# 11

案例研究



閱讀本章後,讀者將:

1. 實驗他關於機電一體化系統設計和實時實施的知識
2. 從A到Z的機電一體化系統設計
3. 能夠執行機電系統設計的不同階段
4. 能夠解決控制問題,建立我們必須即時實施的控制法
5. 能夠使用中斷概念編寫 C 語言的程式,實現即時實現



## 11.1 簡介

在前幾章中,我們開發了一些概念,通過學術示例來說明其應用,向讀者展示結果如何應用。更具體地說,我們已經看到了如何設計機電一體化系統,我們已經提出了我們必須遵循的二ff元步驟,以成功設計所需的機電一體化系統。我們介紹了在設計以下方法時必須使用的方法:

* 機械元件
* 電子電路
* 即時實時實現的 C 中的程式

這些工具被應用於一些實用的系統,並給出了更多的細節,以説明讀者執行自己的設計。

對於控制algorithms,我們介紹的大多數例子都是具有完美模型的學術。不幸的是,對於一個實用的系統,我們將有一個實現,可以描述系統在某些特定條件,由於某些原因,這個模型不會完美地工作,如預期的那樣,實時實現演演演算法。這可能是由忽略的動態引起的ff,這些動態可能會改變某些頻率的行為。

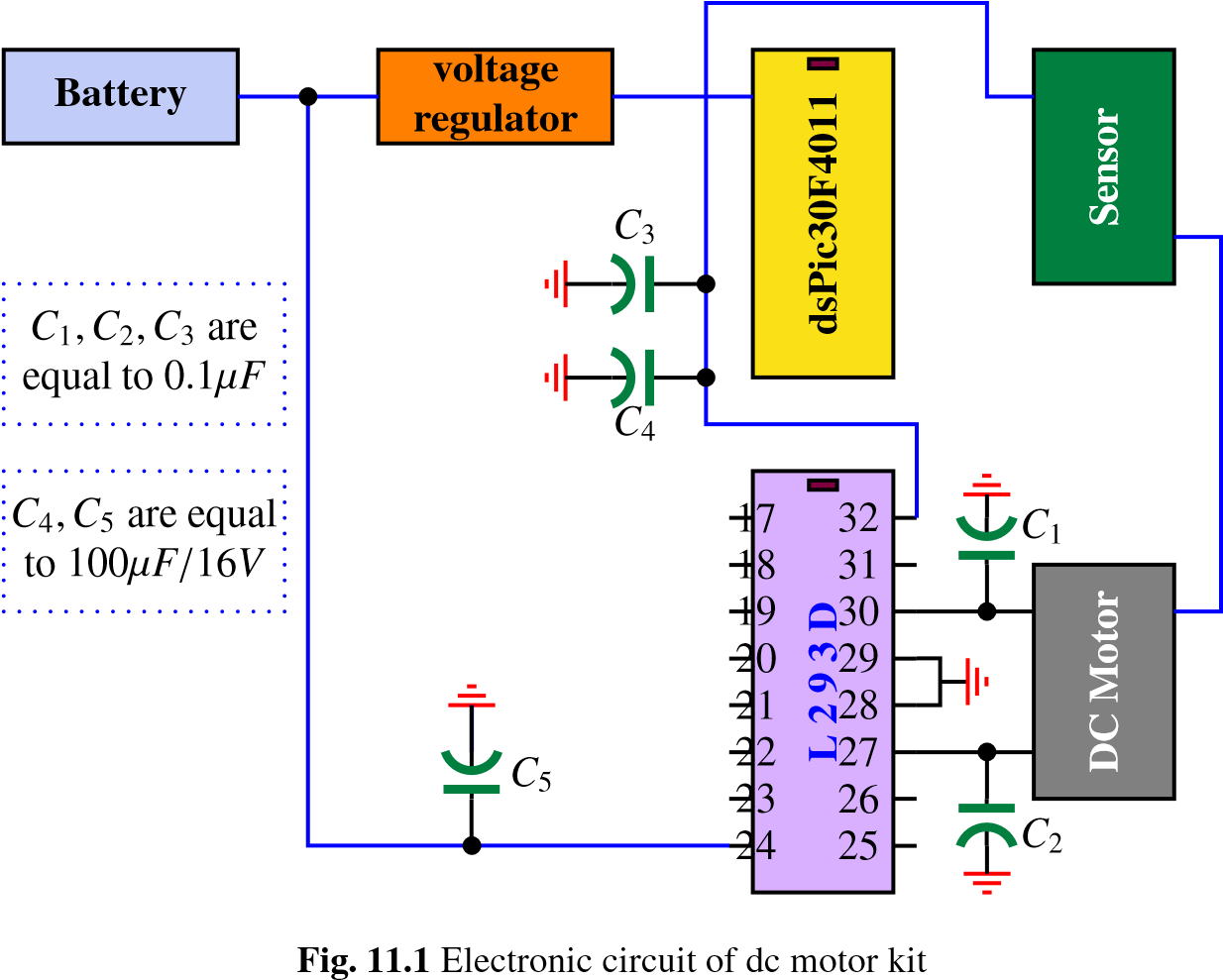
本章的目的是向讀者展示我們如何即時實現我們前面為實踐系統開發的前幾章中開發的理論成果。我們將逐步前進,並展示所有步驟,以方便讀者閱讀。我們在本章中考慮的案例研究是前幾章中討論和設計的案例研究。

## 11.2 直流電機套件的速度控制

作為第一個例子,讓我們考慮直流電機驅動機械部件的速度控制。此示例的 choice 非常重要,因為大多數系統都將使用此類直流電機。我們會考慮的直流電機是由Maxon公司製造的。該電機非常重要,因為它配備了變速箱(比率 6:1)和編碼器,每轉一圈提供一個 hundred 脈衝,每轉 600 個脈衝,我們利用四位法將其帶到每轉 2,400 個脈衝。我們在此示例中使用的系統用於我們前面介紹的 Contr ol 演演演算法的即時實現(如果更靈活且具有ff更多優勢)。

該電機的數據表給出了所有重要的參數,因此可以輕鬆獲得該執行器的傳輸功能。我們在此示例中考慮的負載是一個小磁碟,帶有脫離,我們希望在速度上和以後的位置進行控制。此設置如圖 11.1-(11.2) 所示。我們正在考慮的磁碟直徑等於 0。06 *m* 和品質等於 0。050*公斤*。通過直流電機的數據表之一,可以在磁碟速度和輸入電壓之間獲得傳輸功能。

首先,讓我們專注於負載的速度控制。在這種情況下,為了建立該系統的傳輸功能(直流電機執行器及其負載),我們可以使用數據表、磁碟上的資訊以及 Boukas [1] 中的結果



或繼續識別。使用第一種方法在 [1] 中第 2 章的結果,我們可以得到:

*K*

*G*(*s*) =  =*s* = 1

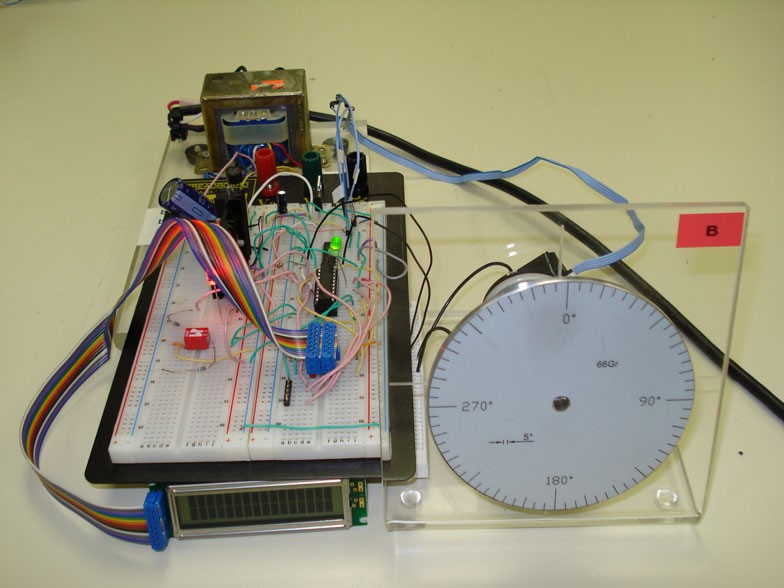
與

*K* = 48。91 × = 63。921 *毫秒*

為了識別我們的系統,我們可以使用即時實現設置和適當的C程序進行即時操作。由於微控制器擁有有限的記憶體,因此標識可以分為兩個步驟。首先,在第一個實驗中確定增益*K,*然後使用此增益計算可用於計算常量時間 \* 的穩定狀態值。

為了設計控制器,我們首先應該指定我們希望我們的系統具有的性能。作為第一個工具,我們需要我們的系統是穩定的。還需要系統速度在瞬態機制上有良好的行為,在穩定狀態制度中零錯誤,以便進行步驟參考。對於瞬態,我們希望負載具有小於或等於 5% 的分載時間  ,而過衝小於或等於 5%。

為了完成適當的控制器的設計,我們可以在連續時間進行設計,然後獲得應在軟體部分程式設計的演演演算法,或者直接在離散時間使用所有設計。在本示例的其餘部分中,我們將選擇第二種方法。



**圖 11.2**實時實現設定

從系統傳輸函數的運算式和所需的性能,它重新蘇爾t,我們至少需要一個比例和整合器 (PI) 控制器。此控制器的傳輸功能由以下人員提供:

*KI*

*C*(*s*) = *KP* |

*s*

*其中KP*和*KII* 是收益決心迫使負載有我們強加的性能。

使用零階持有者和*Z* 轉換表,我們得到:

*Kz*1 = *e*= *T*= 1 = *z*=1

*G*(*z*) |

(*從* +1 起 )*z* = *e*=*T*|

*K* 1 = *e*= *T*|

=

*z* = *e*= *T*|

對於控制器,使用梯形離散化,我們得到:

*U*(*z*) *T z* + 1

*C*(*z*) =  K *K* *P* = *KI*



*E*(*z*) 2 *z* − 1

*KP* = *TK*2 *I* *z* + K*KP* = *TK*2 *I*

=

*z* = 1

將分子和分母除以*z* 並返回時間,我們得到:

*TKI TKI*

*u*(*k*) = *u*(k*=* − 1) = *KP* =  *e*(*k*) = k P   =  *e*=  − 1)

2 2

結合執行器的傳輸功能及其負載,以及控制器之一,得到以下閉環傳輸功能:

*K* 1 + *e*+ *T*= *KP* = *TK*2 *I* *z* = *K* 1 + *e*+*T*= *KP* = *TK*2 *I*

*F*(*z*) = *T T T*

*z*2 + *K KP* = *TK*2 *I* 1 + *e*   − + \* • 1 1 + *e*+ τ *z* + *K* 1 + *e*= τ = K P = *TK*2 *I* = *e*= *T*=

使用現在所需的性能,很容易得出結論,佔主導地位的極點



*s*1,2 = n  = *jwn* !1 = 2

其中α和\_n*n* 分別表示系統控制的閉環的阻尼比和自然頻率。

從控制理論(見Boukas [1])眾所周知,過沖*d%*和沉降時間*t*  s 在5% 是由:

d%

*ts* 

*否。*

我們利用我們的表演和這些表達方式得出結論:

[ = 0。707

5

[*n* =  = 110.6387*rad*/*s*

J

給予以下主導極:

*s*1,2 = 7878。2216 × 78.2452*j*

使用轉換*z* = *eTs* 與 *T* = =  = 0。0064,我們在 *Z*網域中取得以下主導極:

*與*1,2 = 0.5317 × 0.2910*j*

有了這些極點,我們有以下特徵方程:

μ*d* = (*z* = 0。5317 × 0.2910*j*) (*z* = 0.5317 × 0.2910*j*)

\**從*2 + 1。0634*與* + 0.3674

現在使用我們得到的極點放置技術:

1 + *C*(*z*)*G*(*z*) = μ*d*

這意味著:

0.3040

*KI* |

*KT* 1 = *e*= *T*|

•0.4308 × 2*e*×*T*|

*KP* =  *T*  |

使用*K*K、T *T* 和 \*的值,我們得到以下表示式的增益 *KP* 和 *KI:*

*KP* = 0。1480

*KI* = 10。1951年

**備註 11.2.1***在這種情況下必須注意,因為我們不關心傳輸函數零的位置,因此在實現此控制器時,我們可能會有一些意外。很顯然,我們將得到的表現(穩定時間和過沖)將取決於零的位置。有關此事的更多詳情,我們將讀者推薦給布卡斯 [1]。*

為了現在實現這個PI控制演演演算法,並確保所需的性能,我們將使用微控制器從微船 [[1]](#footnote-1) 。這一選擇是由於我們在此類微控制器方面的經驗。讀者可以記住,任何其他來自其他製造商的微控制器與一些小的變化將做的工作。在此示例中,我們將使用微分距的微控制器 dsPIC30F4011。 Microhip.

我們實現的代碼是用*C*C語言編寫的.採用這種語言是為了簡單起見。實作有以下結構:

//

把包括放在這裡

//

#include "p30F4011.h" // proc specific header

//

|  |  |
| --- | --- |
| 定義結構  類型def結構 |  PI 增益 |  |
| 浮K\_P; | 依比例增益 |
| 浮動K\_I; | 積分增益 |

//

PI 常數

//

浮動Const1\_pid;KP = T KI/2 浮Const2\_pid;-KP = T KI/2 浮點參考; // 速度參考

//

系統變數

//

浮動y\_k; // y\_m[k] -> 測量時 k 浮點u\_k的輸出; // u\_k] -> 輸出時間 k 浮點e\_k; // e\_k] -> 錯誤在時間 k

//

系統過去的變數

//

浮u\_prec; // u\_k-1] -> 輸出時 k-1 浮點e\_prec; // e[k-1] -> 錯誤時 k-1

\*結構;

PI.Const1= PI.K\_P\_T\_PI.K\_I/2;PI.Const2\_-PI.K\_P\_T\_PI.K\_I/2;PI.參考=600;

//

功能 // 浮動讀取速度(空隙);浮動計算控制(空);浮動發送控制(無效);

//

此處使用計時器 1 中斷程式(計數器r 計時器 1的溢出)

不合法\_\_ISR \_T1Interrupt (不合法 )//中斷一般代碼

{

中斷服務例程代碼轉到此處浮動Position\_error;

//

讀取速度

PI.y\_m\_閱讀速度();PI.e\_k=PI.參考-PI.y\_m;

//

計算控制項

//

計算錐();

//

傳送控制項

//

發送控制();

|  |  |
| --- | --- |
| IFS0位.T1IF=0;  } | 停用中斷 |
| 主 ( 空白 ) | 主應用程式代碼的開頭 |

{

應用程式代碼在這裡, i;

初始化變數參考和 ThePID.y\_m(可以從輸入讀取)參考 = 0x8000;十六進位數字

(0b...二進位號) PID = 0x8000;

初始化寄存器

TRISC=0x9fff;RC13 與 RC14(接腳 15 與 16)設定為輸出

IEC0位.T1IE=1; // 啟用計時器 1 上的中斷

不確定迴圈,而 (1)

[ ] 傳回 0 }

% 讀取速度功能 int 讀取速度 (空)

{

}

% 計算控制功能 int 計算控制 (虛空)

[PI.u\_k\_pi.u\_prec\_\_thePI.Const1\_e\_k\_\_thePI.Const2\_thePI.e\_prec;[

% 傳送控制功能 int S端控制 (void)

\* 傳送控制()

//

更新過去的資料

//

PI.u\_prec\_PI.u\_k;PI.e\_prec\_PI.e\_k;

}

從這個結構可以看出,首先我們注意到系統將進入迴圈,並在每次中斷函數的調用:

* 讀取速度;
* 計算控制;
* 發送控制;

制定並採取適當的操作。

ReadSpeed 函數在計算控制函數將使用的每個採樣時間返回負載速度。發送控制功能通過 L293D 晶片向執行器發送適當的電壓。

使用編譯器 HighTec C 獲取十六進位碼和 PicKit-2 將檔上傳到微控制器的記憶體中。有關如何獲取十六進位碼的更多詳細資訊,我們邀請讀者閱讀編譯器 HighTec C 或 Microc hip 的編譯器 C30 手冊。

在這種情況下,國家方法微不足道,我們不會發展它。

## 11.3 直流電機套件的位置控制

讓我們專注於負載位置控制。遵循與前一節中開發的負載速度控制類似的步驟,我們首先選擇我們希望我們的系統具有所需的性能。實施以下表演:

* 系統穩定在閉環中;
* 穩定時間*ts* 在 2% 等於我們所能擁有的最佳
* 過沖等於 5%
* 作為輸入的步進函數的穩態等於零

使用性能和傳輸功能,很容易得出比例控制器*KP* 足以滿足這些性能的結論。

在此示例中,我們將使用連續時間方法進行控制器的設計。基於過去一章,我們系統的模型由:

*K*

*G*(*s*) =  *s*(s  = 1)

*其中 K* 和 \* 採用與速度控制相同的值。讓傳輸控制器透過以下操作進行:

*C*(*s*) = *KP*

使用這些運算式,閉環傳輸函數由以下函數給出:

*C*(*s*)*G*(*s*)

*F*(*s*) |

1 = *C*(*s*)*G*(*s*)

*KKP*

= τ

*s*2 × 1\* *s* = *KK*= *P*

由於系統的類型為 1,因此使用比例控制器,導致步進函數作為輸入的錯誤等於零。

根據規格,以下複雜極:



*s*1,,2 = =*wn* = *jwn* !1 = 2

將執行該工作,對應 g特徵方程由以下公式給出:

*s*2 - 2-*wns* - *w*2*n* - 0。

將這與我們得到的閉環系統之一等同:

12 'w*wn'* = 

\**在*2*n* =  *KKP* 。

T

要確定最佳結算時間*ts* 為 2%,請注意,我們有:

4

*ts* =  . =*wn* 現在使用事實:

1 'w*wn'* = 

2\*

我們獲得:

*ts* = 8|

因此,我們在此控制器的 %2 的最佳穩定時間是系統恆定的時間的 8 倍。任何小於將實現的值都可以實現。事實上,如果我們在改變*KP*時查看閉環系統的根位點,這微不足道。This由圖 11.3 給出。要固定控制器的增益,所需的極數*為*11、2  = =7。5= *j,*我們使用圖形並選擇 += 0。707. 這給出 K *KP* = 0。1471年。

+20

+15

+10

#5

0

5

+10

+8

#6

#4

#2

0

2

4

6

8

10

0.99

0.2

0.4

0.58

0.72

0.83

0.91

0.96

0.99

2.5

5

7.5

10

12.5

15

17.5

0.2

0.4

0.58

0.72

0.83

0.91

0.96

根洛庫斯

真實軸

想像軸

**圖 11.3**帶比例控制器的直流電機的根位

使用此控制器,振幅等於 30 度的步進函數的時間回應由圖 11.4 表示,由此我們得出結論,設計控制器滿足所有解壓縮性能,穩定時間為 %2 等於 0。5115 *s*.但是,如果我們實現這個控制器,現實將是從類比,因為齒輪箱的抗雷度不包括在使用的模型,因此在即時的結果,我們將是different和錯誤永遠不會為零。為了克服這個問題,我們可以使用比例控制器和導數控制器,在%2 時提供更好的穩定時間。讓此控制器的傳輸功能由以下人員提供:

*C*(*s*) = *KP* = *KDs*

*其中 KP* 和 *KD* 是要確定的增益。

**備註 11.3.1***請務必注意,比例控制器和導數控制器的使用將在閉環傳輸中引入零,如果放置良好,可能會縮短結算時間。根據 its 位置,過沖和沉降時間將是一個*ff*ect。有關此事的更多詳細信息,我們將讀者推薦至 [1]。*

以 30 度為振幅的步進函數的時間回應

0

0.1

0.2

0.3

0.4

0.5

0.6

0.7

0.8

0.9

1

0

5

10

15

20

25

30

35

以秒(秒)表示的時間

角位置度

**圖 11.4**以 30 度為振幅的步進函數的時間回應

使用此控制器,閉環傳輸功能由以下人員提供:

*C*(*s*)*G*(*s*)

*G*(*s*) |

1 = *C*(*s*)*G*(*s*)

*KKDs*=*KKP*

= τ

*s*2 + 1+*KK*τ *D s* + *KK*τ *P*

與之前一樣,兩個複雜的極點用於控制器的設計。如果我們將兩個特徵方程等同起來,

|  |  |
| --- | --- |
| 2-*wn* | D  T |
| *w*2*n* | *KKP*  = . |

1 = *KK*

T

在這種情況下,我們有兩個未知的*K*變數K *P*和*KD*和兩個代數方程,它們唯一地決定了增益。它們的運算式由:

22*n* *w*

*KP* =  *K*

2 -*wn*

*KD* = 

*K*

使用現在所需的性能,我們得出與之前類似的結論,即等於振幅等於 30 度的步進函數的輸入的穩誤差等於零,與對應於 %5 的過衝的阻尼比等於 0。707.我們可以確定的建立時間*(t*  s 為 %2) 為系統的時間常數的比例,給出:

4 *w* *n*  否  。

|*ts*

現在,如果我們將結算時間固定在 3+,我們會得到:

*w No.* = 29.4985.

使用這些值,我們得到以下值,以便獲得控制器增益:

*KP* = 1。1. 374

*KD* - 0。0545.

要給出以下複雜的極點:

*s*1,2 = 2828。6763 × 6.9163*j*.

零點在:

*z* = =20。8618.

使用此控制器,圖 11.5 中表示振幅等於 30 度的輸入的時間回應。從這個數位可以看出,過沖和沉降時間較少,使用比例控制器獲得。

To 實現比例或比例和導數控制器,我們需要獲得控制定律的循環方程。此,我們需要使用前面預先格式化的 different 方法來離散化控制器的傳輸函數。讓我們使用梯形法,該方法由by *T*2 *z zz*==1 11替換*s。* 這給出了:

比例控制器的C(*z*) = *Kp*

2*的* = 1

*C*(*z*) = 對比例控制器與導數控制器T*z* = 1的*KP* = *KD* +



如果我們用*u*u (*k*) 與 *e*(*k*)表示控制項與參考與輸出之間的錯誤在即時 *kT,*我們得到以下表示式: 1.

*u*(*k*) = *kEe*(*k*)

以 30 度為振幅的步進函數的時間回應

0

0.1

0.2

0.3

0.4

0.5

0.6

0.7

0.8

0.9

1

0

5

10

15

20

25

30

35

以秒(秒)表示的時間

角位置度

**圖 11.5**以 30 度為振幅的步進函數的時間回應

2. 對於比例控制器和導數控制器

2*KD* 2*KD*

*u*(*k*) = u*u*(k*k* = 1) = *KP* =  *e*(*k*) = *k*  P =  *e*=  − 1)

*T T*

實現是此控制器使用相同的函數與一些未成年人更改。相應的函數是:

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Main program %

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

主要

% 資料

% 變數

% 雖然(1) 時,loo p

讀取速度;

計算控制;

發送控制;結束;

% 讀取速度功能

% 計算控制功能

% 傳送控制功能

現在,讓我們使用此示例的狀態空間表示形式,並設計一個狀態反饋控制器,保證所需的性能。在這種情況下,我們將首先假設完全訪問狀態,其次,我們放寬這一假設by假設我們只能訪問該位置。正如我們之前所做的那樣,我們可以在連續時間或離散時間進行。

以前,我們建立這個系統的狀態空間描述,它由:

0 1 0

*x*μ(*t*) = *x*(*t*) = *K* *u*(*t*)

0 =|

其中*x*(*t*) = R2 (*x*1(*t*) = =(*t*)和*x*2(*t*) =μs( t)和*u*(*t*) = R (施加的電壓)。 = θ

從在 %2 等τ於 3*n* ω 的所需效能中,我們得到與以前相同的主導極,因此相同的特徵方程,μ  d (*s*) = *s*2 + 2μ*wns* = *w* = 0= 0 (與 #= 0)。707 與*wn* =  。使用控制器表達式提供閉環特徵方程,

det (*s*I = *A* = *BK*) = 0

通過平衡這兩個方程,我們獲得了以下收益:

*K*1 = 1。1. 46

*K*2 = 0。0326

使用此控制器,圖 11.6 中還具有等於 30 度的 mplit 的輸入的時間回應。從這個數位可以看出,過沖和結算時間是我們想要的。重要的是要注意到錯誤在穩定的狀態制度的存在。如果我們在迴圈中添加積分操作,則此錯誤或可以消除。有關此的更多詳細資訊,我們將讀者引至 [1]。

對於第二種情況,由於我們無法訪問負載速度,我們可以從位置計算它,或者使用觀察者來估計 the 系統狀態。如前所述,我們用於觀察者設計的極點應比控制器設計中使用的極數快。

選擇以下極點*(s*1,2  4倍於控制器設計中使用的極點的實際部分): 1



*s*1,2 = 44°*wn* = *jwn* !1 = 2

83.4218 - 20.8618*D*

0

0.1

0.2

0.3

0.4

0.5

0.6

0.7

0.8

0.9

1

0

5

10

15

20

25

30

以秒表示的時間

出口

**圖 11.6**以 30 度為振幅的步進函數的時間回應

我們為觀察者獲得以下收益:

*L*1 - 151.2

*L*2 - 5029.4

控制器的增益與狀態向量的完整存取大小寫相同。

在下面的 Matlab 中,我們同時提供控制器和觀察者的設計,並給出模擬,顯示系統狀態和觀察者相對於時間的行為。

全部清除

%資料 tau=0.064 k=48.9

1. [0 1;0 -1/tau];
2. [0; k/tau];
3. [ 1 0] ;
4. = 0;

% 控制器設計

1. [ a, B, -3+3]j -3-3\_j]);
2. [ a', C', \_-12\_3\_j -12-3\_j]' ;

% 模擬數據 Ts = 0.01;x0 = [1 ; 1];z0 = [1.1 ; 0.9];Tf = 2; %最終時間

%增強系統

啊 = [A -B+K;

L\*C A-B\*K-L\*C];

Bh = 零(大小(Ah,1),1);Ch = [C D\_K];

Dh = 零(大小(Ch,1),1);xh0 = [x0 ; z0];

t=0:Ts:Tf;u = 零(大小(t);m = ss(啊,Bh,Ch,Dh);

%類比

[y,t,x] = lsim(m,u,t,xh0);

%繪圖圖;情節(t,y);標題("輸出");xlabel("秒內時間")ylabel("輸出")網格

圖;圖(t,x(:,1:大小(A,1));標題("系統狀態");xlabel("在sec 中"Ti 我)ylabel("系統狀態")網格

圖;圖(t,x(:,大小(A,1)=1:結束);標題("觀察者狀態");xlabel("時間在秒)"ylabel("觀察者狀態")網格

圖 (11.7)-(11.9) 說明瞭輸出、系統狀態和觀察者狀態。

0

0.2

0.4

0.6

0.8

1

1.2

1.4

1.6

1.8

2

±0.2

0

0.2

0.4

0.6

0.8

1

1.2

出口

以秒表示的時間

出口

**圖 11.7**輸出與時間

我們還可以使用線性二次穩壓器設計狀態反饋控制器。事實上,如果我們為成本函數選擇了以下矩陣*:Q* =

1 0

0 5 *R* = 10

**注 11.3.2***通常,對於成本函數的矩陣選擇沒有神奇的規則。但通常,我們使用高值控制,例如,將迫使控件採用小值,並可能防止飽和。*

使用這些矩陣與 Matlab 函數 **lqr,**我們得到:

*K* = 0。3162 0.6875.

我們還可以使用強健控制部件上的結果設計狀態反饋控制器。由於系統沒有不確定性,也沒有外部干擾,我們可以為標稱動力學設計狀態反饋控制器。使用系統數據和瑪律特拉布,我們得到: *X* = |

1.1358 0.3758

•0.3758 1.1465

*Y* - 00。0092 0.0228

0

0.2

0.4

0.6

0.8

1

1.2

1.4

1.6

1.8

2

±2.5

#2

±1.5

#1

±0.5

0

0.5

1

1.5

系統狀態

以秒表示的時間

系統狀態

**圖 11.8**系統狀態與時間

指定相應的控制器增益:

*K* = =0。0017 0.0193.

**備註 11.3.3***由於我們擁有直流電機套件的連續時間模型,因此我們已使用它來設計控制器增益。在這種情況下,我們已經解決了以下 LMI:*

*AX* + *XA* + *BY* = *YB* < 0

*增益 K 由: K* = *YX*=1*。*

*有關連續時間案例的更多詳細信息,我們將讀者參考 Boukas [2] 及其參考。*

## 11.4 平衡機器人控制

從控制的角度來看,平衡機器人是一個具有挑戰性的系統,因為它是一個不穩定的開放循環系統。Thi的系統吸引了很多研究人員,為此提出了許多設計建議。在這裏,我們將介紹在美卡電子實驗室開發和測試的設計,在蒙特爾理工學院。

0

0.2

0.4

0.6

0.8

1

1.2

1.4

1.6

1.8

2

±2.5

#2

±1.5

#1

±0.5

0

0.5

1

1.5

觀察者狀態

以秒表示的時間

觀察員狀態

**圖 11.9**觀察員狀態與時間

圖 11.10-11.11 給出了機器人的概念。它開發的目的是研究,使機電一體化的學生實現他們的控制演演演算法,並熟悉複雜的系統。機器人有兩個獨立的車輪,每個車輪由直流電機通過齒輪驅動,比率為1:6。每個電機都有一個編碼器來測量軸的速度。兩個電機連接到機器人的車身上。其他感測器(如加速度計和陀螺儀)用於測量傾斜角度。引入了適當的濾除器,以消除測量的雜訊,從而獲得有用的控制信號。

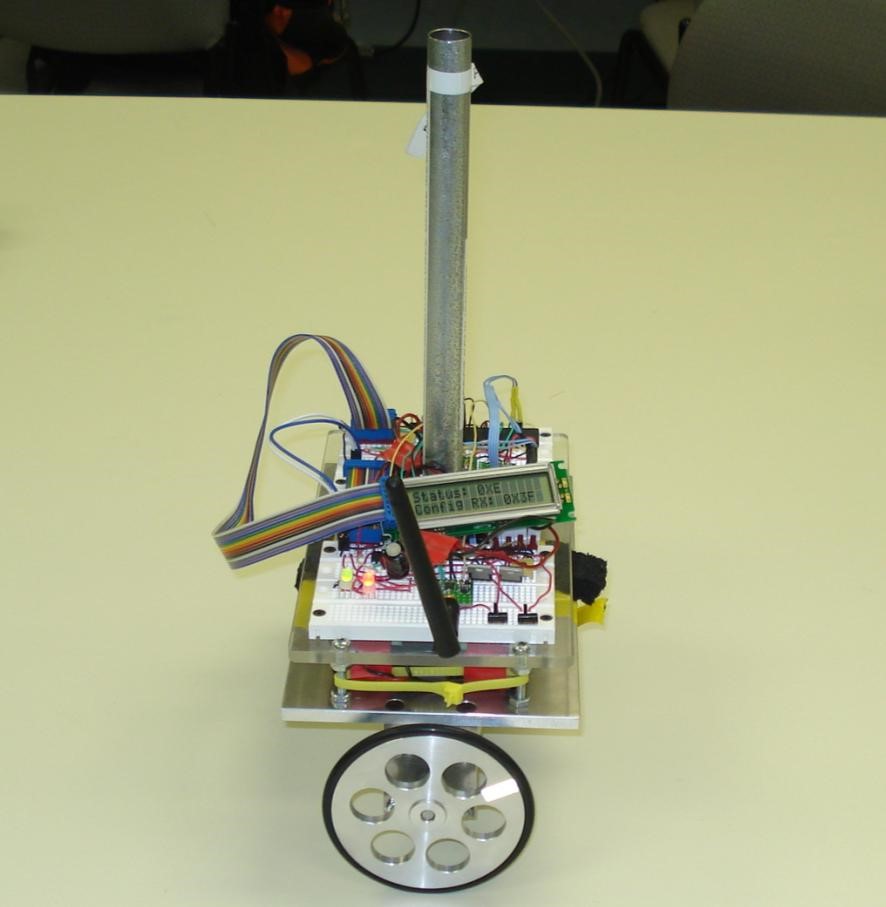
機器人的大腦圍繞家庭30F4011的微晶元dsPIC構建。所有程式設計在C中完成,並插入dsPIC,在利用PCKit2通過Microchip的C30引入可執行代碼後。如果我們提到第 4 章,數學模型由以下方法給出:

好

*x*̇(*t*)=*斧頭*(*t*)+*這*(*t*)

好

[*y*y(*t*) = *Cx*(*t*)



**圖11.10**平衡機器人

其中

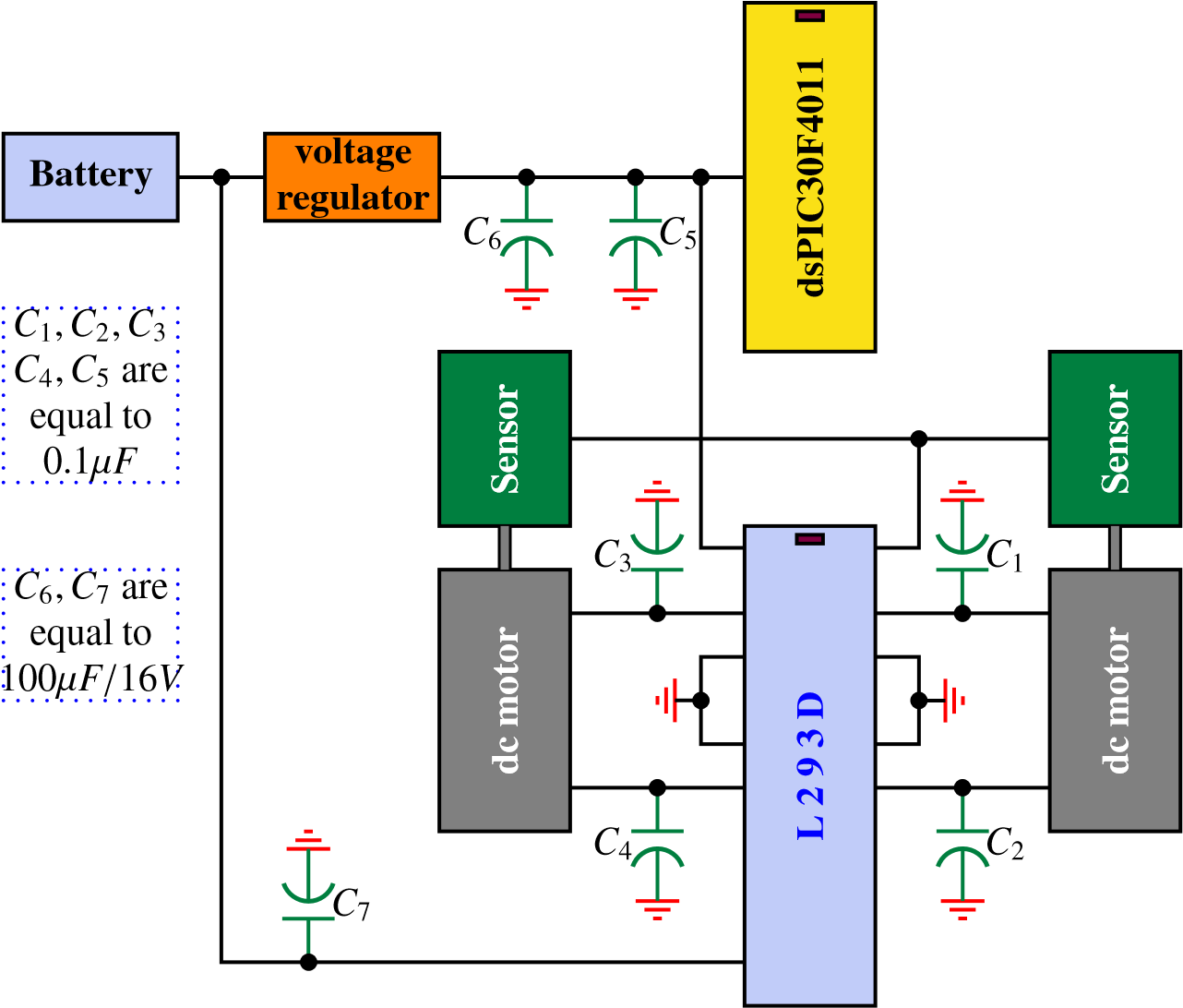
(*t*) ψ(

*t* (x ( t ) √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ √ ) = ⎢⎢⎢⎢⎢⎢⎢⎢⎢⎢ψ *xx* ˙˙ ((( *ttt* ))) ⎥⎥⎥⎥⎥⎥⎥⎤⎥⎥⎥⎦⎥⎥ ⎢⎣⎢

## *A* = ⎡⎢⎢⎢⎢⎢⎢⎢⎢⎢⎢147.0002931 −−00..0429 04864 001 00 −−100.9371.106325⎥⎥⎤⎥⎥⎥⎥⎥⎥⎥⎥⎥⎥⎦,

"??

1. [ ]010 1。.1295468700 ⎥⎥⎥⎥⎤⎥⎥⎥⎥⎥⎥⎥⎦⎥ ,
2. = 1 0 0 00 0 10。



**圖11.11**平衡機器人電子電路

由於系統在開放環路中不穩定,讓我們設計一個狀態反饋控制器,提供以下性能:

1. 系統在閉環中穩定;
2. 過沖少或等於5%;
3. 2% 的結算時間等於 1.5 s;從規格 we 得到:

[ = 0。707



*w* = 3。7718 rad/s = = 1.5

相應的主導極由以下給定:

*s*1,2 = 22。6667 × 2.6675*j*.

由於矩陣*A* 是排名四,我們需要再放置兩個極點來確定狀態回饋器增益 K  。讓我們選擇以下佔主導地位的兩極:

*s*3 = =13。3335 *s*4 = =13。3335 使用函數 *acker,*我們得到以下增益:

*K* = 339。5604 27.5946 ×132.5973 ×76.8450.

0

0.5

1

1.5

2

2.5

3

±0.4

±0.2

0

0.2

0.4

0.6

0.8

1

以秒表示的時間

出口

**圖 11.12**輸出與時間

該控制器的模擬結果如圖 11.12-11.13 所示。系統從初始條件*x*x 0 = 1 0 0 0開始。5 0,輸入為零。如果我們

嘗試發送輸入引用,我們將在狀態或輸出中出錯。克服

*需要*添加整體操作。如果我們用*[x*   (*t*) = 0 (x*xr*)*t*= x ( t) = *x*(*t*)*dt*表示,x (*t*)是位置參考, 在 [1] 下, 我們得到:

[ (*t*) = *A*˜=(*t*) = *Bu*=(*t*)

*y*(*(t)* ) - *C*˜-*(t)* ) (

0

0.5

1

1.5

2

2.5

3

#7

#6

#5

#4

#3

#2

#1

0

1

以秒表示的時間

國家

**圖 11.13**狀態與時間

其中

*x*(*t*)*x*̃(*t*)

((*t*) =,

*A* = *a* 0,

•*C* 0

*B*= B  ,

0

*C*- = *C* 0

新的動態變成順序五,我們需要修復三個佔主導地位的極點加上來自規格的主導極。這些極點固定在以下幾極:

*s*3,4 = =13.3335 × 2.6675*j s*5 = 1313。3335

使用函數*acker,*我們取得以下好處:

*K* = 339。5604 27.5946 ×132.5973 ×76.8450 0.1. .

我們還可以使用線性二次控制技術設計狀態反饋控制器。事實上,如果我們選擇以下矩陣:

⎡⎢1 0 0 0

* 1. [ 05 00 0 10 0 0 200 ⎤⎥⎥⎥⎥⎥⎥⎥⎥⎥⎥⎥⎥⎦ ]

·

* 1. = 10

並解決里卡蒂方程式使用Matlab函數*lqr,*我們得到:

*K* = 218。9494 17.7264 ×1.0000 ×15.7658

閉環的相應 eigen 值由以下值給出:

*s*1 = =12。7439 *s*2 = =11。5936 *s*3 = 00。9420 *s*4 = 00。1371

可以同樣獲得該控制器的模擬結果,並省略細節。

對於這個系統,我們還可以使用穩健的控制理論設計一個狀態反饋控制器。這可以在連續時間或離散時間完成。由於我們的模型是連續的,我們將在連續時間使用 LMI 進行設計。解決適當的LMI(我們使用的一個是直流電機套件與二等尺寸的矩陣,我們得到:

1. [ ] =0000...017710530595.2156 1=000. .77329238.0705 12156 0+00.44961053.3439 10705 0+00...502877320177.3439⎤⎥⎥⎥⎥⎥⎥⎥⎦⎥⎥⎥⎥⎥ ⎣⎢⎢⎢⎢年
2. [ ]7。2149 27.5211 ×13.6001 7.4801

為狀態回饋控制器提供以下增益:

*K* = =174。2563 ×10.5741 6.0437 13.8536

閉環的相應 eigen 值由以下值給出:

*s*1,2 = =6。9770 × 6.4079*j s*3,4 = =0.6028 × 0.9598*j*

這種方法的優點是,我們不必提供極點,如桿放置技術。最重要的是,LMI 技術可以更恰當地處理輸入中是否存在飽和度(如果是這種情況)。

舉一個例子,讓我們考慮*H*∞ 兩個輪子機器人的H+控制問題。在這種情況下,我們在狀態動態中添加一個術語。這個術語是*Bw*(*t*), 其中 *w*(*t*)是具有有限能量的外部擾動.使用 = 00求解相應的LMI。1,我們得到:

.7517 61

1. [ ] [ ]1012.2667 141233 =2.05127517 0=1。.02322742.5973 14×13.2667...493040554932

.2742 =1..40555973 2+3。4932 16

1. [ ]143。3219 316.5973 ×60.7376 ×33.8808

為狀態回饋控制器提供以下增益:

*K* = =193。7547 ×9.4711 39.7088 29.5092

閉環的相應 eigen 值由以下值給出:

*s*1,2 = =3。8372 × 8.9414*j s*3,4 = =1.9189 × 2.0781*j*

對於其他控制器的設計可以很容易地獲得,我們讓這個作為讀者的練習,因為設計帶來了一個程式Matlab類似於我們在文本中提供的程式。

### 11.5 磁浮浮系統

在本節中,我們將介紹我們前面介紹的磁懸浮系統。這個機電一體化系統在我們的機電實驗室中開發,由兩部分組成:一個表示線圈的固定部分,產生電磁力,另一個鐵磁物體,我們希望通過作用線圈產生的電磁力,放置在一定位置。系統的目標是透過輸入電壓調節電磁電磁層中的電流來控制移動物體的垂直位置。使用霍爾 effect 感應器測量物件位置。圍繞 dsPIC30F4011 構建的電體電路透過積體電路 L298 提供線圈,其電流與執行器的 comm 和電壓成正比。由於磁力只能吸引人,因此相互傳導放大器會為負命令旋轉 off。 圖 11.14 說明瞭此系統。

該系統的數學模型由以下方程給出:

*ml*= (*t*) = *毫克*= *F*1 = *F*2

*其中 m* 是移動物體的品質*,l*(*t*) = R= 是從電磁鐵測量的距離,F  1 和 *F*2分別是當電流為*i*(*t*)時線圈產生的力以及放置移動物體頭部的電磁磁鐵和永磁體的電磁力。

這些力的表達由:

*i*2(*t*)

*F*1 - *K*1  *l*2(t*t*)

1

*F*2 - *K*2  *l*2(t*t*)

這個非線性模型,我們可以線性化,得到以下(見第1章):

好

*x*̇(*t*)=*斧頭*(*t*)+*這*(*t*)

好

) *y* + ( *t* 'Cx (t) - 杜 (t) ( ) = *Du* *t* ) *t*

其中

*x*(*t*)=*x*1(*t*)(*位置*)

*x*2(*t*)(*速度*)??????? 02*e* 1???????

2

[

*標誌*

(

*美國*

2

)

*K*

*C*

*美國*

2

*和*

+

*K*

*P*

*R*

2

]



1. =

3 0 *mR x*

0

1. =\_\_"⎣−2*標誌*(2*u*)2*kcu和*""??⎦

好

好

*mR xe*

33*Cp*4*與* 0

1. = 0。032*x*

*Cb*

1. =

0.032*R*

系統的數據由表 11.1 提供。使用此資料,矩陣由以下指定:

0 1

1. =,

2490.8 0

0

1. =,

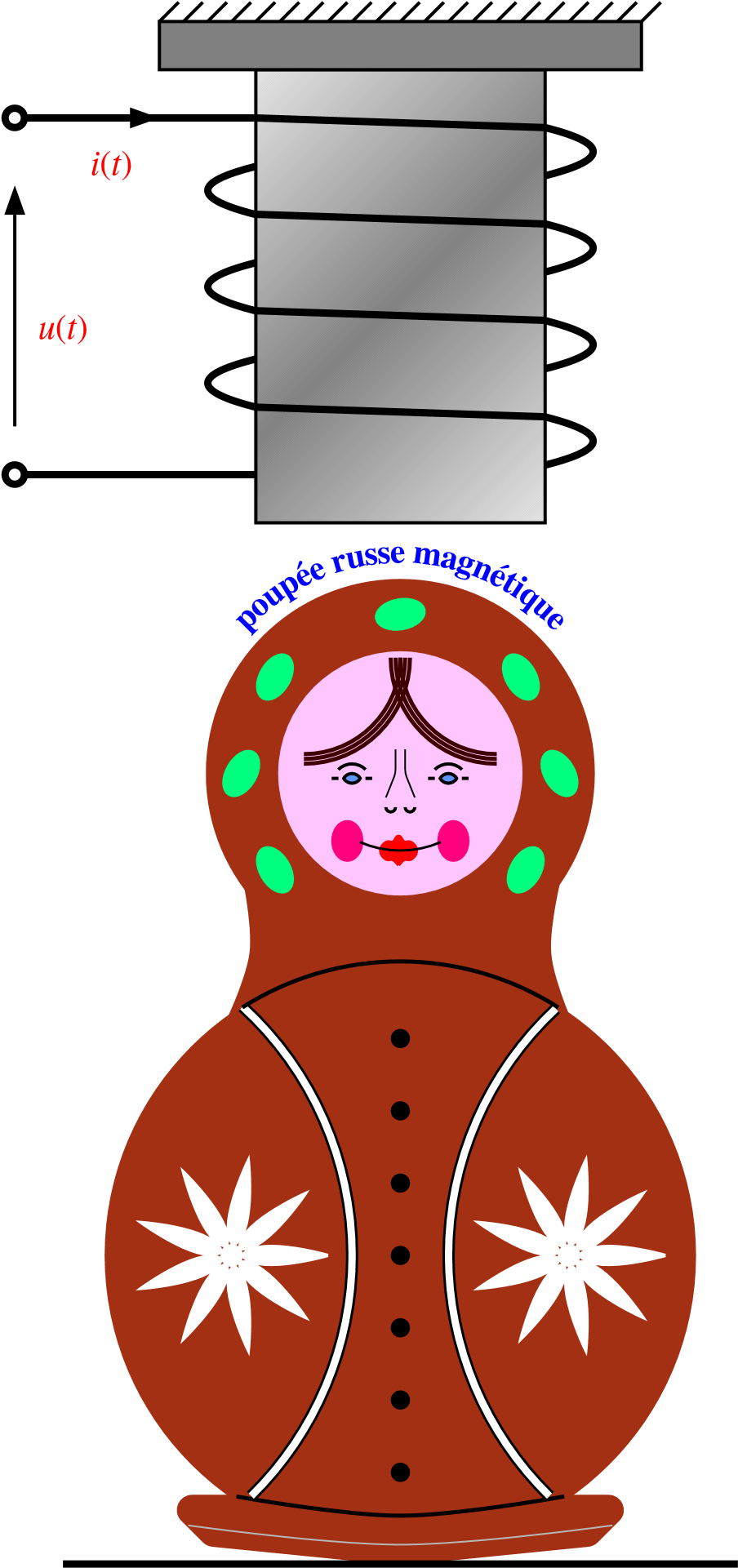
•1.2. 711

1. # 473.5711 0,
2. [ ]0。0833,

請務必注意,系統在開放迴圈中不穩定,因為它具有具有正實部分的極點。這可以通過計算矩陣*A*的 eigen 值來檢查。

讓我們設計一個狀態回饋控制器,以確保以下效能:

1. 系統在閉環中穩定
2. 過沖少或等於0.2 %
3. 在% 2 處的建立時間等於 0.05 秒



**圖11.14**磁浮浮系統**表11.1** 磁懸浮系統資料



|  |  |
| --- | --- |
| 變數 | 價值 |
| *R* | 62.7 € |
| *我* | 60 mH |
| m(物體品質) | 7.64 克 |
| *kc* | 5. .9218 10×4 |
| *kp* | 4. .0477 10×6 |
| *Cb*  *Cp*  永磁體的直徑 | •0.1671年  •1.9446 10×8  9 釐米 |



由於過衝小於或等於 0.2%,因此會導致 \* = 0。9. 建立時間為 2% 由:

4 *ts* |

*wn*

*其中wn*是自然脈衝。

如果我們將結算時間固定在 0。05 秒我們得到:

4

*n*  no. 88 =  = 88.8889

0.05 × 0.9

然後,設計的主要極點由:



*s*1,,2 = =*wn* = *jwn* !1 = 2 = 8080。000 × 38。75*j*

使用這對極,我們得到:

*K*1 = =175。6

*K*2 = =125。9

使用此控制器,從給定初始條件開始的時間回應在圖 11.15 中表示。從這個數位可以看出,過沖和結算時間是我們想要的。

對於第二種情況,因為我們沒有負載速度的塞,我們可以從位置計算它,或者使用觀察者來估計系統狀態。如前所述,我們用於觀察者設計的極點應比控制器設計中使用的極數快。

Choo唱以下極點(s*s*1,,2 4是控制器設計中使用的部分的實際部分):



*s*1,2 = 44°*wn* = *jwn* !1 = 2

$320. .0-38 ± .75*d*

0

0.1

0.2

0.3

0.4

0.5

0.6

0.7

0.8

0.9

1

±0.16

±0.15

±0.14

±0.13

±0.12

±0.11

±0.1

±0.09

±0.08

線性位置的時間回應

儀表中的線性位置

顯示時間(秒為單位)

**圖 11.15**移動物件的時間回應

我們為觀察者獲得以下收益:

*L*1 - 1.3. 51

*L*2 - 224.5. 927

我們可以像對直流電機套件和兩個車輪機器人一樣試驗所有其他控制器,但我們更願意讓這部分作為練習,讓讀者練習這些工具。請注意,我們邀請他/她在連續時間和離散時間的情況下進行設計,並比較結果。該系統的維度允許這一點。

給出一個用於程式碼系統的狀態回饋控制的範例:

#include <p30fxxxx.h>

#include <stdio.h>

#include <stdlib.h>

#include <adc10.h>

#include <math.h>

#include <uart.h>

//

設定

//

內頻 (30 MIPS) 指令/ 秒

樣品數量: 7,37\*16/4 = 29480000

\_FOSC(CSW\_FSCM\_OFF和FRC\_PLL16);

\_FWDT(WDT\_OFF);

\_FBORPOR(PBOR\_OFF和MCLR\_DIS);

\_FGS(CODE\_PROT\_OFF);

\_FICD(ICS\_NONE ); \_\_C30\_UART=2;

//

變數

#define Freq\_pic 29480000 // PIC 頻率

#define A11 7.8510061454215840e-001

#define A12 2.2727760074413661e-004

#define A21 5.0829248960838420e+001

#define A22 1.000643300984893e+000

#define B11 3.7771272752681438e-005

#define B12 4.5392069137012870e-004

#define B21 8.74963852857344e-003

#define B22 1.0852723324638587e-001

#define Ts 2.272727272727272727e-004

#define u\_max 1.178999999999999e+001

#deref\_tension5000万万万万万万万万万万万万万万万万万万万万万万萬萬萬萬萬萬萬萬萬

#define ref\_pic 1.024000000000000e+003

#define duty\_cycle\_ref 5.8481764206955049e+001

#define x\_ref 7.8768775539549939e-003

#define u\_ref 2.0000000000000000e+000

#define y\_ref 8.5691877396730676e-001

#define K0 5.2128678707944724e+004

#define K1 3.9336557697049994e+002

雙 y[2] - "0.0,0.0";雙 u[2] -0.0, 0.0;雙y\_tilde[2] - "0.0,0.0";雙tension\_tilde[2] - "0.0,0.0";雙電壓 - 0.0;雙duty\_cycle\_tilde - 0.0;雙lim\_Sup - 0.0;雙lim\_Inf -0.0; 雙position\_tilde[2] -0.0, 0.0;雙vitesse\_tilde[2] '0.0, 0.0'; 雙integrale\_tilde [2] '0.0, 0.0; 雙duty\_cycle - 0.0; 雙temps\_total - 0.0; 雙 n - 6553500.0/65536.0; int 計數器 -0; int compteur\_freq '0;int uart\_flag '1; 未簽名的長 Val\_reg '0;

//

功能

虛無(無效)\*

INTCON1位.NSTDIS=0;啟動中斷等級

TRISE = 0;PORTE 為輸出的設定

TRISD = 0;PORTD 為輸出的設定

埠.RE8 = 1;

埠.RE2 = 0;

PORTDbits.RD0 = 1;

ADPCFG= 0xFFFF;PORTB 引腳設定為數字

I/O = 空init\_ADC(空)\*

SetChanADC10(ADC\_CHX\_POS\_SAMPLEA\_AN3AN4AN5 &

ADC\_CHX\_NEG\_SAMPLEA\_NVREF);

配置IntADC10(ADC\_INT\_DISABLE);

openADC10(ADC\_MODULE\_ONADC\_IDLE\_CONTINUE及ADC\_FORMAT\_INTG

ADC\_CLK\_AUTOADC\_AUTO\_SAMPLING\_ONADC\_SAMPLE\_SIMULTANEOUS,

ADC\_VREF\_AVDD\_AVSSADC\_SCAN\_OFF及ADC\_CONVERT\_CH\_0ABC

ADC\_SAMPLES\_PER\_INT\_1及ADC\_ALT\_BUF\_OFF及ADC\_ALT\_INPUT\_OFF、ADC\_SAMPLE\_TIME\_1及ADC\_CONV\_CLK\_SYSTEM及ADC\_CONV\_CLK\_32Tcy,

ENABLE\_AN4\_ANA及ENABLE\_AN5\_ANA,SCAN\_NONE);

[ 空init\_Timer1 (空) ]

INTCON1bit.NSTDIS=0;// 啟動模式 16 位元計時器1

T1CONbits.TON - 1;計時器1 授權

T1CONbits.TGATE - 0;D 啟動杜模式計時器門

T1CONbits.TSIDL=1;定時器1 的同步速度(即空閒模式 T1CONbits.TCKPS = 0;預縮放器的喬冰 1:1

(1×1:8,2×1:64)

T1CONbits.TCS=0; // 選擇實習時鐘 (0=FOSC/4)

IFS0位.T1IF = 0;將計時器中斷的溢出位歸零 1

IEC0位.T1IE = 1;啟動計時器的中斷1

PR1 = 6699;4400 Hz 環境處的取樣頻率

IPC0位.T1IP = 5;優先權 5 用於中斷

計時器1

}

/= "保護"程式 [/ 空init\_UART (空)]

設定intUART2(UART\_RX\_INT\_DIS和UART\_RX\_INT\_PR0

UART\_TX\_INT\_DIS和UART\_TX\_INT\_PR0);

寄存器 U2MODE 的設定

U2MODEbit.UARTEN = 1;UART 引腳由 UART 控制

U2MODEbit.USIDL = 0;UART 通訊繼續在

閒置模式

U2MODEbit.WAKE = 1;睡眠模式開啟喚醒

U2MODEbit.LPBACK = 0;關閉回環模式

U2MODEbits.ABAUD = 0;自動包工序進程禁用 U2MODEbit.PDSEL = 0;8 位元資料,無奇偶校驗 U2MODEbit.STSEL = 0;1 個停止位。

U2STA 註冊表設定

U2STAbit.UTXISEL = 0;傳輸中斷模式

選擇位

U2STAbit.UTXBRK = 0;UxTX 引腳工作正常

U2STAbit.UTXEN = 1; // 傳輸啟用

U2STAbit.URXISEL = 1;收到一個字元時發生中斷

U2STAbits.加登 = 0; // 位址偵測已關閉

U2BRG = 31; // 值 57600 bps 波特酸鹽

}

//

互補模式PWM的初始化

//

空init\_PWM(空)\*

Val\_reg - 1023;約 30,000 Hz lim\_Sup - (u\_max (2-Val\_reg - 1)/(2-Val\_reg - 2) - u\_ref;lim\_Inf - -u\_max - u\_ref;

PTCONbits.PTEN = 1;啟用時間基礎

PTCONbits.PTSIDL = 1;閒置模式下的設定

PTCONbits.PTCKPS -0;// 選擇4TCY(前尺度:00 - 1:1;

01× 1:4;10 = 1:16;11 = 1:64)

PTCONbits.PTMOD = 0;選擇自由執行模式

PTMRbits.PTDIR = 0;時基的增量

PTMRbits.PTMR = Val\_reg;時間基礎的註冊值

PTPER = Val\_reg;信號週期的值

PWMCON1位.PMOD1 = 0;選擇模式 PWM 互補

PWMCON1位.PEN1H = 1;啟動模式 PWM 的引腳

PWMCON1位.PEN1L = 1;啟動模式 PWM 的引腳

DTCON1位.DTAPS = 0;時基單位為 1TCY

DTCON1位.DTA = 0;DT 對統一 A 的價值

PDC1 = 0;佔空週期零

[空\_\_attribute\_\_(中斷,auto\_psv))))\_T1Interrupt(空)\*(IFS0位.T1IF)\*

波特位.RE2 = !埠.RE2;

PDC1 - (2.0 - (Val\_reg - 1) - duty\_cycle)/100.0;計算 PDC1 註冊表的值

y[0] = (讀取ADC10(2)\*ref\_tension)/ref\_pic;伏特中感應器的訊號

y\_tilde [0] 和 {0} - y\_ref;

position\_tilde[1] = position\_tilde[0];vitesse\_tilde[1] = vitesse\_tilde[0];integrale\_tilde[1] = integrale\_tilde[0];y\_tilde[1] = y\_tilde[0];tension\_tilde[1] = tension\_tilde[0];

position\_tilde[0] = (a11\_position\_tilde[1]a12\_vitesse\_tilde[1]

[b11]tension\_tilde[1]b12\_y\_tilde[1]);

vitesse\_tilde[0] = (a21\_position\_tilde[1]a22\_vitesse\_tilde[1]

[b21]tension\_tilde[1]b22\_y\_tilde[1]);

tension\_tilde[0] = (K0\*position\_tilde[0]) |

(K1-vitesse\_tilde[0]);// "N 參考;

如果(tension\_tilde{0}>lim\_Sup)[張力\_tilde[0] = lim\_Sup;]

如果(tension\_tilde[0]<lim\_Inf)[tension\_tilde[0] = lim\_Inf;\*

張力 = u\_ref = tension\_tilde[0];duty\_cycle\_tilde = tension\_tilde[0](50.0/u\_max);duty\_cycle = duty\_cycle\_ref = duty\_cycle\_tilde;以百分比表示佔空比的計算

temps\_total = Ts;compteur\_freq = 0;

反-如果(計數器 -10)/Print 數據每 1 毫秒計數器 -0.0;uart\_flag -1;

IFS0位.T1IF = 0;置於溢位零

}

}

/= 課程主題 [/ int 主 (虛無)]

init();init\_PWM();init\_ADC();init\_UART();init\_Timer1();(1)\* 如果 (uart\_flag)= printf("%lf %lf %lf %lf %lf %lf %lf %lf\n\r",temps\_total、張力、u\_ref、y[0]、y\_ref、position\_tilde[0]、x\_ref);uart\_flag = 0;

}

}

}

### 11.6 結論

本章介紹了在蒙特爾理工學院機電一體化實驗室開發的一些案例研究。我們介紹了機電一體化系統設計的所有步驟,其中有一系列細節。重點介紹了每個系統控制演演演算法的desig n。

### 11.7 問題

1. 讓我們考慮一個動態離散時間系統,具有以下數據:

*A*  [ ] 0100 +2 =103 =

[00 0 *B* ] [ 100 1 ]

*C* - 1 1 0, *D* - 0

11.7. 問題

|  |  |
| --- | --- |
| *DA*  *和A* | [ 00000 ].32 ·  = 0。1 ×0.2 ×0。1, |
| *DB*  *和B* | 00.1  [ 00000 ].32 ·  = 0。1 ×0.2, |
| *DC* | = 0。1, |
| *與C* | = 0。1 0.2 ×0。1, |

具有以下規範限制的不確定性: \*00。1·

1. 標稱系統設計以下控制器:
   1. 狀態回饋控制器
   2. 靜態輸出回饋控制器
   3. 動態輸出回饋控制器
2. 不確定系統的設計:以下控制器:
   1. 狀態回饋控制器
   2. 靜態輸出回饋控制器
   3. 動態輸出回饋控制器
3. 如果我們有一個額外的術語在*A*state動態,添加外部干擾: *x*(*k* + 1) \* \* \*  A+ *x*(*k*  ) + B [ +*B*+*u*(*k*) = *Bww*(*k*)。為標稱和不確定系統設計控制器(狀態反饋、靜態輸出、動態輸出反饋),以確保 *H*= 性能。
4. 設計狀態回收費、靜態輸出反饋和動態反饋控制器,確保所有可接受的不確定性的保證成本。
5. 在這個問題中,我們邀請你繼續設計一艘小船,例如,您可以使用歡樂的k來控制它,讓它在小乳光中移動。
   1. 提供原理圖設計(電池、馬達等)
   2. 建立數學模型
   3. 修復您希望擁有的規格,並設計適當的控制器,可以提供這樣的性能
6. 在這個問題中,我們邀請您繼續設計小型飛機,您可以使用操縱桿進行控制,使其飛行。
   1. 提供s-運動設計(電池、馬達等)
   2. 建立數學模型
   3. 修復您希望擁有的規格,並設計適當的控制器,可以提供這樣的性能
7. 在這個問題中,我們要求設計一個真空吸塵器。這一點應該是自動的,避免在其環境中的障礙。設計一個廉價的,可以通過發射器和接收器和操縱桿進行無線通信也是很重要的。
   1. 給出原理圖設計(電子電路、電機等)
   2. 建立數學模型
   3. 修復您希望擁有的規格,並設計適當的控制器,可以提供這樣的性能
8. 在這個問題中,我們邀請您繼續設計一個機電一體化系統,該系統控制著一個小球的 pi ngpong 的位置。壓空氣可用於定位球。
   1. 給出原理圖設計(電子電路、電機等)
   2. 建立數學模型
   3. 修復您希望擁有的規格,並設計適當的控制器,可以給這樣的執行nces
9. 在這個問題中,我們希望設計一個單腿機器人,可以在保持垂直位置時使用一個輪子移動。提供這種機電一體化系統的設計。
   1. 給出原理圖設計(電子電路、電機等)
   2. 建立數學模型
   3. 修復您希望擁有的規格,並設計適當的控制器,可以提供這樣的性能
10. 太陽能是可以使用的另一種電源。在這個問題中,我們要求您設計一種太陽能,以最大化太陽能電池板產生的能量。
    1. 給出原理圖設計(電子電路、電機等)
    2. 建立數學模型
    3. 修復你想擁有的規格,並設計適當的控制器,可以給予這樣的rform
11. 在這個問題中,我們要求設計一個可以控制通過發射器和接收器使用操縱桿在水上密封的轉向。
    1. 給出原理圖設計(電子電路、電機等)
    2. 建立數學模型
    3. 修復您希望擁有的規格,並設計適當的控制器,可以提供這樣的性能

# 附錄A

C 語言教學

本附錄的目的是回顧C語言,以刷新讀者的記憶,並説明他開始寫他的程式與o ut閱讀有關這個主題的大書。我們的意圖不是取代這些不熟悉這個主題的讀者,強烈鼓勵查閱克尼漢和里奇的書 [[2]](#footnote-2) 。

首先,我們邀請讀者下載C編譯器和文字編輯器。要在ffC 中試驗我們將提供的程式,讀者必須鍵入程式,保存它們,然後編譯它們。有關如何執行此操作的更多詳細資訊,請參閱已使用的C編譯器手冊。

要開始我們的教程,讓我們考慮我們的第一個簡單程式。本計劃旨在寫"歡迎機電一體化課程」。此程式的清單由以下行給出:

#include < stdio.h>

空主()

[printf("\n歡迎參加美式電子課\n");

要查看此簡單程式的輸出,我們需要有一個編輯器和一個 C 編譯器。

每個 C 程式由變數和函數組成,並且必須具有"主"函數。除了一些保留的單詞,所有變數和函數都必須聲明才能使用它們。變數可以採用以下類型之一:

* *整數*
* *真正*
* *字元*
* 等。

這些函數指定程式必須執行的任務以及程式為其設計的任務。"函數建立代碼的整體邏輯。

讓我們檢查一下以前的程式。第一個語句

#include < stdio.h>

包含 C I/O 函式庫的規範。".h"檔是約定的"头檔",其中包含程式運行所必需的變數和函數的定義,無論是使用者編寫的代碼部分,還是標準C 庫的一部分。指令

#include

告訴 C 編譯器在代碼的這一點上插入指定文件的內容。

符號

<...>

指示 C 編譯器在某些"標準"系統目錄中尋找檔。

"main"的*void*表示主類型為"void"類型,即它沒有與其關聯的類型,這意味著在另一個意義上,它不能在執行期間返回結果。

";"表示語句的結尾。語句塊被放入 braces [ ] |,如函數的定義。所有 C 語句都以自由格式定義,即沒有指定的佈局或列分配。空白(選項卡或空格)從來就不重要,除非內部引號作為字串的一部分。

statement *printf* 行在「stdout」(與運行代碼的 X 終端機對應的輸出流)上列印訊息「歡迎加入機電一體化課程」,而語句

\n

列印一個"新行"字元,將游標帶到下一行。通過構造,函數 printf 永遠不會自行插入此字元,這是讓給程式師的。

標準 C 語言具有一些保留的關鍵字,程式師不能將其用於命名變數或其他目的,他必須像建議的那樣使用它們,否則在編譯過程中會出現錯誤。這些關鍵字列在表 A.1 中。

**表 A.1** C 語言關鍵字清單



|  |  |  |  |
| --- | --- | --- | --- |
| 車 | 雙 | Int | 結構 |
| 打破 | 還 | 長 | 開關 |
| 情況 下 | 枚舉 | 註冊 | 類型def |
| 字元 | 外部 | 返回 | 聯盟 |
| 常量 | 浮動 | 短 | 符號 |
| 繼續 | 對於 | 簽署 | 無效 |
| 預設 | 去 | 大小 | 揮發 性 |
| 的 | 如果 | 靜態 | 而 |



**表 A.2**數位表示

|  |  |  |  |
| --- | --- | --- | --- |
| 基地 | 表示 | 允許的奇ffres | 例子 |
| 十進位 (10) |  | 0123456789 | 5 |
| 二進位 (2) | 0b... | 01 | 0b10101010 |
| 八進制 (8) | 0... | 01234567 | 05 |
| 十六進位 (16) | 0x... | 0123456789ABCDEF | 0 x 5A |



**表 A.3**整數表示

|  |  |  |  |
| --- | --- | --- | --- |
| 類型 | 大小(位元) | 最小值 | 麥克斯 |

|  |  |  |  |
| --- | --- | --- | --- |
| 字元,簽名字符 | 8 | -128 | •127 |
| 無號字元 | 8 | 0 | 255 |
| 短,簽名短 | 16 | -32768 | +32767 |
| 無號短 | 16 | 0 | 65535 |
| int, 簽名 int | 16 | -32768 | +32767 |
| 無號 int | 16 | 0 | 65535 |
| 長,簽長  未簽署長  長,簽長無符號長長 | 32  32  64  64 | •231  0  •263  0 | +231 × 1 232 × 1  +263 × 1  2.64 1 |

−

**表 A.4**十進位表示

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 類型 | 大小(位元) | 埃明 | 埃馬克斯 | 內金 | Nmax |
| 浮動 | 32 | -128 | •127 | 2×126 | 2128 |
| 雙 | 32 | -128 | •127 | 2×126 | 2128 |
| 長雙 | 64 | -1022 | +1023 | 2×1022 | 21024 |

讓我們看一下常量和變數。在使用它之前,必須定義常量。我們用於定義這些常量的結構是:

#define名称值

單詞名稱是我們對常量給出的,值是此常量獲取的值。下面的範例給出了一些常量:

#define加速 9.81

#define pi 3.14

#define莫特"歡迎參加機電一體化課程"

#define false 0

通常,當我們操作數據時,我們使用 dfferent 數位表示形式。

表 A.2 給出了在程式設計微控制器時通常使用的基。

對於變數,我們還必須在之前聲明它們。使用以下語法:

類型 <name>;

類型為以下類型之一:

* int (對於 integer 變數)
* 短(用於短整數)
* 長(長整數)
* 浮點(用於單一精確度實值(浮點)變數)
* 雙精度(用於雙精度實(浮點)變數)
* 字元(用於字元變數(單個字節))

表 A.3 和 A.4 提供了每種類型中採用的值。請務必指出,編譯器會檢查程式中使用的所有變數類型的一致性,以防止錯誤。下面的 example 顯示了一些變數:

int i,j,k;浮點 x,y;無符號字元 var1;

無符號字元 var2[10] = "歡迎";

一旦定義了這些變數和常量,我們可以做什麼?答案是,我們可以做很多事情,如:

* 算術運算(對一個或多個變數執行作業)
* 邏輯操作(對一個或多個變數執行) = 關聯操作(對一個或多個變數執行)
* 等。

表 A.5 和 A.6ff給出了一 個有關我們可以對常量或變數執行的操作的想法。

**表 A.5**算術運算

|  |  |  |
| --- | --- | --- |
| 象徵 | 意義 | 例子 |
| + | 除了 | *u*=*k*= = + *1*1=*k*= = *2*2=*k*=; |
| -  / | 襯 底  劃分 | *在*[*k*] = *在*1=*k*= = *在*2=*k*\*;*在*[*k*] = *在*1=*k*+/*在*2=*k*\*;(u*u*2[k*k*]必須ff從零開始 ) |
| \*  % | 莫杜洛乘法 | *u*[*k*] = *k* = *x*=*k*=;*u*=*k*= = *1*1=*k*=%*u*2=*k*\*; |



**表 A.6**邏輯操作

|  |  |  |
| --- | --- | --- |
| 象徵 | 意義 | 例子 |

|  |  |  |
| --- | --- | --- |
| & | 和 | *u*=*k*= u  1=*k*+和*u*2=*k*=; |
| |  ∧  ! | 或  Xor  反向移位左移右 | *u*=*k*= = *1*1=*k*= |*u*2=*k*+;  *u*[*k*] = *1*1=*k*= u  2=*k*=;*u*=*k*= [ x ]  *k*=;  *u*=*k*= = *1*1=*k*= *2*2=*k*=;*u*=*k*= = *1*1=*k*= *2*2=*k*=; |



更多的時候,我們需要在螢幕上或液晶屏上列印數據。為此,可以使用*列印*功能。可以指示此功能正確列印整數、浮點和字串。一般語法是 printf("格式",變數);

其中格式指定轉換規範和變數是

要列印的數量。

有用的格式是:

%nd 整數(選擇 n = 欄位;如果為 0,則填充為零)

m.nf浮動或雙數的 %(可選 m = 欄數,n = 小數位數)

%ns 字串 (選擇性的 n = 欄位)

%c 查拉cter

\n \t 以引入新行或選項卡

\g 在終端上按鈴("蜂鳴")

在某些程式中,迴圈的概念比執行需要對資料流或內存區域重複操作的計算更可取。有幾種方法可以迴圈在標準 C 中,以下是最常用的迴圈:

循環時(運算式)

• 要執行的語片

}

用於迴圈(expression\_1;expression\_2;expression\_3)

【 要執行的語片

在某些情況下,我們需要根據某種條件的實現或依賴於給定變數的值來執行操作。在這種情況下,可以使用單詞*if* 或 單詞 *if* 和開關。使用以下結構:

如果 (conditional\_1)

• 當conditional\_1為 true 時要執行的語句區塊

• 否則(conditional\_2)

• 當conditional\_2為 true 時要執行的語句區塊

• 否則

• 否則要執行的語句塊 \*

表 A.7 和 A.8 顯示了我們可能在運算式中使用的最常用的 conditi onal 運算符。

**表 A.7**邏輯操作

|  |  |  |
| --- | --- | --- |
| 象徵 | 意義 | 例子 |
| & | 和 | *x*&y |
| ||  ! | 或不 | *x* *並且*  *x*!*並且* |

**表 A.8**邏輯操作

|  |  |  |
| --- | --- | --- |
| 象徵 | 意義 | 例子 |
| < | 小於 | *x* < *y* |
| [lt;] | 小於或等於 | *x* <y *y* |
| == | 等於 | *x* == *與* |
| != | 不等於 | *x*!並且*y* |
| [gt;] | 大於或等於 | *x* >y *y* |
| > | 大於 | *x* > *y* |



開關(運算式)

【 案例const\_expression\_1:

•要執行的語句塊中斷;

• 案例const\_expression\_2:

• 要執行的語句塊中斷;

• 預設值:

• 語句塊

}

}

C 語言允許程式師使用指標直接訪問記憶體位置。為了顯示其工作原理,讓我們考慮定義如下的變數 X定位:

浮動 X定位;X定位 = 1.5;

例如,我們想要取得變數 Xposition 的位址,可以使用以下功能:

浮點= pX定位;浮動 X定位;

X定位 = 1.5;pX定位 = 和 X 定位;

在此代碼中,我們定義指向浮點類型、X定位和將其設置為等於變數 X定位位址的指標。

要獲取指標引用的記憶體位置的內容,請使用"\*"運算符(這稱為取消引用的指標)。因此,\_pX定位是指X定位的值。

任何類型的陣列在 C 中起著重要作用。陣列聲明的語法很簡單,下面給出如下:

類型 varname\_dim\_;

其中暗淡是我們想要 to 給陣列 varname 的維度。

在標準 C 中,陣列從位置 0 開始,其所有元素都佔用記憶體中的相鄰位置。因此,如果*矩陣*是陣列,則=*矩陣*與*矩陣*[0] 相同,而 [(*矩陣* = 1) 與*矩陣*[1] 相同,等等。

1. 請參考www.microchip.com [↑](#footnote-ref-1)
2. 布賴恩·克尼漢和鄧尼斯·裡奇,C程式設計語言+ANSI C,普倫蒂斯·霍爾,1988年 [↑](#footnote-ref-2)