



บทที่ 4: ชนิดของข้อมูลและการใช้งานตัวแปรสำหรับโปรแกรม C51

ไมโครโปรเซสเซอร์ 1

Sanya Kuankid, Ph.D.
<http://pws.npru.ac.th/sanya/>

บทที่ 4 ชนิดของข้อมูลและการทำงานตัวแปรสำหรับโปรแกรม C51

4.1 บทนำ

4.2 ชนิดของข้อมูลสำหรับโปรแกรม C51

4.3 การใช้งานตัวแปรสำหรับโปรแกรม C51

4.4 แถวลำดับและสายอักขระสำหรับโปรแกรม C51

4.5 ตัวชี้

4.6 ตัวแปรโครงสร้างและยูเนียน

4.7 สรุป

4.8 แบบฝึกหัดท้ายบท



4.2 ชนิดของข้อมูลสำหรับโปรแกรม C51

ชนิดของข้อมูลสำหรับโปรแกรม C51



ชนิดของข้อมูล	ขนาด		ค่าที่เก็บได้
	บิต	ไบต์	
bit	1	0	0 ถึง 1
signed char	8	1	-128 ถึง 127
unsigned char	8	1	0 ถึง 255
enum	16	2	-32,768 ถึง +32,767
signed short int	16	2	-32,768 ถึง +32,767
unsigned short int	16	2	0 ถึง 65,535
signed int	16	2	-32,768 ถึง 32,267

ที่มา: ประจัน พลังสันติกุล และชัยวัฒน์ ลัมพรจิตรวิไล, 2550

ชนิดของข้อมูลสำหรับโปรแกรม C51



ชนิดของข้อมูล	ขนาด		ค่าที่เก็บได้
	บิต	ไบต์	
unsigned int	16	2	0 ถึง 65,535
signed long int	32	4	-2,147,483,648 ถึง +2,147,483,647
unsigned long int	32	4	0 ถึง 4,294,967,295
float	32	4	$\pm 1.17549\text{E}-38$ ถึง $\pm 3.402823\text{E}+38$
sbit	1	0	0 หรือ 1
sfr	8	1	0 ถึง 255
sfr16	16	2	0 ถึง 65,535

ที่มา: ประจัน พลังสันติกุล และชัยวัฒน์ ลัมพรจิตรวิไล, 2550



1. ชนิดของข้อมูลแบบบิต

ตัวอย่างที่ 4.1 โปรแกรมกำหนดชนิดของข้อมูลแบบบิต

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      bit my_flag;
5      my_flag = 1;
6  }
```

ผลลัพธ์ของการรันโปรแกรม

Name	Location/Value	Type
MAIN	C:0x080C	
my_flag	1	bit



2. signed char และ unsigned char

ตัวอย่างที่ 4.2 โปรแกรมกำหนดชนิดข้อมูลตัวอักษร

```
1 #include <REGX52.H>
2 void main (void)
3 {
4     unsigned char var1, var2;
5     var1 = 0xA4 ;
6     var2 = var1 ;
7 }
```

ผลลัพธ์ของการรันโปรแกรม

Name	Location/Value	Type
MAIN	C:0x080C	
var1	0xA4 'ต'	uchar
var2	0xA4 'ต'	uchar



3. signed int และ unsigned int

ตัวอย่างที่ 4.3 โปรแกรมกำหนดชนิดข้อมูลจำนวนเต็ม

```
1 #include <REGX52.H>
2 void main (void)
3 {
4     unsigned int n1, n2;
5     n1 = 10;
6     n2 = 2*n1;
7 }
```

ผลลัพธ์ของการรันโปรแกรม

Name	Location/Value	Type
MAIN	C:0x0800	
n1	10	uint
n2	20	uint



4. signed long และ unsigned long

ตัวอย่างที่ 4.4 โปรแกรมกำหนดชนิดข้อมูลจำนวนเต็มช่วงยาว

```
1 #include <REGX52.H>
2 void main (void)
3 {
4     unsigned long temp;
5     temp = 250000;
6 }
```

ผลลัพธ์ของการรันโปรแกรม

Name	Location/Value	Type
MAIN	C:0x0800	
temp	250000	ulong



5. ชนิดข้อมูลตัวเลขจำนวนทศนิยม (float)

ตัวอย่างที่ 4.5 โปรแกรมกำหนดชนิดข้อมูลตัวเลขจำนวนทศนิยม

```
1 #include <REGX52.H>
2 #include <math.h>
3 void main (void)
4 {
5     float t1, t2;
6     t1=25.4;
7     t2=sqrt(t1);
8 }
```

ผลลัพธ์ของการรันโปรแกรม

Name	Location/Value	Type
MAIN	C:0x0A8C	
t1	25.4	float
t2	5.039841	float

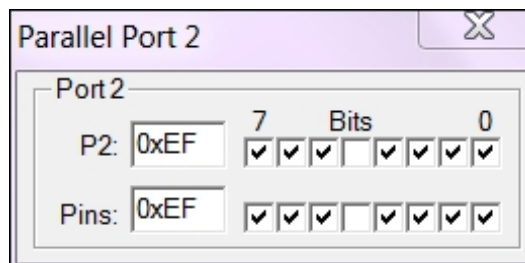


6. ชนิดข้อมูลแบบ sbit

ตัวอย่างที่ 4.6 โปรแกรมกำหนดชนิดข้อมูลแบบ sbit

```
1 #include <REGX52.H>
2 sbit mybit = P2^4;
3 void main (void)
4 {
5     mybit = 1;
6     mybit = 0;
7 }
```

ผลลัพธ์ของการรันโปรแกรม





7. ชนิดข้อมูลแบบ sfr

ตัวอย่างที่ 4.7 โปรแกรมกำหนดชนิดข้อมูลแบบ sfr

```
1  sfr P1 = 0x90;
2  sfr P2 = 0xA0;
3  void main (void)
4  {
5      unsigned char my_data;
6      P1 = 0x55;
7      my_data = P1;
8      P2 = my_data + 1;
9  }
```

ผลลัพธ์ของการรันโปรแกรม

Parallel Port 1		Parallel Port 2	
Port 1		Port 2	
P1: 0x55	7 Bits 0	P2: 0x56	7 Bits 0
Pins: 0x55	☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒	Pins: 0x56	☒ ☒ ☒ ☒ ☒ ☒ ☒ ☒



4.3 การใช้งานตัวแปรสำหรับโปรแกรม C51



- ตัวแปร (Variables) คือ ชื่อที่ประกาศขึ้นมาเพื่อใช้ในการเก็บข้อมูลที่ต้องการใช้งานในระหว่างที่โปรแกรมกำลังทำงานอยู่ ไม่ว่าจะเป็นการเก็บข้อมูลไว้ชั่วคราว พักข้อมูล การนำข้อมูลมากระทำทางคณิตศาสตร์หรือใช้ในการเปรียบเทียบ โดยแทนที่หน่วยความจำนั้นด้วยการกำหนดชื่อขึ้นโดยต้องมีการประกาศการใช้งานและกำหนดชนิดข้อมูลของตัวแปรก่อนใช้งาน



4.3.1 กฎในการตั้งชื่อตัวแปร

จะเหมือนกับมาตรฐาน ANSI-C ซึ่งมีกฎในการตั้งชื่อ ดังนี้

1. ต้องไม่ตั้งชื่อซ้ำกับคำหลัก (Keywords) ของภาษาซี มาตรฐานและภาษาซีสำหรับไมโครคอนโทรลเลอร์
2. ชื่อที่ตั้งประกอบด้วยตัวอักษรภาษาอังกฤษ ซึ่งอาจเป็นตัวอักษรพิมพ์เล็ก ตัวอักษรพิมพ์ใหญ่ ตัวเลข โดยอักษรตัวแรกต้องเป็นตัวอักษรหรือเส้นใต้อักขระเท่านั้น

อย่างไรก็ตามการตั้งชื่อตัวแปรควรตั้งชื่อให้สื่อกับวัตถุประสงค์ของการทำงานของตัวแปร เพื่อสื่อความหมายที่ตรงกัน

คำหลักในภาษาซี ตามมาตรฐาน ANSI-C



auto	break	case
char	const	continue
default	do	double
else	enum	extern
float	for	goto
if	int	long
register	return	short
signed	sizeof	static
struct	switch	typedef
union	unsigned	void
volatile	while	-

ที่มา: ARMKIEL Microcontrollers Tools, 2014

คำหลักในภาษาซี ตามมาตรฐาน ANSI-C



<code>_at_</code>	<code>far</code>	<code>sbit</code>
<code>alien</code>	<code>idata</code>	<code>sfr</code>
<code>bdata</code>	<code>interrupt</code>	<code>sfr16</code>
<code>bit</code>	<code>large</code>	<code>small</code>
<code>code</code>	<code>pdata</code>	<code>_task_</code>
<code>compact</code>	<code>_priority_</code>	<code>using</code>
<code>data</code>	<code>reentrant</code>	<code>xdata</code>

ที่มา: ARMKIEL Microcontrollers Tools, 2014

4.3.2 การประกาศใช้งานตัวแปร



1. การประกาศใช้งานตัวแปรที่เกี่ยวข้องกับการใช้งานหน่วยความจำ
2. การประกาศใช้งานตัวแปรที่เกี่ยวข้องกับเรจิสเตอร์ฟังก์ชันพิเศษ

1. การประกาศใช้งานตัวแปรที่เกี่ยวข้องกับการใช้งานหน่วยความจำ



```
data_type [memory_space] variable_name;
```

โดยที่ data_type	คือชนิดข้อมูลที่กำหนดให้กับตัวแปร
memory_space	คือพื้นที่หน่วยความจำของตัวแปร
variable_name	คือชื่อตัวแปรที่ประกาศ



ตัวแปรที่เกี่ยวข้องกับหน่วยความจำที่ใช้งานที่สำคัญ มีดังนี้

- 1) การใช้งานหน่วยความจำโปรแกรม
 - ใช้หน่วยความจำชนิด code
- 2) การใช้งานหน่วยความจำข้อมูลภายใน
 - ใช้หน่วยความจำชนิด data, idata และ bdata
- 3) การใช้งานหน่วยความจำข้อมูลภายนอก
 - ใช้หน่วยความจำชนิด data, pdata และ xdata

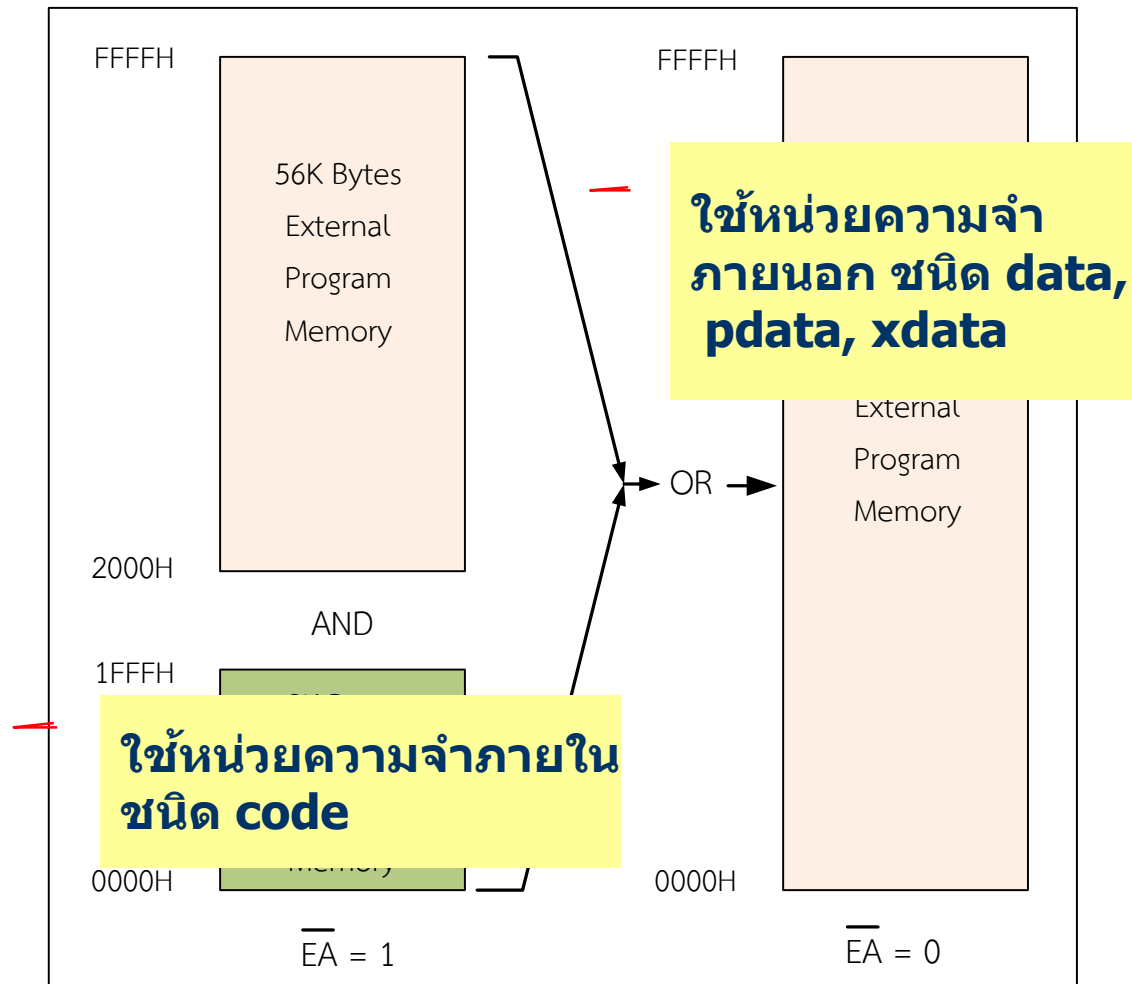
ประเภทของหน่วยความจำที่เกี่ยวข้องกับการประกาศตัวแปร



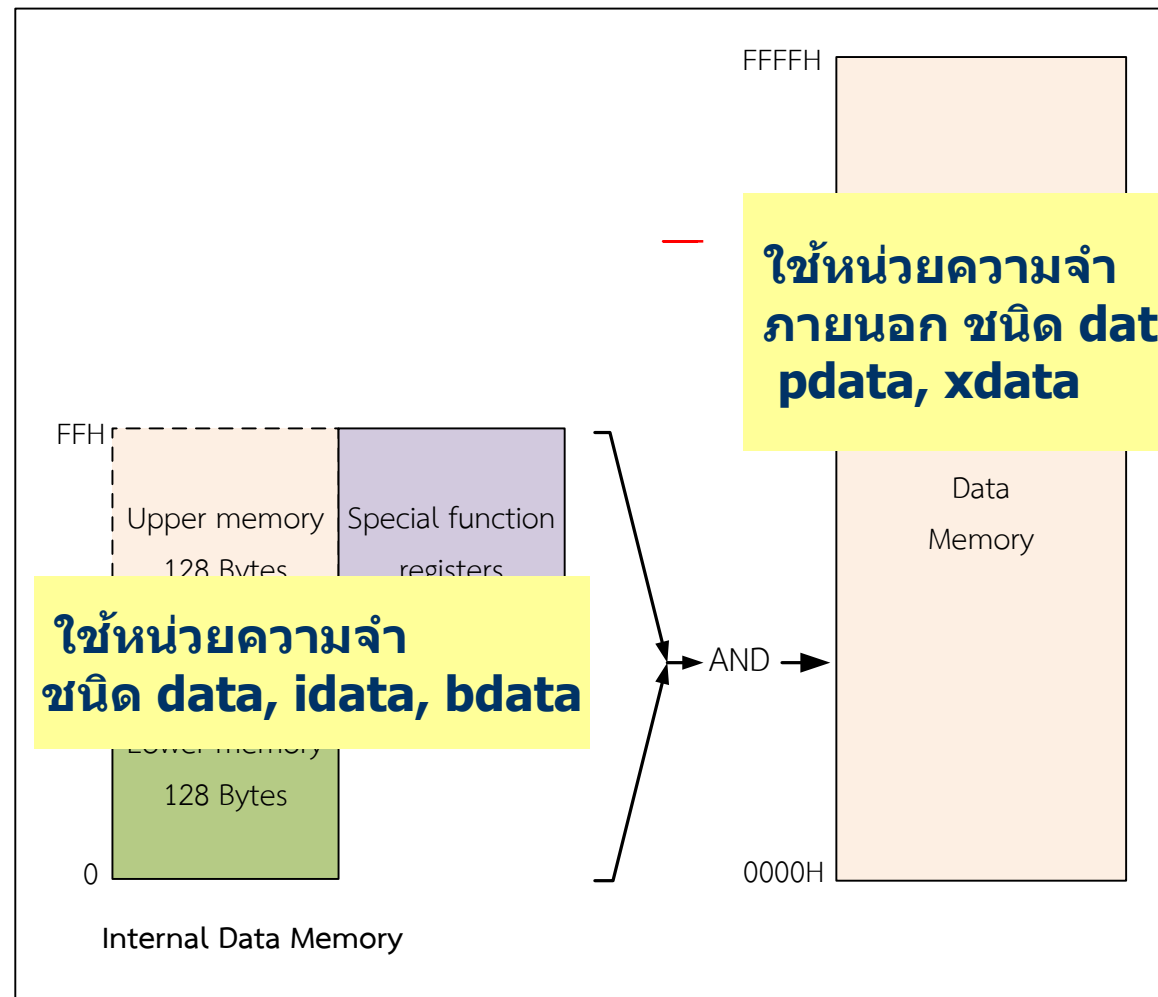
หน่วยความจำ	รายละเอียด
code	หน่วยความจำโปรแกรม (64 KBytes)
data	หน่วยความจำข้อมูลภายใน สามารถเข้าถึงแบบอั่งอิงโดยตรง (128 bytes)
idata	หน่วยความจำข้อมูลภายใน สามารถเข้าถึงแบบอั่งอิงโดยทางอ้อม (256 bytes)
bdata	หน่วยความจำข้อมูลภายใน สามารถเข้าถึงได้ระดับบิต (16 bytes)
xdata	หน่วยความจำข้อมูลภายนอก สามารถเข้าถึงได้ระดับบิต (64 KBytes)
pdata	หน่วยความจำข้อมูลภายนอกแบบเพจ (256 KBytes)

ที่มา: ARMKIEL Microcontrollers Tools, 2014

หน่วยความจำโปรแกรมของไมโครคอนโทรลเลอร์เบอร์ AT89C52



หน่วยความจำข้อมูลของไมโครคอนโทรลเลอร์เบอร์ AT89C52





ตัวอย่างที่ 4.8 โปรแกรมการประกาศใช้งานตัวแปร

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int i1, i2;
5      int i3 = 20;
6      float f1, f2;
7      char ch1, ch2;
8      i2 = 5;
9      i1 = i2;
10     i3 = i3+ i1;
11     f1 = 12.5;
12     f2 = 20.7;
13     f2 = f2 + f1;
14     ch1 = 'a';
15     ch2 = 'b';
16 }
```



```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int i1, i2;
5      int i3 = 20;
6      float f1, f2;
7      char ch1, ch2;
8      i2 = 5;
9      i1 = i2;
10     i3 = i3+ i1;
11     f1 = 12.5;
12     f2 = 20.7;
13     f2 = f2 + f1;
14     ch1 = 'a';
15     ch2 = 'b';
16 }
```

ผลลัพธ์ของการรันโปรแกรม

Name	Location/Value	Type
MAIN	C:0x0942	
i1	5	int
i2	5	int
i3	25	int
f1	12.5	float
f2	33.2	float
ch1	97 'a'	char
ch2	98 'b'	char



ตัวอย่างที่ 4.9 โปรแกรมการใช้งานตัวแปรแบบระบุพื้นที่หน่วยความจำโปรแกรม

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      char code msg1[] = "Keil C51";
5      code char table[5] = {'0','1','2','3','4'};
6  }
```

การใช้งานตัวแปรแบบระบุพื้นที่หน่วยความจำโปรแกรม



ผลลัพธ์ของการรันโปรแกรม

```
1 #include <REGX52.H>
2 void main (void)
3 {
4     char code msg1[] = "Keil C51";
5     code char table[5] = {'0', '1', '2', '3', '4'};
6 }
```

Name	Value	Type
msg1	C:0x0003 "Keil C51"	array[9] of char
[0]	0x4B 'K'	char
[1]	0x65 'e'	char
[2]	0x69 'i'	char
[3]	0x6C 'l'	char
[4]	0x20 ' '	char
[5]	0x43 'C'	char
[6]	0x35 '5'	char
[7]	0x31 '1'	char
[8]	0x00	char
table	C:0x000C "01234"	array[5] of char
[0]	0x30 '0'	char
[1]	0x31 '1'	char
[2]	0x32 '2'	char
[3]	0x33 '3'	char
[4]	0x34 '4'	char



ตัวอย่างที่ 4.10 โปรแกรมการใช้งานตัวแปรแบบระบุพื้นที่หน่วยความจำข้อมูล

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      char msg1[] = "Keil C51";
5      char table[5] = {'0', '1', '2', '3', '4'};
6  }
```

การใช้งานตัวแปรแบบไม่ได้รับพื้นที่หน่วยความจำโปรแกรม



ผลลัพธ์ของการรันโปรแกรม

```
1 #include <REG52.H>
2 void main (void)
3 {
4     char msg1[] = "Keil C51";
5     char table[5] = {'0','1','2','3','4'};
6 }
```

Name	Location/Value	Type
MAIN	C:0x08F6	
msg1	D:0x08 "Keil C51"	array[9] of char
[0]	0x4B 'K'	char
[1]	0x65 'e'	char
[2]	0x69 'i'	char
[3]	0x6C 'l'	char
[4]	0x20 ' '	char
[5]	0x43 'C'	char
[6]	0x35 '5'	char
[7]	0x31 '1'	char
[8]	0x00	char
table	D:0x11 "01234"	array[5] of char
[0]	0x30 '0'	char
[1]	0x31 '1'	char
[2]	0x32 '2'	char
[3]	0x33 '3'	char
[4]	0x34 '4'	char



เปรียบเทียบการใช้งานตัวแปร

ตัวอย่างที่ 4.9 โปรแกรมการใช้งานตัวแปรแบบระบุพื้นที่หน่วยความจำโปรแกรม

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      char code msg1[] = "Keil C51";
5      code char table[5] = {'0', '1', '2', '3', '4'};
6  }
```

ตัวอย่างที่ 4.10 โปรแกรมการใช้งานตัวแปรแบบระบุพื้นที่หน่วยความจำข้อมูล

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      char msg1[] = "Keil C51";
5      char table[5] = {'0', '1', '2', '3', '4'};
6  }
```

2. การประกาศใช้งานตัวแปรที่เกี่ยวกับเรจิสเตอร์ฟังก์ชันพิเศษ



ตัวแปรที่เกี่ยวกับเรจิสเตอร์ฟังก์ชันพิเศษ ใช้เฉพาะสำหรับโปรแกรม C51 เท่านั้น เนื่องจากการประกาศใช้งานตัวแปรดังกล่าว เกี่ยวข้องกับบิตข้อมูลของไมโครคอนโทรลเลอร์ MCS-51 โดยใช้ชนิดของข้อมูลแบบ bit, sbit และ sfr

2. การประกาศใช้งานตัวแปรที่เกี่ยวกับเรจิสเตอร์ฟังก์ชันพิเศษ



ตัวอย่างที่ 4.11 โปรแกรมการใช้งานตัวแปรที่เกี่ยวกับเรจิสเตอร์ฟังก์ชันพิเศษ

```
1  #include <REGX52.H>
2  sbit mybit = P1^5;
3  void main (void)
4  {
5      mybit = 0;
6      if (mybit==1)
7          P0 = 0x55;
8      else
9          P2 = 0xAA;
10 }
```

ผลลัพธ์ของการรันโปรแกรม

Parallel Port 1		Parallel Port 2	
Port 1		Port 2	
P1:	0xDF	P2:	0xAA
Pins:	0xDF	Pins:	0xAA



4.4 แถวลำดับและสายอักขระสำหรับ โปรแกรม C51

4.4.1 แกลวลำดับ



แกลวลำดับ (Arrays) คือตัวแปรชุดหรือตัวแปรที่กำหนดลำดับของตัวแปรโดยมีชื่อเหมือนกันแต่สามารถเก็บข้อมูลได้มากเท่ากับจำนวนลำดับที่ได้กำหนดไว้ให้กับตัวแปร โดยการอ้างถึงตัวแปรแกลวลำดับแต่ละตัวทำได้โดยการอ้างลำดับของตัวแปรนั้น



ตัวอย่างแถวลำดับของจำนวนเต็ม

ตำแหน่งของแถวลำดับ

ชื่อของแถวลำดับ

Num[0]	1
Num[1]	2
Num[2]	3
Num[3]	4
Num[4]	5
Num[5]	6
Num[6]	7
Num[7]	8
Num[8]	9
Num[9]	10

$$\text{data-type} \quad \text{array-name} [n] = \{\text{value list}\}; \quad (4)$$

(1)	คือรูปแบบการประกาศใช้งานแถวลำดับขนาด 1 มิติ
(2)	คือรูปแบบการประกาศใช้งานแถวลำดับขนาด 2 มิติ
(3)	คือรูปแบบการประกาศใช้งานแถวลำดับขนาด n มิติ
(4)	คือรูปแบบการประกาศใช้งานและกำหนดค่าแถวลำดับ
data_type	คือชนิดข้อมูลที่กำหนดให้กับตัวแปร
array-name[n]	คือชื่อตัวแปร
n, m	คือจำนวนเต็มที่กำหนดขนาดของแถวลำดับ

2. การใช้งานแถวลำดับในกรณีที่มีตัวแปรหลายตัว



โดยทำการรวบรวมข้อมูลแบบเดียวกันมากกว่าหนึ่งตัวขึ้นไปให้อยู่ในแถวเดียวกัน ตัวอย่างที่เกี่ยวข้องในชีวิตประจำวันที่มีการนำแถวลำดับไปใช้งานคือตารางข้อมูล ยกตัวอย่างเช่น ตารางแสดงคะแนนสอบนักเรียน

เลขที่นั่งสอบ นักเรียน	ตำแหน่งของแถว ลำดับ	คะแนนที่ได้
001	0	70.5
002	1	89.0
003	2	87.5
...	...	...
050	49	64.2

2. การใช้งานแถวลำดับในกรณีที่มีตัวแปรหลายตัว



ตัวอย่างที่ 4.12 โปรแกรมการใช้งานแถวลำดับ

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int i, num [10];
5      for(i=0;i<10;i++)
6      {
7          num[i] = i;
8      }
9      while (1);
10 }
```

2. การใช้งานแถวลำดับในกรณีที่มีตัวแปรหลายตัว



ผลลัพธ์ของการรันโปรแกรม

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int i, num [10];
5      for(i=0;i<10;i++)
6      {
7          num[i] = i;
8      }
9      while (1);
10 }
```

Name	Location/Value	Type
MAIN	C:0x0800	
i	10	int
num	D:0x08	array[10] of int
[0]	0	int
[1]	1	int
[2]	2	int
[3]	3	int
[4]	4	int
[5]	5	int
[6]	6	int
[7]	7	int
[8]	8	int
[9]	9	int

2. การใช้งานแถวลำดับในกรณีที่มีตัวแปรหลายตัว



ตัวอย่างที่ 4.13 โปรแกรมการใช้งานแถวลำดับ

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int num [3][3];
5      int i, j;
6
7      for(i=0;i<3;i++)
8      {
9          for(j=0;j<3;j++)
10         {
11             num [i][j] = i+j;
12         }
13     }
14     while (1);
15 }
```

2. การใช้งานแถวลำดับในกรณีที่มีตัวแปรหลายตัว



ผลลัพธ์ของการรันโปรแกรม

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int num [3][3];
5      int i, j;
6
7      for(i=0;i<3;i++)
8      {
9          for(j=0;j<3;j++)
10         {
11             num [i][j] = i+j;
12         }
13     }
14     while (1);
15 }
```

Name	Location/Value	Type
MAIN	C:0x0800	
num	D:0x08	array[3][3] of int
[0]	D:0x08	array[3] of int
[0]	0	int
[1]	1	int
[2]	2	int
[1]	D:0x0E	array[3] of int
[0]	1	int
[1]	2	int
[2]	3	int
[2]	D:0x14	array[3] of int
[0]	2	int
[1]	3	int
[2]	4	int
i	3	int
j	3	int

4.4.2 สายอักขระ



สายอักขระ (Strings) คือแถวลำดับของตัวแปรชนิดตัวอักษรขนาด 1 มิติ เนื่องจากภาษาซีมาตรฐานจะไม่มีชนิดข้อมูลสายอักขระจะมีเพียงชนิดข้อมูลชนิดตัวอักษรเท่านั้น ดังนั้นในการสร้างและใช้งานสายอักขระจึงเป็นการนำตัวแปรชนิดตัวอักษรมารวมกันในรูปแบบของแถวลำดับที่มีชนิดข้อมูลเป็น char

4.4.2 สายอักขระ



ในการใช้งานสายอักขระโดยทั่วไปต้องปิดด้วย null character (มีรหัส ASCII เท่ากับ “\0”) โดยการกำหนดรหัส “\0 ตัวอย่างเช่น

```
char str [16];           // สายอักขระ str ขนาด 15 อักขระ

char msg [5] [11];       // สายอักขระ msg จำนวน 5 ตัว แต่ละตัวมี
ขนาดเท่ากับ 10 อักขระ

char str1 [6] = {'a', 'b', 'c', 'd', 'e'}; // สายอักขระ str1 ขนาด 5 อักขระ

char str2 [ ] = "ABCDEF"; // ตัวแปรสตริง str2 ประกอบไป
ด้วยอักขระดังนี้ 'A', 'B', 'C', 'D', 'E', 'F'
```



ตัวอย่างที่ 4.14 โปรแกรมการใช้งานสายอักขระ

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      unsigned char mynum[] = "ABCDEF";
5      unsigned char z;
6      for (z=0; z<=6; z++)
7          P1 = mynum[z];
8      while (1);
9  }
```



ผลลัพธ์ของการรันโปรแกรม

```
1  #include <REGX52.H>
2  void main (void)
3  {
4  unsigned char mynum[] = "ABCDEF";
5  unsigned char z;
6  for (z=0; z<=6; z++)
7  P1 = mynum[z];
8  while (1);
9  }
```

Name	Location/Value	Type
MAIN	C:0x08F6	
mynum	D:0x08 "ABCDEF"	array[7] of uchar
[0]	0x41 'A'	uchar
[1]	0x42 'B'	uchar
[2]	0x43 'C'	uchar
[3]	0x44 'D'	uchar
[4]	0x45 'E'	uchar
[5]	0x46 'F'	uchar
[6]	0x00	uchar
z	0x07	uchar



4.5 ตัวชี้

ตัวชี้ (Pointer) คือตัวแปรที่ใช้ในการชี้ไปยังแอดเดรสของตัวแปรอื่นเพื่อที่จะเข้าถึงตัวแปรนั้นได้โดยผ่านทางตัวชี้ข้อมูล ดังนั้นตัวชี้ก็คือตัวแปรที่ทำหน้าที่ในการเก็บแอดเดรสของตัวแปรที่อยู่ในหน่วยความจำนั่นเอง

การใช้งานตัวชี้สำหรับโปรแกรม C51 มี 2 รูปแบบ

- ตัวชี้ทั่วไป (Generic pointers)
- ตัวชี้หน่วยความจำพิเศษ (Memory-specific pointers)

4.5.1 ตัวชี้ทั่วไป



```
data_type      * variable_name;
```

โดยที่ data_type คือชนิดข้อมูลที่กำหนดให้กับตัวแปร
* เป็นเครื่องหมายแสดงว่าตัวแปรที่ประกาศ
ตามหลังคือตัวชี้
variable_name คือชื่อตัวแปรชนิดตัวชี้



4.5.1 ตัวชี้ทั่วไป

ชนิดข้อมูลที่กำหนดการใช้งานตัวชี้อาจกำหนดเป็นแบบข้อมูลใดๆ ก็ได้ ตัวอย่างเช่น

```
int *iptr;           // ตัวชี้ไปยังแหล่งข้อมูลชนิดจำนวนเต็ม
char *cptr;          // ตัวชี้ไปยังแหล่งข้อมูลชนิดตัวอักษร
```

- ในการใช้งานตัวชี้ นั้นจะเกี่ยวข้องกับตัวดำเนินการสองตัวคือ
- เครื่องหมายดอกจัน (*) ใช้ในการสร้างตัวชี้และใช้ในการชี้ไปยังตำแหน่งที่ตัวชี้เก็บอยู่เพื่อนำข้อมูลในตัวแปรที่ชี้ไปกลับมา
 - เครื่องหมายแอมเพอแอนด์ (&) คือแอดเดรสของตัวแปรที่จะกำหนดให้กับตัวชี้ นั่นเอง

รูปแบบของตัวชี้ โดยใช้คำสั่ง `int *iptr;`



นิพจน์	คำอธิบาย
<code>iptr</code>	ชื่อตัวแปรที่ทำหน้าที่เป็นตัวชี้ นิพจน์นี้ให้ค่าเป็น <code>int</code> หรือ <code>unsigned int</code> ซึ่งเป็นที่อยู่ที่ตัวชี้นี้ชี้ไป
<code>&iptr</code>	นิพจน์นี้ระบุที่อยู่ของตัวชี้ <code>iptr</code> ในหน่วยความจำ
<code>*iptr</code>	นิพจน์นี้ให้ค่าข้อมูลที่เก็บอยู่ในที่อยู่ที่ตัวชี้ <code>iptr</code> กำลังชี้ไป



ตัวอย่างที่ 4.15 โปรแกรมการใช้งานตัวชี้

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int *iptr;
5      int i,j;
6      i = 20;
7      iptr = &i;
8      j = *iptr;
9      while (1);
10 }
```

ผลลัพธ์ของการรันโปรแกรม

Name	Location/Value	Type
MAIN	C:0x082B	
iptr	I:0x0B	ptr3
i	20	int
j	20	int

4.5.2 ตัวชี้หน่วยความจำพิเศษ



```
data_type      memory-type      * variable_name;
```

โดยที่ memory type

คือชนิดของหน่วยความจำ

4.5.2 ตัวชี้หน่วยความจำพิเศษ



ชนิดของหน่วยความจำที่กำหนดการใช้งานตัวชี้ อาจกำหนดเป็นหน่วยความจำใดๆ ก็ได้ ยกตัวอย่างเช่น

char data *str;	// ตัวชี้ที่ชี้ไปยังชนิดข้อมูลตัวอักษรเก็บในหน่วยความจำข้อมูลภายใน
int xdata *number;	// ตัวชี้ที่ชี้ไปยังชนิดข้อมูลจำนวนเต็มเก็บในหน่วยความจำข้อมูลภายนอก
long code *cost;	// ตัวชี้ที่ชี้ไปยังชนิดข้อมูลจำนวนเต็มช่วงยาวเก็บในหน่วยความจำโปรแกรม



ตัวอย่างที่ 4.16 โปรแกรมการใช้งานตัวชี้หน่วยความจำพิเศษ

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int idat[5]= {10,20,30,40,50};
5      int j[5]= {0,0,0,0,0};
6      int *iptr;
7      int i;
8      iptr = idat;
9      for(i=0;i<5;i++)
10     {
11         j[i] = *(iptr+i);
12     }
13     while (1);
14 }
```



ผลลัพธ์ของการรันโปรแกรม

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      int idat[5]= {10,20,30,40,50};
5      int j[5]= {0,0,0,0,0};
6      int *iptr;
7      int i;
8      iptr = idat;
9      for(i=0;i<5;i++)
10     {
11         j[i] = *(iptr+i);
12     }
13     while (1);
14 }
```

Name	Location/Value	Type
MAIN	C:0x092E	
idat	D:0x08	array[5] of int
[0]	10	int
[1]	20	int
[2]	30	int
[3]	40	int
[4]	50	int
j	D:0x12	array[5] of int
[0]	10	int
[1]	20	int
[2]	30	int
[3]	40	int
[4]	50	int
iptr	I:0x08	ptr3
[0]	10	int
i	5	int



4.6 ตัวแปรโครงสร้างและยูเนียน

4.6.1 ตัวแปรโครงสร้าง



ตัวแปรโครงสร้าง (Structures) เป็นตัวแปรประเภทหนึ่ง ที่รวบรวมตัวแปรที่เหมือนหรือแตกต่างกันมาไว้รวมกัน ซึ่งอยู่ภายใต้ชื่อเดียวกัน ถ้าต้องการสร้างข้อมูลแบบโครงสร้างต้องทำการกำหนดว่าจะใช้ข้อมูลแบบใดบ้างที่ต้องการเก็บไว้ในโครงสร้าง ซึ่งข้อมูลเหล่านี้เรียกว่าข้อมูลสมาชิก เพราะเป็นองค์ประกอบของโครงสร้าง จุดประสงค์ของการใช้งานข้อมูลแบบโครงสร้างคือ การนำข้อมูลหลายๆ รูปแบบมาเก็บไว้รวมกัน เพื่อประโยชน์ในการใช้งานตัวแปรที่ต้องเกี่ยวข้องกัน

4.6.1 ตัวแปรโครงสร้าง



การประกาศตัวแปรประเภทโครงสร้างประกาศได้ 2 รูปแบบ

1. รูปแบบที่ 1

```
struct tag-name {  
    type element1;  
    type element2;  
    type element3;  
    ...  
} variable-list;
```

โดยที่	tag-name	คือชื่อของโครงสร้างหรือป้ายโครงสร้าง
	type element	คือชนิดของตัวแปรภายในตัวแปรโครงสร้าง
	variable-list	คือชื่อที่ใช้ในการอ้างอิงตัวแปรโครงสร้างจะกำหนดหรือไม่ก็ได้

4.6.1 ตัวแปรโครงสร้าง



2. รูปแบบที่ 2

```
struct tag-name {  
    type element1;  
    type element2;  
    type element3;  
    ...  
};
```

โดยที่	tag-name	คือชื่อของโครงสร้างหรือป้ายโครงสร้าง
	type element	คือชนิดของตัวแปรภายในตัวแปรโครงสร้าง
	variable-list	คือชื่อที่ใช้ในการอ้างอิงตัวแปรโครงสร้างจะกำหนดหรือไม่ก็ได้



ตัวอย่างที่ 4.17 โปรแกรมการใช้งานตัวแปรโครงสร้าง

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      struct person{
5          int ID;
6          char Name;
7          int Age;
8      } ;
9
10     struct person MrA,MrB;
11     MrA.ID = 1;
12     MrA.Name = 'A';
13     MrA.Age = 20;
14     MrB.ID = 2;
15     MrB.Name = 'B';
16     MrB.Age = 25;
17     while (1);
18 }
```

การใช้งานตัวแปรโครงสร้าง



ผลลัพธ์ของการรันโปรแกรม

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      struct person{
5          int ID;
6          char Name;
7          int Age;
8      } ;
9
10     struct person MrA,MrB;
11     MrA.ID = 1;
12     MrA.Name = 'A';
13     MrA.Age = 20;
14     MrB.ID = 2;
15     MrB.Name = 'B';
16     MrB.Age = 25;
17     while (1);
18 }
```

Name	Location/Value	Type
MAIN	C:0x0800	
MrA	D:0x08	struct person
ID	1	int
Name	65 'A'	char
Age	20	int
MrB	D:0x0D	struct person
ID	2	int
Name	66 'B'	char
Age	25	int

4.6.2 ตัวแปรยูเนียน



การประกาศตัวแปรแบบยูเนียน (Union) มีลักษณะที่คล้ายกับตัวแปรแบบโครงสร้าง แต่การประกาศใช้งานตัวแปรแบบยูเนียนนี้จะประหยัดการใช้งานหน่วยความจำได้ เนื่องจากตัวแปรแบบยูเนียนจะใช้หน่วยความจำในตำแหน่งเดียวกันในเวลาหนึ่ง (ไม่ได้ใช้งานพร้อมกัน) เนื่องจากการประกาศหน่วยความจำที่เก็บข้อมูลต่างชนิดกันไว้

4.6.2 ตัวแปรยูเนียน



การประกาศตัวแปรยูเนียนประกาศได้ 2 รูปแบบ

1. รูปแบบที่ 1

```
union tag-name {  
    type element1;  
    type element2;  
    type element3;  
    ...  
} variable-list;
```

โดยที่	tag-name	คือชื่อของยูเนียนหรือป้ายยูเนียน
	type element	คือชนิดของตัวแปรภายในตัวแปรยูเนียนหรือสมาชิกนั่นเอง
	variable-list	คือชื่อที่ใช้ในการอ้างอิงตัวแปรยูเนียนจะกำหนดหรือไม่ก็ได้

4.6.2 ตัวแปรยูเนียน



2. รูปแบบที่ 2

```
union tag-name {  
    type element1;  
    type element2;  
    type element3;  
    ...  
};
```

โดยที่	tag-name	คือชื่อของยูเนียนหรือป้ายยูเนียน
	type element	คือชนิดของตัวแปรภายในตัวแปรยูเนียนหรือสมาชิกนั่นเอง
	variable-list	คือชื่อที่ใช้ในการอ้างอิงตัวแปรยูเนียนจะกำหนดหรือไม่ก็ได้



ตัวอย่างที่ 4.18 โปรแกรมการใช้งานตัวแปรยูเนียน

```
1  #include <REGX52.H>
2  void main (void)
3  {
4      union person{
5          int ID;
6          char Name;
7          int Age;
8      } MrA,MrB;
9
10     MrA.ID = 1;
11     MrA.Name = 'A';
12     MrA.Age = 20;
13     MrB.ID = 2;
14     MrB.Name = 'B';
15     MrB.Age = 25;
16     while (1);
17 }
```



ผลลัพธ์ของการรันโปรแกรม

ผลการทำงานของโปรแกรมได้ผลลัพธ์เช่นเดียวกับตัวแปรโครงสร้างในตัวอย่างที 4.17 แต่จะประหยัดการใช้งานหน่วยความจำได้มากกว่า เนื่องจากตัวแปรยูเนียนจะใช้หน่วยความจำในตำแหน่งเดียวกันในการเก็บข้อมูล



4.7 สรุป

ได้นำเสนอพื้นฐานการพัฒนาโปรแกรมด้วยภาษาซี สำหรับโปรแกรม C51 การเลือกใช้งานชนิดของข้อมูลให้เหมาะสมกับไมโครคอนโทรลเลอร์ที่มีหน่วยความจำค่อนข้างจำกัด และการใช้งานตัวแปรชนิดต่างๆ ที่สำคัญได้แก่

- แกว่ลำดับและสายอักขระ
- ตัวชี้
- ตัวแปรโครงสร้างและยูเนียน

สำหรับเนื้อหาในบทถัดไปได้อธิบายถึงตัวดำเนินการซึ่งเป็นตัวเชื่อมในการเขียนโปรแกรมเพื่อหาผลลัพธ์ที่ต้องการในการไมโครคอนโทรลเลอร์ไปประยุกต์ใช้งานต่อไป



4.8 แบบฝึกหัดท้ายบท



4.8 แบบฝึกหัดท้ายบท

1. จงอธิบายการประกาศตัวแปรชนิดข้อมูลตัวอักษร ตัวเลขจำนวนเต็ม และทศนิยม
2. จงอธิบายข้อแตกต่างระหว่างการประกาศตัวแปรแบบ bit sbit และ sfr
3. การตั้งชื่อตัวแปรในข้อใดไม่ถูกต้องตามกฎหมายในการตั้งชื่อตัวแปรสำหรับโปรแกรม C51
 - 3.1) Result99
 - 3.2) small
 - 3.3) My name
 - 3.4) _test
 - 3.5) 10Points
 - 3.6) @temperature
 - 3.7) sizeof

4.8 แบบฝึกหัดท้ายบท



4. จงอธิบายประเภทของหน่วยความจำที่เกี่ยวข้องกับการประกาศตัวแปรสำหรับโปรแกรม C51
ดังนี้

4.1) หน่วยความจำ code

4.2) หน่วยความจำ data

4.3) หน่วยความจำ idata

4.4) หน่วยความจำ bdata

4.5) หน่วยความจำ xdata

4.6) หน่วยความจำ pdata

4.8 แบบฝึกหัดท้ายบท



5. จงเขียนโปรแกรมเพื่อประกาศตัวแปรแถวลำดับตามที่กำหนด

5.1) แถวลำดับชื่อว่า ID มีสมาชิก 10 ตัว กำหนดค่าเริ่มต้นให้มีค่าเท่ากับ 100 , 200, 300, 400, 500, 600, 700, 800, 900 และ 1000 ตามลำดับ

5.2) แถวลำดับชื่อว่า Key ขนาด 2 มิติ มีสมาชิกจำนวน 16 ตัว กำหนดค่าเริ่มต้นทุกตัวเท่ากับ :

5.3) แถวลำดับมีชื่อว่า Alpha มีสมาชิกจำนวน 7 ตัว กำหนดค่าเริ่มต้นเท่ากับ 'K', 'e', 'i', 'L', 'C', '5' และ '1' ตามลำดับ

4.8 แบบฝึกหัดท้ายบท



6. กำหนดให้ a และ p เป็นแถวลำดับและตัวชี้ตามลำดับ โดยกำหนดค่าดังนี้

```
int a[ ]= {1,-4,4,3,10,25,19,0,2};
```

```
int *p = &a[3] ;
```

จงหาค่าของนิพจน์ต่อไปนี้

6.1) $*(p+2)$;

6.2) $p[-2]$;

6.3) $a[*p+1]$;

4.8 แบบฝึกหัดท้ายบท



7. จงเขียนโปรแกรมโดยใช้ ตัวแปรโครงสร้าง สำหรับบันทึกประวัตินักศึกษาโดยมีข้อมูลที่ต้องการบันทึกดังนี้ ชื่อ-นามสกุล รหัสนักศึกษา สาขาวิชา และคณะที่ศึกษา โดยให้แสดงผลดังตัวอย่าง ดังนี้

Name: Tony Barber

Student ID: 56010361503

Department: Electronics

Faculty: Engineering



ดอนสัน ปงผาบ. (2554). ปฏิบัติการไมโครคอนโทรลเลอร์และการประยุกต์ใช้งาน 1. กรุงเทพฯ: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น).

ทีมงานสมาร์ทเลิร์นนิ่ง. (2555). เริ่มต้นเรียนรู้ไมโครคอนโทรลเลอร์ MCS-51 ด้วยภาษา C. กรุงเทพฯ: สมาร์ทเลิร์นนิ่ง

ธีรวัฒน์ ประกอบผล. (2540). การประยุกต์ใช้งานไมโครคอนโทรลเลอร์. กรุงเทพฯ: สมาคมส่งเสริมเทคโนโลยี (ไทย-ญี่ปุ่น).

นคร ภัคดีชาติ และชัยวัฒน์ ลิ้มพรจิตรวิไล. (2550). ทดลองและใช้งานไมโครคอนโทรลเลอร์ MCS-51 ด้วยโปรแกรมภาษา C ฉบับ P89V51RD2. กรุงเทพฯ: อินโนเวทีฟ เอ็กเพอริเมนต์.

นวนดา สงวนวงศ์ทอง. (ม.ป.ป.). เอกสารประกอบการสอน วิชาการโปรแกรมคอมพิวเตอร์ 1. ค้นเมื่อ มีนาคม 1 2557 จาก http://www.it.tl.ac.th/~witsarut/subject_c.html

ประจัน พลังสันติกุล และชัยวัฒน์ ลิ้มพรจิตรวิไล. (2550). ปฏิบัติการไมโครคอนโทรลเลอร์ MCS-51 กับ Keil C51 คอมไพเลอร์. กรุงเทพฯ: อินโนเวทีฟ เอ็กเพอริเมนต์.

सानนท์ เจริญฉาย. (2546). การเขียนโปรแกรมและอัลกอริทึม. กรุงเทพฯ: มหาจุฬาลงกรณราชวิทยาลัย.



Architecture and Programming of 8051 MCUS. Retrieved March 1, 2014, from
<http://www.mikroe.com/products/view/267/architecture-and-programming-of-8051-mcu-s/>

ARMKIEL Microcontrollers Tools. **Cx51 User's Guide.** Retrieved April 1, 2014, from
<http://www.keil.com/support/man/docs/c51/>

Michael, P. (2001). **Patterns for time-triggered embedded systems: Building reliable applications with the 8051 family of microcontrollers.** New York: ACM Press Books.

Michael, P. (2002). **Embedded C.** London: Addison-Wesley Professional.

Microcontroller with 8K Bytes Flash AT89C52. Retrieved March 1, 2014, from
<http://www.atmel.com/images/doc0313.pdf>

Z80 CPU User Manual. Retrieved April 1, 2014, from
http://www.zilog.com/appnotes_download.php?FromPage=DirectLink&dn=UM0080&ft=User%20Manual&f=YUhSMGNEb3ZMM2QzZHK1NmFXeHZaeTVqYjIwdlpHOWpjeTk2T0RBdlZVMHdNRGd3TG5Ca1pnPT0=