

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Language for controlling the ARM processor

Lets you run sequences of instructions from Flash

Manipulate data variables in SRAM

C code is written in text files (extension “.c”)

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Tell the reader what the code does

Compiler ignores comments

Often have a company-standard copyright notice

`/* block comment */`

`// end of line comment`

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

SRAM-based data

int8 / int16 / int32

Simple numbers

int8: -128..127

int16: -32768..32767

uint8 / uint16 / uint32

Unsigned numbers

uint8: 0..255

uint16: 0..65535

Use memorable names

int8 ADCvalue; // good name

int8 myvar; // bad name

Assignment

=

`x=2`

Set x to 2

Arithmetic

+ - * /

`x+2`

Add up x and 2

Compound (shortcuts)

`+=` `-=` `*=` `/=`

`x*=2`

`x = x * 2`

Increment and Decrement

`++` `--`

`x++`

`x = x + 1`

Comparison

`==` `!=` `<` `<=` `>` `>=`

`x==2`

Compare x with 2

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Bundles of instructions

Better than copy-paste of similar code

Run functions by typing the name and parentheses

```
function1();
```

Pass values inside the parentheses

```
function2( 5 );
```

Get data back in the “return value”

```
pos = getPos();
```

Compare true with false

Zero means false

Non-zero means true, operators yield 1

Logical AND (&&) is true if BOTH are true

`0 && 1 yields 0`

`1 && 2 yields 1`

Logical OR (||) is true if EITHER are true

`0 || 1 yields 1`

`1 || 2 yields 1`

`0 || 0 yields 0`

Often use multiple comparisons

`((x > 7) && (y < 10))`

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Execute instructions if a condition is true

Format:

```
if( condition )  
{  
    /* Execute block */  
}
```

If the condition is not true, skip the block

Always use a block { ... }

Conditionals – “if” and “else”

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Alternative action

If the condition is not true, do something else

```
if( condition )  
{  
    /* Execute if true */  
}  
  
else  
{  
    /* Execute if false */  
}
```

Note: Indenting (tabbing) the code is good style

Loops – “while”

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Repeat instructions until the condition is not true

Format:

```
while( condition )  
{  
  
}
```

Condition typically the changes in the loop

```
while( x != 0 )  
{  
    x = ADC_GetResult16();  
    /* more code */  
}
```

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Three expressions

Format:

```
for( init; cond; incr )  
{  
  
}
```

Bundle loop controls into one statement

```
int i;  
for( i=0; i<100; i++ )  
{  
    /* Loop 100 times */  
}
```

```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Embedded systems never stop

Power on and run forever

Use never-false condition

```
while( 1 )
```

```
{  
  
}
```

Can also omit for-loop conditions

```
for( ; ; )
```

```
{  
  
}
```

Understanding Numbers – uint8

Bit #	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	== 0	(0x00)
	0	0	0	0	0	0	0	1	== 1	(0x01)
	0	0	0	0	0	0	1	0	== 2	(0x02)
	0	0	0	0	0	1	0	0	== 4	(0x04)
	0	0	0	0	1	0	0	0	== 8	(0x08)
	0	0	0	1	0	0	0	0	== 16	(0x10)
	0	0	1	0	0	0	0	0	== 32	(0x20)
	0	1	0	0	0	0	0	0	== 64	(0x40)
	1	0	0	0	0	0	0	0	== 128	(0x80)

Understanding Numbers – uint8

Bit #	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	== 0	(0x00)
	0	0	0	0	0	0	0	1	== 1	(0x01)
	0	0	0	0	0	0	1	0	== 2	(0x02)
	0	0	0	0	0	1	0	0	== 4	(0x04)
	0	0	0	0	1	0	0	0	== 8	(0x08)
	0	0	0	1	0	0	0	0	== 16	(0x10)
	0	0	1	0	0	0	0	0	== 32	(0x20)
	0	1	0	0	0	0	0	0	== 64	(0x40)
	1	0	0	0	0	0	0	0	== 128	(0x80)
	0	0	0	0	0	0	1	1	== 3	(0x03)
	0	0	0	0	1	1	1	1	== 15	(0x0F)

Understanding Numbers – uint8

Bit #	7	6	5	4	3	2	1	0		
	0	0	0	0	0	0	0	0	== 0	(0x00)
	0	0	0	0	0	0	0	1	== 1	(0x01)
	0	0	0	0	0	0	1	0	== 2	(0x02)
	0	0	0	0	0	1	0	0	== 4	(0x04)
	0	0	0	0	1	0	0	0	== 8	(0x08)
	0	0	0	1	0	0	0	0	== 16	(0x10)
	0	0	1	0	0	0	0	0	== 32	(0x20)
	0	1	0	0	0	0	0	0	== 64	(0x40)
	1	0	0	0	0	0	0	0	== 128	(0x80)
	0	0	0	0	0	0	1	1	== 3	(0x03)
	0	0	0	0	1	1	1	1	== 15	(0x0F)
	1	1	1	1	0	0	0	0	== 240	(0xF0)
	0	0	1	1	0	0	1	1	== 51	(0x33)

Compare the bit patterns

Bitwise AND (&) sets the bit if BOTH are 1

0x03 & 0x02

0	0	0	0	0	0	1	1
&	&	&	&	&	&	&	&
0	0	0	0	0	0	1	0
=	=	=	=	=	=	=	=
0	0	0	0	0	0	1	0

== 0x02

Compare the bit patterns

Bitwise AND (&) sets the bit if BOTH are 1

0x03 & 0x02 == 0x02

0	0	0	0	0	0	1	1
&	&	&	&	&	&	&	&
0	0	0	0	0	0	1	0
=	=	=	=	=	=	=	=
0	0	0	0	0	0	1	0

0x03 & 0x08 == 0x00

0	0	0	0	0	0	1	1
&							
0	0	0	0	0	1	0	0

Compare the bit patterns

Bitwise AND (&) sets the bit if BOTH are 1

0x03 & 0x02

0	0	0	0	0	0	1	1
&	&	&	&	&	&	&	&
0	0	0	0	0	0	1	0
=	=	=	=	=	=	=	=
0	0	0	0	0	0	1	0

== 0x02

0x03 & 0x08

0	0	0	0	0	0	1	1
&							
0	0	0	0	0	0	1	0

== 0x00

Bitwise OR (|) sets the bit if EITHER are 1

0x03 | 0x02

0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0

== 0x03

Compare the bit patterns

Bitwise AND (&) sets the bit if BOTH are 1

0x03 & 0x02

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

 == 0x02

 & & & & & & &

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

 = = = = = = = =

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

0x03 & 0x08

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

 &

0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

 == 0x00

Bitwise OR (|) sets the bit if EITHER are 1

0x03 | 0x02

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

 |

0	0	0	0	0	0	1	0
---	---	---	---	---	---	---	---

 == 0x03

0x03 | 0x08

0	0	0	0	0	0	1	1
---	---	---	---	---	---	---	---

 |

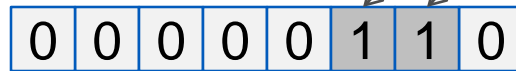
0	0	0	0	1	0	0	0
---	---	---	---	---	---	---	---

 == 0x0B (11)

<< move all the bits left (up)

>> move all the bits right (down)

0x03 << 1



== 0x06

<< move all the bits left (up)

>> move all the bits right (down)

0x03 << 1



== 0x06

0x03 << 2

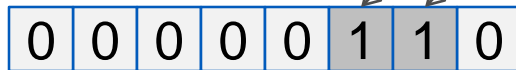
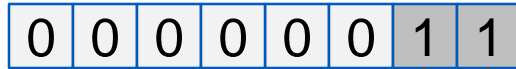


== 0x0C

<< move all the bits left (up)

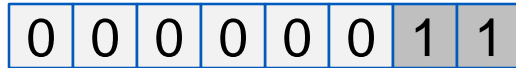
>> move all the bits right (down)

0x03 << 1



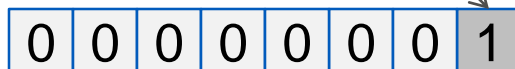
== 0x06

0x03 << 2



== 0x0C

0x03 >> 1



== 0x01

Consecutive group of variables

e.g. lookup tables

Use square brackets to create and use arrays

```
/* Array of four 8-bit signed integers */  
int8 buffer[4];  
  
/* set the specific element to 99 */  
buffer[2] = 99;
```

Arrays always start at zero

buffer[0] is the first element in the array

buffer[3] is the last element in the array

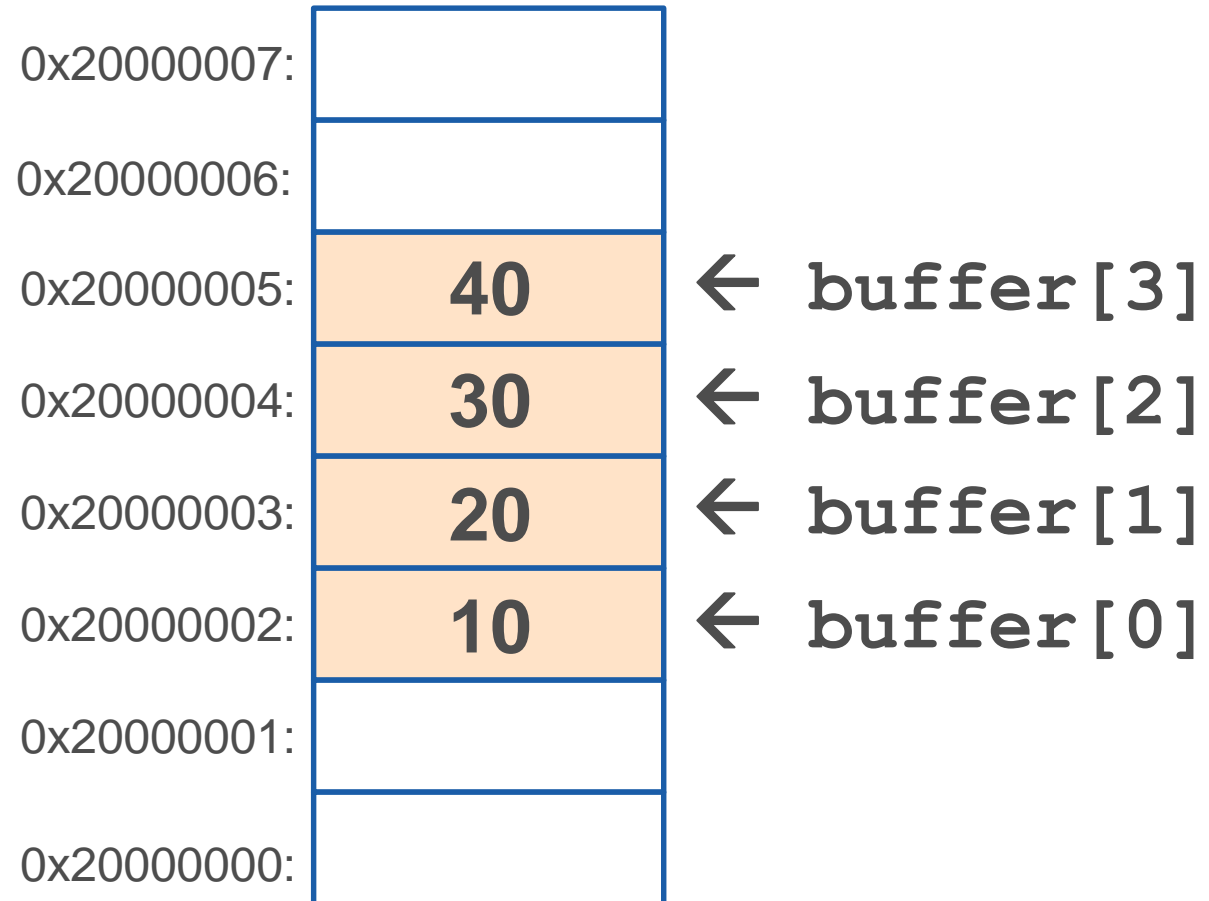
Strings are arrays of chars

Use double quotes "Print me"

Array Example

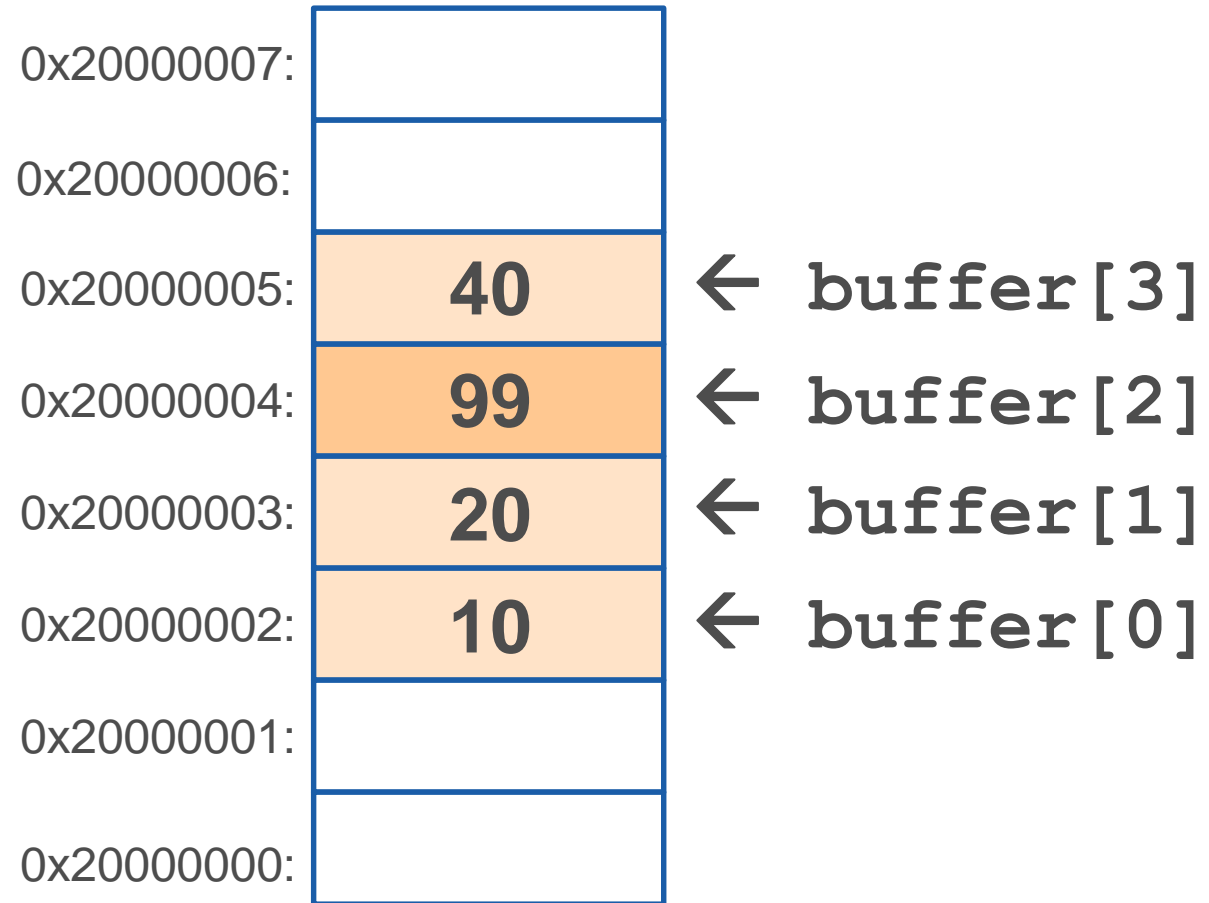
Initialize arrays with { comma-separated list }

```
int8 buffer[] = { 10, 20, 30, 40 };
```



Array Example

```
int8 buffer[] = { 10, 20, 30, 40 };  
buffer[2] = 99;
```



```
/* *****  
 * Copyright 2014, YFS Corporation. All rights reserved.  
 * ***** */  
  
#include <project.h>  
  
#define MYCHANNEL (0)  
  
/*  
 * Function Name: main  
 * Reads the ADC and displays the voltage on the LCD.  
 */  
int main()  
{  
    int16 output;  
  
    /* Start the components */  
    LCD_Start();  
    ADC_Start();  
  
    ADC_StartConvert(); // Start the ADC conversions  
  
    /* Display the value of ADC output on LCD */  
    LCD_Position( 0, 0 );  
    LCD_PrintString( "ADC_Output" );  
  
    while( 1 )  
    {  
        /* If ADC has data - read, convert to mV and print */  
        if( ADC_IsEndConversion( ADC_RETURN_STATUS ) )  
        {  
            output = ADC_GetResult16( MYCHANNEL );  
            output = ADC_CountsTo_mVolts( MYCHANNEL, output );  
  
            LCD_Position( 1, 0 );  
            LCD_PrintInt16( output );  
        }  
    }  
}
```

Simple text substitution

#define MYCHANNEL 0

#define MSG "Hello!"

Improve readability

Magic numbers

Improve quality

Need to change the ADC
channel?

Just one edit and rebuild

Cannot forget to change every
use

