 10 ms. The setting command is effective only for externally connected keypads; it will not affect the operation of the built-in keypad.																																																																								
SetKeypadMode(<i>Mode</i>)	<p>Set the keypad mode specified by the byte variable Mode, ranging from 0 to 8 representing one of the following keypad operation modes, which returns the key value as shown below, respectively. (See Note 2.)</p> <p>0: Key value mode (After the key is pressed, it returns the key press value of the Key ID value in order.)</p> <table border="1"> <tr><td>0</td><td>1</td><td>2</td><td>3</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>7</td></tr> <tr><td>8</td><td>9</td><td>10</td><td>11</td></tr> <tr><td>12</td><td>13</td><td>14</td><td>15</td></tr> </table> <p>1: Hexadecimal mode (After the key is pressed, it returns the key press value of 0~F.)</p> <table border="1"> <tr><td>1</td><td>2</td><td>3</td><td>A</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>B</td></tr> <tr><td>7</td><td>8</td><td>9</td><td>C</td></tr> <tr><td>F</td><td>0</td><td>E</td><td>D</td></tr> </table> <p>2: Numeric mode (After the key is pressed, it returns the key press value of the ASCII codes of 0~9.)</p> <table border="1"> <tr><td>1</td><td>2</td><td>3</td><td>(11)</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>(12)</td></tr> <tr><td>7</td><td>8</td><td>9</td><td>(13)</td></tr> <tr><td>*</td><td>0</td><td>#</td><td>(14)</td></tr> </table> <p>3: Upper case character input</p> <table border="1"> <tr><td>!?</td><td>ABC</td><td>DEF</td><td>F1</td></tr> <tr><td>GHI</td><td>JKL</td><td>MNO</td><td>F2</td></tr> <tr><td>PQRS</td><td>TUV</td><td>WXYZ</td><td>F3</td></tr> <tr><td>,</td><td>Space</td><td>.</td><td>F4</td></tr> </table> <p>4: Lower case character input</p> <table border="1"> <tr><td>!?</td><td>Abc</td><td>def</td><td>F1</td></tr> <tr><td>ghi</td><td>Jkl</td><td>mno</td><td>F2</td></tr> </table>	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	1	2	3	A	4	5	6	B	7	8	9	C	F	0	E	D	1	2	3	(11)	4	5	6	(12)	7	8	9	(13)	*	0	#	(14)	!?	ABC	DEF	F1	GHI	JKL	MNO	F2	PQRS	TUV	WXYZ	F3	,	Space	.	F4	!?	Abc	def	F1	ghi	Jkl	mno	F2
0	1	2	3																																																																						
4	5	6	7																																																																						
8	9	10	11																																																																						
12	13	14	15																																																																						
1	2	3	A																																																																						
4	5	6	B																																																																						
7	8	9	C																																																																						
F	0	E	D																																																																						
1	2	3	(11)																																																																						
4	5	6	(12)																																																																						
7	8	9	(13)																																																																						
*	0	#	(14)																																																																						
!?	ABC	DEF	F1																																																																						
GHI	JKL	MNO	F2																																																																						
PQRS	TUV	WXYZ	F3																																																																						
,	Space	.	F4																																																																						
!?	Abc	def	F1																																																																						
ghi	Jkl	mno	F2																																																																						

	<table border="1"> <tr><td>pqrs</td><td>Tuv</td><td>wxyz</td><td>F3</td></tr> <tr><td>,</td><td>Space</td><td>.</td><td>F4</td></tr> </table> <p>5: Symbol mode</p> <table border="1"> <tr><td>!?</td><td>% @\$</td><td>+ - =</td><td>F1</td></tr> <tr><td>/ \ _</td><td>() &</td><td>< > </td><td>F2</td></tr> <tr><td>: ; `</td><td>[] ^</td><td>“ “ “</td><td>F3</td></tr> <tr><td>, *</td><td>Space</td><td>. #</td><td>F4</td></tr> </table> <p>6: Calculator mode</p> <table border="1"> <tr><td>1</td><td>2</td><td>3</td><td>/</td></tr> <tr><td>4</td><td>5</td><td>6</td><td>*</td></tr> <tr><td>7</td><td>8</td><td>9</td><td>-</td></tr> <tr><td>.</td><td>0</td><td>=</td><td>+</td></tr> </table> <p>7: User defined mode</p> <table border="1"> <tr><td>USER 0</td><td>USER 1</td><td>USER 2</td><td>USER 3</td></tr> <tr><td>USER 4</td><td>USER 5</td><td>USER 6</td><td>USER 7</td></tr> <tr><td>USER 8</td><td>USER 9</td><td>USER 10</td><td>USER 11</td></tr> <tr><td>USER 12</td><td>USER 13</td><td>USER 14</td><td>USER 15</td></tr> </table> <p>8: Extended keypad input mode (For the corresponding key ID values of the extended keypads, please refer to Appendix 3.)</p> <table border="1"> <tr><td>240</td><td>241</td><td>242</td><td>243</td></tr> <tr><td>244</td><td>245</td><td>246</td><td>247</td></tr> <tr><td>248</td><td>249</td><td>250</td><td>251</td></tr> <tr><td>252</td><td>253</td><td>254</td><td>255</td></tr> </table>	pqrs	Tuv	wxyz	F3	,	Space	.	F4	!?	% @\$	+ - =	F1	/ \ _	() &	< >	F2	: ; `	[] ^	“ “ “	F3	, *	Space	. #	F4	1	2	3	/	4	5	6	*	7	8	9	-	.	0	=	+	USER 0	USER 1	USER 2	USER 3	USER 4	USER 5	USER 6	USER 7	USER 8	USER 9	USER 10	USER 11	USER 12	USER 13	USER 14	USER 15	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
pqrs	Tuv	wxyz	F3																																																																						
,	Space	.	F4																																																																						
!?	% @\$	+ - =	F1																																																																						
/ \ _	() &	< >	F2																																																																						
: ; `	[] ^	“ “ “	F3																																																																						
, *	Space	. #	F4																																																																						
1	2	3	/																																																																						
4	5	6	*																																																																						
7	8	9	-																																																																						
.	0	=	+																																																																						
USER 0	USER 1	USER 2	USER 3																																																																						
USER 4	USER 5	USER 6	USER 7																																																																						
USER 8	USER 9	USER 10	USER 11																																																																						
USER 12	USER 13	USER 14	USER 15																																																																						
240	241	242	243																																																																						
244	245	246	247																																																																						
248	249	250	251																																																																						
252	253	254	255																																																																						
SetRepeatRate(<i>Rate</i>)	Set the repeat rate specified by the byte variable <i>Rate</i> for determining the repeated triggering rate. The unit is 10 ms. (See Note 3.)																																																																								
SetRepeatTime(<i>Time</i>)	Set the repeat time specified by the byte variable <i>Time</i> for determining the time before the repeated triggering starts. The unit is 10 ms. (See Note 3.)																																																																								

Note 1:
If the “Repeat” related settings are not configured, after the key is pressed and GetKeyID is executed, the status will be reset to 0, even the key is still held down. If there are several keys being pressed continuously and GetKeyID is executed, all the key press values will be stored. When the number of key presses exceeds the capacity of the buffer (32 key press values), the

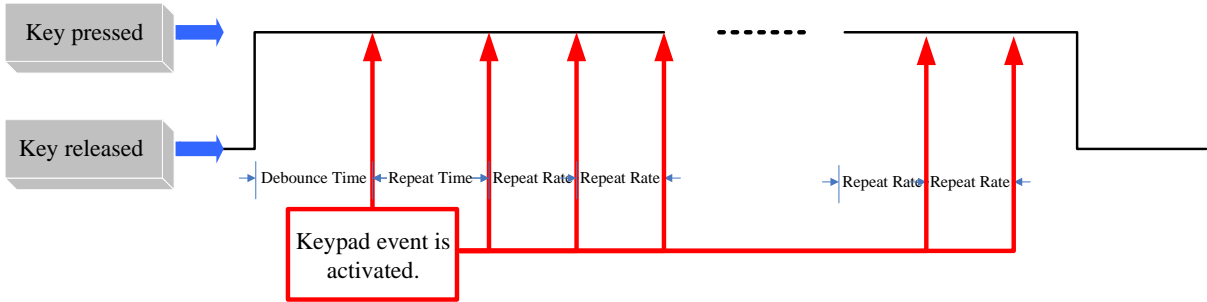
follow-up key presses will be ignored.

Note 2:

When keypad is working in mode 3, 4 or 5 for multiple-character input, if the returned value is 1, it indicates the character stored in the module is the final confirmed character. If the returned value is 2, it indicates the character stored in the module is a transient character.

Note 3:

The effects of setting the Repeat Time and the Repeat Rate are as follows.



After the key is pressed and held for a time interval specified by the Repeat Time, another keypressed event will be activated. After this, the same event will be activated repeatedly for every subsequent time interval specified by the Repeat Rate until the key is released.

Event	Description
KeyBufferFullEvent()	After EnableBufferFullEvent() is executed, if the key pressed exceeds 32, the capacity of the buffer, the KeyBufferFullEvent() will be triggered.
KeyPressedEvent()	After EnableKeyPressedEvent() is executed, if any key is pressed, the KeyPressedEvent() will be triggered.

Example Program:

```

Peripheral myKeypad As KeypadA @ 0 'Set module number to 0

Dim PressKeyD As Byte 'Store the event whether D is pressed
Dim KeyStatus As Byte 'Determine whether a key press value is obtained
Dim KeyID As Byte 'Store the obtained key press value
Dim RepeatTime As Byte 'Store the obtained value of repetition time
Dim RepeatRate As Byte 'Store the obtained value of repetition rate
Dim DebounceTime As Byte 'Store the obtained value of the debounce time
Dim RepeatCount As Byte 'Store the number of repeated key presses.
Dim CustomTable(15) As Byte 'Array for storing the user defined key press values
Dim i As Byte 'Store the loop counts
    
```

Sub Main()

Debug CLS

KEYID_CHECK:

'The following loop will be repeatedly executed. According to the pressed key determined by KeyStatus, the input key press messages will be shown in the Terminal Window.

'If the pressed key is determined to be "D", the loop will be exited.

'This part is used for determining the key press value by using Event.

myKeypad.SetKeypadmode(0) 'Set the keypad mode to 0, the default mode

Pause 100

myKeypad.GetKeyID(KeyID) 'Get the initial key status

RepeatCount=0

KeyStatus=0

myKeypad.EnableKeyPressedEvent() 'Enable the keypad event

Debug "Press D to exit the loop.", CR

PressKeyD=0

Do

Loop Until PressKeyD>0

REPEAT_CHECK:

myKeypad.GetRepeatTime(RepeatTime) 'Get the initial RepeatTime set in the system

myKeypad.GetRepeatRate(RepeatRate) 'Get the initial RepeatRate set in the system

'Display the initial Repeat Time set in the system

Debug "Repeat Time is currently set as ", RepeatTime, " * 10 ms...", CR

'Display the initial Repeat Rate set in the system

Debug "Repeat Rate is currently set as ", RepeatRate, " * 10 ms...", CR

myKeypad.SetRepeatTime(50) 'Set the RepeatTime as 500 ms

myKeypad.SetRepeatRate(2) 'Set the RepeatRate as 20 ms

RepeatCount=1

'The following loop will be executed repeatedly. According to the pressed key determined by KeyStatus.

'If the key remains pressed and held for over half a second, the repeated keypad events will be activated according to the RepeatTime setting.

'The keypad event will be activated repeatedly every 20 ms according to the RepeatRate setting.

Debug "Please press and hold any key", CR

Do

Loop Until RepeatCount>100

myKeypad.DisableKeyPressedEvent() 'Disable the keypad event

'Set RepeatTime and RepeatRate as 0 to disable the repeated keypad event while pressing and holding the key

```
myKeypad.SetRepeatTime(0)
```

```
myKeypad.SetRepeatRate(0)
```

```
CUSTOM_TABLE:
```

```
For i=0 To 15
```

```
    CustomTable(i)=100+i
```

```
Next
```

```
myKeypad.SetCustomTable(CustomTable) 'Set the user defined key press values as 100~115
```

```
myKeypad.SaveCustomTable(0)         'Store the user defined key press values in Table 0
```

```
myKeypad.SetKeypadmode(7)           'Set the keypad mode to 7, the user defined mode
```

```
Pause 100
```

```
myKeypad.GetKeyID(KeyID)
```

```
KeyStatus=0
```

'The following loop will be executed repeatedly. According to the pressed key determined by KeyStatus,

'The input key press messages will be shown in the Terminal Window.

'If the pressed key is determined to be "D", the loop will be exited.

'It can be observed that the returned Key ID will become the user defined key press value

'Here, the status and key value of the pressed key is obtained by polling. There is no keypad event.

```
Debug "Press D to exit the loop", CR
```

```
Do
```

```
    If myKeypad.GetKeyID(KeyID)<>0 Then
```

```
        Select Case KeyID
```

```
            Case 100 : Debug "Press the button 1! (The returned value is 100)", CR
```

```
            Case 101 : Debug "Press the button 2! (The returned value is 101)", CR
```

```
            Case 102 : Debug "Press the button 3! (The returned value is 102)", CR
```

```
            Case 103 : Debug "Press the button A! (The returned value is 103)", CR
```

```
            Case 104 : Debug "Press the button 4! (The returned value is 104)", CR
```

```
            Case 105 : Debug "Press the button 5! (The returned value is 105)", CR
```

```
            Case 106 : Debug "Press the button 6! (The returned value is 106)", CR
```

```
            Case 107 : Debug "Press the button B! (The returned value is 107)", CR
```

```
            Case 108 : Debug "Press the button 7! (The returned value is 108)", CR
```

```
            Case 109 : Debug "Press the button 8! (The returned value is 109)", CR
```

```
            Case 110 : Debug "Press the button 9! (The returned value is 110)", CR
```

```
            Case 111 : Debug "Press the button C! (The returned value is 111)", CR
```

```
            Case 112 : Debug "Press the button *! (The returned value is 112)", CR
```

```
            Case 113 : Debug "Press the button 0! (The returned value is 113)", CR
```

```
            Case 114 : Debug "Press the button #! (The returned value is 114)", CR
```

```
            Case 115 : Debug "Press the button D! (The returned value is 115)", CR
```

```
        End Select
```

```
        If KeyID=115 Then
```

```
            Goto KEYID_CHECK
```

```
        End If
```

```

End If
Loop
End Sub

Event myKeypad.KeyPressedEvent()
KeyStatus=myKeypad.GetKeyID(KeyID)    'Store the obtained key press value in the parameter KeyID
If RepeatCount>100 Then
    Return
Elseif RepeatCount>0 Then
    RepeatCount+=1
    Debug "Count the number of key presses ", RepeatCount, CR
Elseif RepeatCount=0 Then
    Select Case KeyID
        Case 0: Debug "Press the button 1!", CR
        Case 1: Debug "Press the button 2!", CR
        Case 2: Debug "Press the button 3!", CR
        Case 3: Debug "Press the button A!", CR
        Case 4: Debug "Press the button 4!", CR
        Case 5: Debug "Press the button 5!", CR
        Case 6: Debug "Press the button 6!", CR
        Case 7: Debug "Press the button B!", CR
        Case 8: Debug "Press the button 7!", CR
        Case 9: Debug "Press the button 8!", CR
        Case 10: Debug "Press the button 9!", CR
        Case 11: Debug "Press the button C!", CR
        Case 12: Debug "Press the button *!", CR
        Case 13: Debug "Press the button 0!", CR
        Case 14: Debug "Press the button #!", CR
        Case 15: Debug "Press the button D!", CR : PressKeyD=1
    End Select
End If
End Event













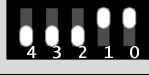



















```

Appendix

1. Known Problems:

- While switching the mode, the determination of events will be reset. If the mode switching operation is performed while a key is being pressed and held, even if the “Repeat” related settings are disabled, a keypad event will still be activated.

2. Module numbers and switch tables:

	0		8		16		24
	1		9		17		25
	2		10		18		26
	3		11		19		27
	4		12		20		28
	5		13		21		29
	6		14		22		30
	7		15		23		31

3. Mode 8 extended keypad returned key press values:

	ROW 1	ROW 2	ROW 3	ROW 4	ROW 5	ROW 6	ROW 7	ROW 8
COL 1	0	1	2	3	4	5	6	7
COL 2	8	9	10	11	12	13	14	15
COL 3	16	17	18	19	20	21	22	23
COL 4	24	25	26	27	28	29	30	31
COL 5	32	33	34	35	36	37	38	39
COL 6	40	41	42	43	44	45	46	47
COL 7	48	49	50	51	52	53	54	55
COL 8	56	57	58	59	60	61	62	63