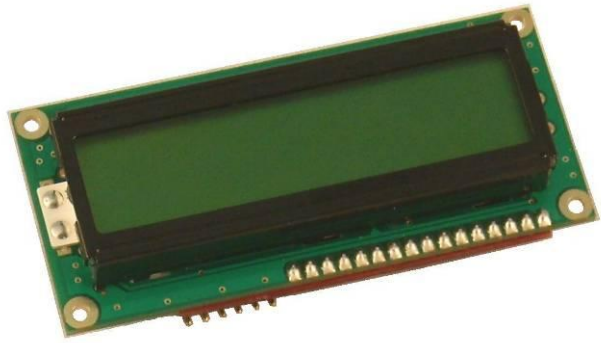


# LCD 2x16A

## 2x16 字元 LCD 顯示模組

版本: V2.0



**產品介紹:** 利基 LCD 2x16A 模組提供多樣化顯示功能，並且可透過簡單的聯接，直接由利基之 Arminno 操控各項應用。在此模組上可同時顯示兩行訊息，各十六字元，另外透過游標控制指令，可隨時變更任意位置的顯示字元。此模組有背光功能，藉由點亮背光，可以讓訊息更容易讀取。另外也可以透過自訂字元，顯示自己所想要的特殊字型。

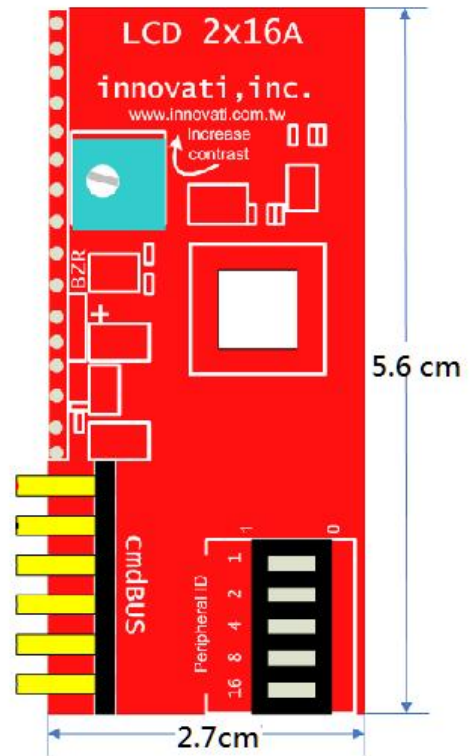
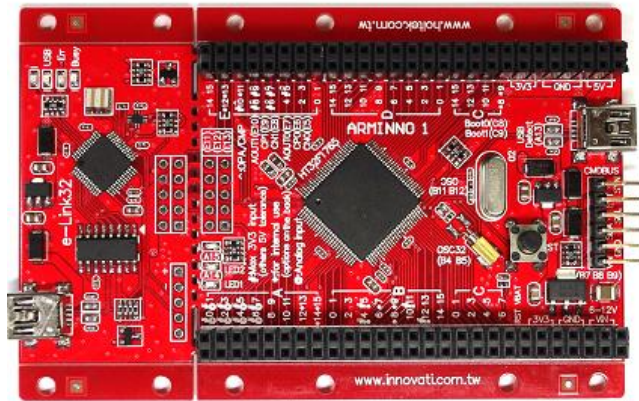
### 應用方向:

- 可加上 RTC 模組即時顯示時間，就是簡單的電子時鐘。
- 於各種應用中即時顯示操作狀態。
- 不經由 PC 直接將錯誤狀態或錯誤訊息顯示於螢幕。
- 藉由自訂字元創造特殊圖案，提供創意訊息。

### 產品特色:

- 可透過輸入 ASCII 碼顯示對應字元。
- 直接使用顯示指令，模組將自動轉換，根據字串或是常數輸入，轉為對應的字元或數字顯示。
- 透過設定，背光可提供 255 段多種亮度顯示。
- 連續輸入時，模組會直接換行顯示，並自動覆蓋原本顯示訊息。
- 各種移動游標顯示方式，可以直接設定游標位址，任意跳行或跳列顯示。當不確定游標位置時，直接輸入 Home 指令，就會回到畫面起始點。
- 多樣化清除螢幕指令，可設定全螢幕清除，往前清除單一字元，自游標清除至列尾，或是由游標處清除到螢幕尾端。
- 不使用時可單獨執行關閉螢幕指令，節省耗電。
- 可透過 cmdBUS 方式，下達指令。

連接方式: 直接將 ID 開關撥至欲設定的編號，再將 cmdBUS 連接至 Arminno 上對應的腳位(如下)，就可透過 Arminno 執行操作。(Vin 與 GND 請與提供 6~12V 之電源與地端連接)



產品規格:

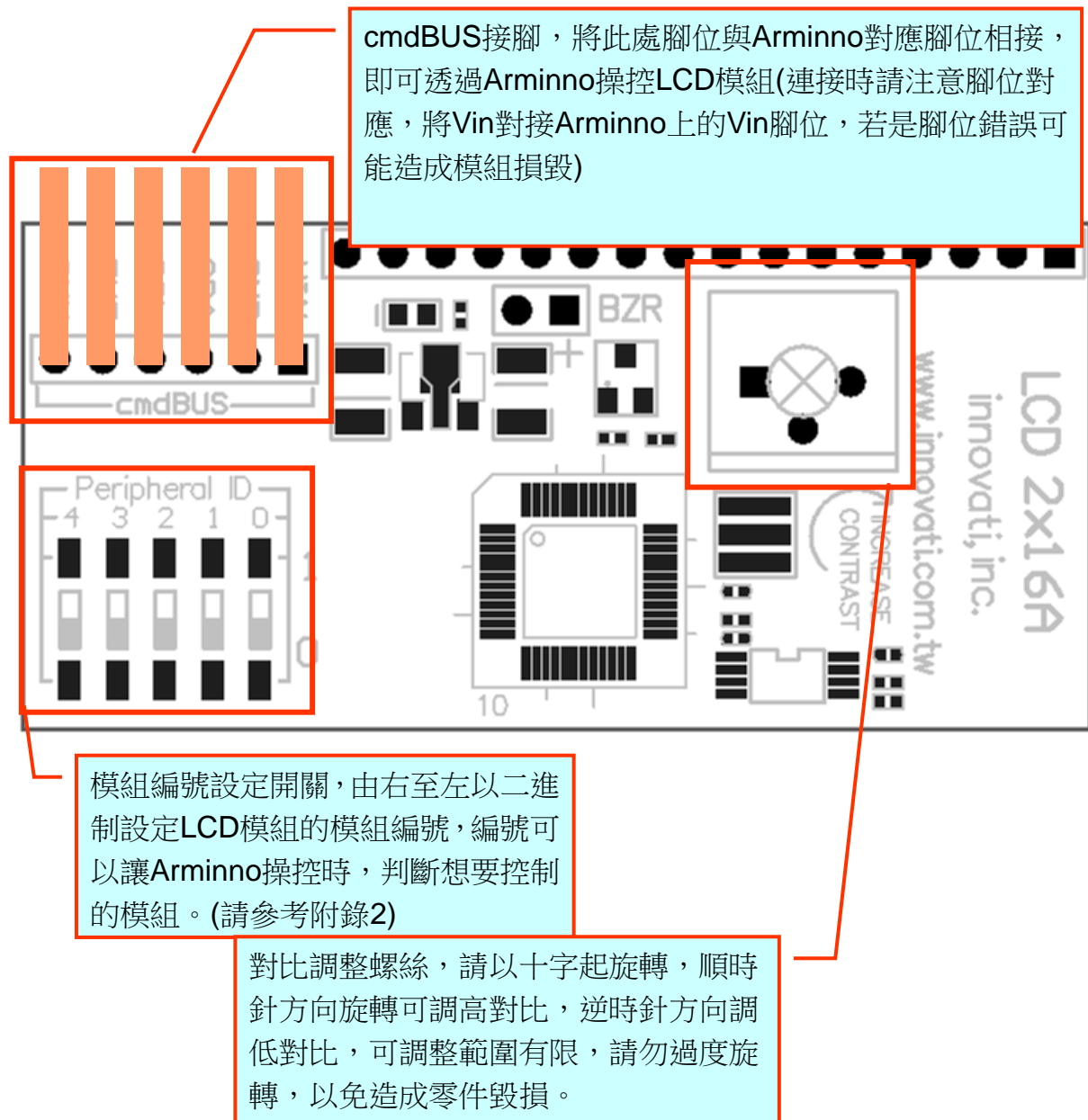


圖 1: 模組腳位與開關介紹

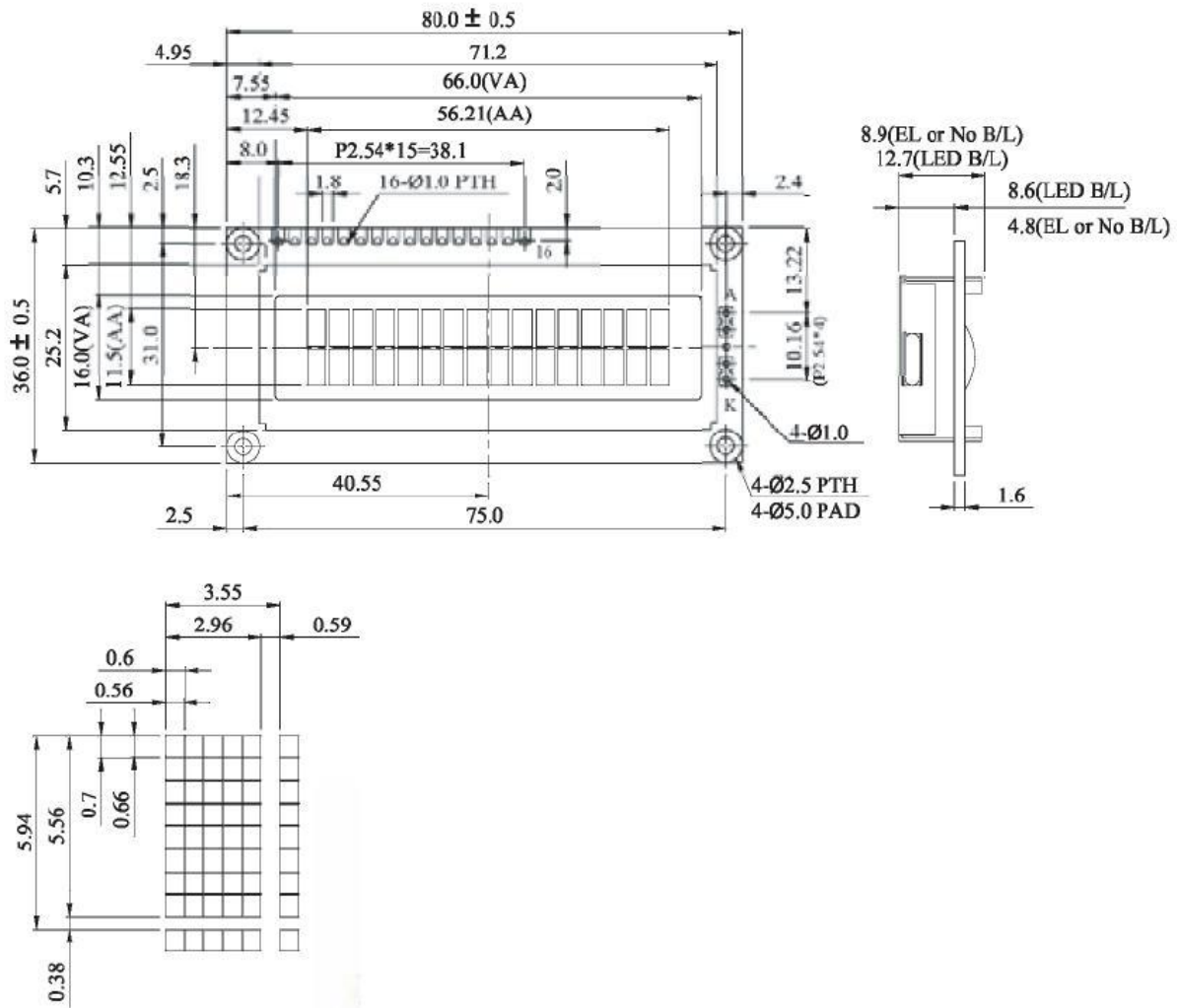


圖 2: LCD 螢幕尺寸規格 (單位 mm)

Item	Standard Value	Unit
Display type	16 characters x 2 Lines	—
Module dimension ( L x W x H )	80.0 x 36.0 x 12.7 (Max) - LED array B/L STN Positive / 6 o'clock / Transflective	mm
Viewing Area	66.0 x 16.0	mm
Active Area	56.21 x 11.5	mm
Dot Size	0.56 x 0.66	mm
Dot Pitch	0.60 x 0.70	mm
Character size ( L x W )	2.96 x 5.56	mm
Character pitch ( L x W )	3.55 x 5.94	mm

表 1: LCD 機構尺寸

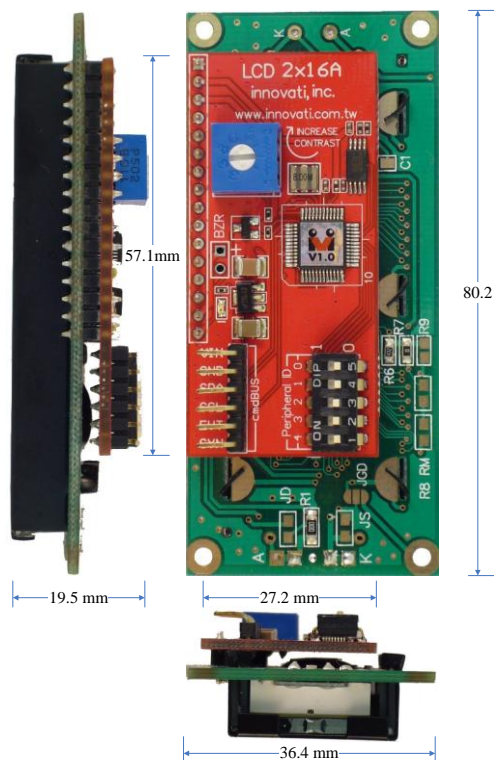


圖 3: LCD 機構尺寸

Symbol	Parameter	Test Conditions		Min.	Typ.	Max.	Unit
		V <sub>DD</sub>	Conditions				
I <sub>DD</sub>	Operating Current	7.5	Backlight On	—	180	—	mA
			Backlight Off	—	5	—	mA

表 1: 工作電流特性 (於 25 °C 之環境)

操作注意事項:

操作溫度 0 °C ~ 70 °C

儲存溫度 -30 °C ~ 80 °C

Item	Symbol	Condition	Min.	Typ.	Max.	Unit
View Angle	(V) $\theta$	CR $\geq$ 2	10		45	deg
	(H) $\varphi$	CR $\geq$ 2	-30		30	deg
Contrast Ratio	CR	—		3		—
Response Time 25°C	T rise	—		100	150	ms
	T fall	—		150	200	ms

表 2: LCD 視角與對比

指令格式	指令功能敘述
移動游標相關指令	
<b>CursorRC(uint8_t Row, uint8_t Col)</b>	將游標移動到 <b>Row</b> 所指定的列與 <b>Col</b> 所指定的行， <b>Row</b> 請輸入 1 或 2， <b>Col</b> 請輸入 1~16 之間的整數值
清除顯示相關指令	
<b>BackSpace(void)</b>	將游標往前移動一個字元，並清除在此位置上顯示的字元
<b>Clear(void)</b>	清除畫面上所有顯示的字元
<b>ClearEOL(void)</b>	清除由游標所在位置開始，到列尾的所有字元
<b>ClearEOS(void)</b>	清除由游標所在位置開始，到螢幕最後顯示的所有字元
顯示字元相關指令	
<b>Display(Parameter)</b>	根據 <b>Parameter</b> 參數形式，如果是 <b>char*</b> 會直接顯示字串，如果是浮點數，會用浮點數表示，其它數值則以十進制顯示
<b>DisplayBin(Value)</b>	將 <b>Value</b> 以二進制顯示， <b>Value</b> 請輸入整數值
<b>DisplayChar(int8_t Chr)</b>	顯示 <b>Chr</b> 所設定的字元， <b>Chr</b> 請輸入 0~255 之間的整數值，也可以輸入 0~7 顯示所代表的自訂字元(可重複輸入多項字元與參數)，輸入值將以 ASCII 碼代表值顯示，請參照附錄 3
<b>DisplayHex(Value)</b>	將 <b>Value</b> 以十六進制顯示， <b>Value</b> 請輸入整數值
各種設定相關指令	
<b>BacklightOff(void)</b>	關閉背光
<b>BacklightOn(uint8_t Time)</b>	以 <b>Time</b> 值設定背光要點亮的時間，若設為 0 則恆亮， <b>Time</b> 請輸入 0~255 之間的整數值
<b>CursorBlinkOff(void)</b>	停止游標閃爍
<b>CursorBlinkOn(void)</b>	讓游標開始閃爍
<b>CursorOff(void)</b>	關閉游標顯示
<b>CursorOn(void)</b>	讓游標顯示於螢幕
<b>DisplayOff(void)</b>	關閉螢幕顯示
<b>DisplayOn(void)</b>	開啟螢幕顯示
<b>RotateLeft(uint8_t Line, uint8_t Spd)</b>	將 <b>Line</b> 所指定的列，各字元不斷向左移動，到最左端的字元會再由右方顯示，移動的速度由 <b>Spd</b> 值決定，越小則速度越快， <b>Line</b> 請輸入 1 或 2， <b>Spd</b> 請輸入 0~255 之間的整數值
<b>RotateOff(void)</b>	停止各行自動向左右移動的動作

<b>RotateRight (uint8_t <i>Line</i>, uint8_t <i>Spd</i>)</b>	將 <b><i>Line</i></b> 所指定的列，各字元不斷向右移動，到最右端的字元會再由左方顯示，移動的速度由 <b><i>Spd</i></b> 值決定，越小則速度越快， <b><i>Line</i></b> 請輸入 1 或 2， <b><i>Spd</i></b> 請輸入 0~255 之間的整數值
<b>SetBacklight (uint8_t <i>Arg</i>)</b>	以 <b><i>Arg</i></b> 設定背光亮度， <b><i>Arg</i></b> 請輸入 0~255 之間的整數值

範例程式：

```
#include "arminno.h"
```

```
LCD2X16A myLCD (0);          //    設定模組編號為 0
```

```
unsigned char byte0 = 12;
```

```
unsigned short word0 = 33;
```

```
float float0 = 1.23;
```

```
int main(void)
```

```
{
```

```
    myLCD.BacklightOn(0);
```

```
    myLCD.Clear();
```

```
    myLCD.Display("Hello");
```

```
    Pause(20000);
```

```
    myLCD.Display(byte0);
```

```
    Pause(20000);
```

```
    myLCD.Display(word0);
```

```
    Pause(20000);
```

































```
    myLCD.Display(float0, 2);
```

```
}
```

## 附錄

1. 已知問題:

2. 模組編號開關對應編號表:

	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. ASCII 表:

- American Standard Code for Information Interchange，美國信息交換標準代碼，是基於拉丁字母的一套電腦編碼系統，此處的 ASCII 碼是根據標準編碼略做調整得到，由使用者輸入的編號轉換為相對應的字元。
- 左方欄位表示的是二進制的低四位元，上方欄位表示的是二進制的高四位元。欄位中的 L 代表 0，H 代表 1，LLLL 就是二進制的 0000，十進制即為 0。
- 最左上方的表格代表，輸入 ASCII 碼 0 時，LCD 會顯示的字元圖案(CG RAM1 是會輸出使用者所設定的自訂字元 1)，往下依序遞增，到 H 所代表的 ASCII 碼輸入值為 16，依此類推，最右下的字元是輸入 255 所得到的顯示畫面。

Upper 4 bit Lower 4 bit	LLLL	LLHH	LLHL	LLHH	LHLL	LHLH	LHHL	LHHH	HLLL	HLLH	HLHL	HLHH	HHLL	HHHL	HHHH
LLLL	CG RAM (1)	CG RAM (2)	CG RAM (3)	CG RAM (4)	CG RAM (5)	CG RAM (6)	CG RAM (7)	CG RAM (8)	CG RAM (9)	CG RAM (10)	CG RAM (11)	CG RAM (12)	CG RAM (13)	CG RAM (14)	CG RAM (15)
LLHH	CG RAM (16)	CG RAM (17)	CG RAM (18)	CG RAM (19)	CG RAM (20)	CG RAM (21)	CG RAM (22)	CG RAM (23)	CG RAM (24)	CG RAM (25)	CG RAM (26)	CG RAM (27)	CG RAM (28)	CG RAM (29)	CG RAM (30)
LLHL	CG RAM (31)	CG RAM (32)	CG RAM (33)	CG RAM (34)	CG RAM (35)	CG RAM (36)	CG RAM (37)	CG RAM (38)	CG RAM (39)	CG RAM (40)	CG RAM (41)	CG RAM (42)	CG RAM (43)	CG RAM (44)	CG RAM (45)
LLHH	CG RAM (46)	CG RAM (47)	CG RAM (48)	CG RAM (49)	CG RAM (50)	CG RAM (51)	CG RAM (52)	CG RAM (53)	CG RAM (54)	CG RAM (55)	CG RAM (56)	CG RAM (57)	CG RAM (58)	CG RAM (59)	CG RAM (60)
LHLL	CG RAM (61)	CG RAM (62)	CG RAM (63)	CG RAM (64)	CG RAM (65)	CG RAM (66)	CG RAM (67)	CG RAM (68)	CG RAM (69)	CG RAM (70)	CG RAM (71)	CG RAM (72)	CG RAM (73)	CG RAM (74)	CG RAM (75)
LHLH	CG RAM (76)	CG RAM (77)	CG RAM (78)	CG RAM (79)	CG RAM (80)	CG RAM (81)	CG RAM (82)	CG RAM (83)	CG RAM (84)	CG RAM (85)	CG RAM (86)	CG RAM (87)	CG RAM (88)	CG RAM (89)	CG RAM (90)
LHHL	CG RAM (91)	CG RAM (92)	CG RAM (93)	CG RAM (94)	CG RAM (95)	CG RAM (96)	CG RAM (97)	CG RAM (98)	CG RAM (99)	CG RAM (100)	CG RAM (101)	CG RAM (102)	CG RAM (103)	CG RAM (104)	CG RAM (105)
LHHH	CG RAM (106)	CG RAM (107)	CG RAM (108)	CG RAM (109)	CG RAM (110)	CG RAM (111)	CG RAM (112)	CG RAM (113)	CG RAM (114)	CG RAM (115)	CG RAM (116)	CG RAM (117)	CG RAM (118)	CG RAM (119)	CG RAM (120)
HLLL	CG RAM (121)	CG RAM (122)	CG RAM (123)	CG RAM (124)	CG RAM (125)	CG RAM (126)	CG RAM (127)	CG RAM (128)	CG RAM (129)	CG RAM (130)	CG RAM (131)	CG RAM (132)	CG RAM (133)	CG RAM (134)	CG RAM (135)
HLLH	CG RAM (136)	CG RAM (137)	CG RAM (138)	CG RAM (139)	CG RAM (140)	CG RAM (141)	CG RAM (142)	CG RAM (143)	CG RAM (144)	CG RAM (145)	CG RAM (146)	CG RAM (147)	CG RAM (148)	CG RAM (149)	CG RAM (150)
HLHL	CG RAM (151)	CG RAM (152)	CG RAM (153)	CG RAM (154)	CG RAM (155)	CG RAM (156)	CG RAM (157)	CG RAM (158)	CG RAM (159)	CG RAM (160)	CG RAM (161)	CG RAM (162)	CG RAM (163)	CG RAM (164)	CG RAM (165)
HLHH	CG RAM (166)	CG RAM (167)	CG RAM (168)	CG RAM (169)	CG RAM (170)	CG RAM (171)	CG RAM (172)	CG RAM (173)	CG RAM (174)	CG RAM (175)	CG RAM (176)	CG RAM (177)	CG RAM (178)	CG RAM (179)	CG RAM (180)
HHLL	CG RAM (181)	CG RAM (182)	CG RAM (183)	CG RAM (184)	CG RAM (185)	CG RAM (186)	CG RAM (187)	CG RAM (188)	CG RAM (189)	CG RAM (190)	CG RAM (191)	CG RAM (192)	CG RAM (193)	CG RAM (194)	CG RAM (195)
HHLH	CG RAM (196)	CG RAM (197)	CG RAM (198)	CG RAM (199)	CG RAM (200)	CG RAM (201)	CG RAM (202)	CG RAM (203)	CG RAM (204)	CG RAM (205)	CG RAM (206)	CG RAM (207)	CG RAM (208)	CG RAM (209)	CG RAM (210)
HHHL	CG RAM (211)	CG RAM (212)	CG RAM (213)	CG RAM (214)	CG RAM (215)	CG RAM (216)	CG RAM (217)	CG RAM (218)	CG RAM (219)	CG RAM (220)	CG RAM (221)	CG RAM (222)	CG RAM (223)	CG RAM (224)	CG RAM (225)
HHHH	CG RAM (226)	CG RAM (227)	CG RAM (228)	CG RAM (229)	CG RAM (230)	CG RAM (231)	CG RAM (232)	CG RAM (233)	CG RAM (234)	CG RAM (235)	CG RAM (236)	CG RAM (237)	CG RAM (238)	CG RAM (239)	CG RAM (240)