



Click & Move

Function Blocks

Function Blocks

To the user, the FBs are crucial. These are the software equivalent of electronic chips. They contain inputs and outputs, with associated names and Data types. Each FB contains code (like a small program) to give it its functionality. The user only sees the interface, being the inputs and outputs. The code itself is hidden - this data encapsulation and hiding is crucial to separate the different levels of programming and maintenance.



Function Block Parts

The name of the function block is located inside the upper section of the block. The instance of the block is shown above the block. The instance name is formed from the name of the library containing the block and the numerical order the block was placed on the schematic. The blocks below were taken from the MATH library.

Input pins are displayed along the left side of the block and output pins are displayed along the right side. The name of the pin is shown inside the block next to the pin and the data type of the pin is shown below the name of the pin.



Addition

Dinc



Project library name: Math Debug: NO Functional init: YES Target list: PCW PLA PCL MAL PCI Compiler for PCW target: MGW, BCB, BC5, MVC Compiler for PLA target: GMU Compiler for PCL target: GAL Compiler for PCL target: MXI

Note: PCW - PC with Microsoft Windows PLA - AMC programmable servo drive platform A PCL - PC with Linux MAL - MACC (Motion Automation Controller Card) with Linux MGW - MinGW32 goc compiler (Version 3.4.2) BCB - Borland 5.5 command line compiler BCB - Borland 5.5 command line compiler GML - MicroBlaze compiler GXL - GCC Cross Compiler On Windows for x86 Linux GAL - ARM CortexA8 compiler

Function Block Library

The library contains a description of the function block, how it works and how it may be used.

The description indicates compatibility of the block with the various components of C&M and the data types supported by the function block pins.

Name	Direction	Init value	Data type	Const	Size	Debug mode	Description
In1	Input	-	TypeIn1	YES	-	NORMAL	
In2	Input		TypeIn2	YES	198	NORMAL	
Out	Output	120	TypeOut	YES	1	NORMAL	Out := In1 + In2

2. For the range of data types see C&M-MC help/Overview/Data types.

3. "-" in the "Init value" column means default value, see C&M-MC help/Overview/Data types.

In general, the addition operation is commutative, but there is an exception, adding two strings is not commutative. An operation is commutative if changing the order of the operands does not change the result.

Allowable types for TypeIn1, TypeIn2 and TypeOut: Int, Int8, Int16, Int32, Int64, UInt8, UInt16, UInt32, UInt64, Float32, Float64, CMString,PString256,PString32,Array TypeIn1, TypeIn2 and TypeOut can be different.

See also C&M-MC help/Overview/Data types in Desktop or start menu.

File Project Run Interface Virtual machine Collected C&M application Tools Settings V	/indow Help
🐧 3월 🎊 - 🐌 🗅 🛤 🔺 🔌 😢 🔤 🕨 🏘 🧟 📑 🗰 🗱 🕸 🤗	T Context-Sensitive help F1 C8M-MC help F1
	C&M+HMI C&H-MC help C&M relat Tops the C&M-MC help file. Errors Select the Desktop meru item, the toolbar button or open the dialog box, for whice Function 1 you want help and press the "F1" key. The desktop window must be selected Ucenses before setting an itemit Ucense. Demo Vid: A C&M-MC (Notion/Machine Control) application is represented by a set of Function Tutorial VI Block Diagrams. FBs may include embedded C++ user programs. Readme Show tips Show tops

Library Help

Use the C&M-MC Help selection on the C&M Desktop to list the C&M function blocks in the library.

G Back Forward Home Print Options		
nts Index Search Favorites	Function Block Library descriptions	
Introduction	Function Block Library descriptions	
General		
Rest C&M practices		
0 Overview	Use Ctrl+F to search within this chapter	
? Major components		
🗉 💑 User interfaces	Note that Function Block description (help) files can also be opened by using a right mouse click on a block	
? Major components' interfaces	Hote that <u>remetrer block</u> description (help) nes can also be opened by doing a right model click on a block.	
? C++ Library Interface	Likewige	
Operating System, platform and compatibility	Libranes	
? Variations to ClickAndMove		
7 Timing considerations	AMCSerialComm	
System performance, Run-time	ArrayManipulation	
Principle of synchronized interpolated motion	BasicElements	
? ∀ariables	BitManipulation	
Punction Block definition	CAN	
Pasic Elements Library	CANapan Protocol	
? Config Elements Library		
? Attributes	OMOIS and the second se	
? Data types	CMSharedMemintr	
? User derived Function Blocks	CommunicationForDataHub	
? Function Block diagram	CommunicationForTcpip	
? Net connections	CommunicationForUdpip	
? Project structure	ConfigElements	
Menu structure	DataCollect	
The CandM MC Debugger	DriveSpecific	
The CandMcontroller program	ErrorHandling	
Distributed Applications	EthernetFieldhue	
Online Operation		
Virtual Machines	EUSBIO EUSPICE	
Collected Application	<u>FileHandling</u>	
Download application to remote device	<u>FpgaTools</u>	
HMI windows	<u> K220</u>	
Creating a ClickAndMove application	IOModule	
Why are State Machines Important?	Logic	
PLCOpen Overview	LogicGates	
Multi axis coordinated motion	Math	
The FBD Editor (Eagle)	Mmod Hilty	
Application Examples		
Function Block Library descriptions	MutorixaesoloupParkw	
Video tutorials	MotionGLoae	
Appendix	MotionInterpreter	

UDFB Overview

You can create a User Derived Function Block from a block of logic. The UDFB is added to the library and can be used in the same way as the C&M supplied function blocks. UDFB is convenient when the same logic is required in multiple locations and as an organizational tool to improve the readability of the project schematics.



Create UDFB

To create the UDFB start with a new empty schematic. On the C&M Desktop click 'File', 'New FBD (schematic)



Name UDFB



This schematic is named 'SUM_PROD'.

Choose 'File', 'Save As' in the top left corner of the Eagle window and give the new schematic a name. Standard convention is to use all upper case characters and under scores as spaces. The name you use will identify the UDFB in the library.





Add Elements to the UDFB

Add inputs, outputs, and logic elements to the UDFB from the library. You can even add other UDFB's to this schematic! For our example we need 3 inputs, one output, an addition block and a multiply block. Click the 'Edit', 'Add...' or use the add button the library.

You can find the inputs in the 'Basic Elements' section of the library.

Name 🖉	Description	<u> </u>
	AMCSerialComm Library Description	
🗄 - ArrayManipul	ArrayManipulation Library Description	
BasicElements	BasicElements Library Description	
GLOBAL	GlobalDefiniton Help	
INPUT	Input Help	
OUTPUT	Output Help	
PROXY_I	Proxy Input Help	
PROXY	Proxy Output Help	
SOURCE	Source Help	
BitManipulation	BitManipulation Library Description	
Ė CAN	CAN Library Description	
+ CANonenPro	CANonenProtocol Library Description	

Choosing an Element

Double clicking on the 'Input' selection under 'Basic Elements' will grab the input item. Alternately, you can single click 'Input' and then click the 'OK' button at the bottom of the window. Caution: Do not click the 'Drop' button, it will temporarily remove the selected item from the menu.



Once the element is grabbed, Eagle returns to the schematic. An element is dropped with each click of the mouse. Use the roller wheel on the mouse to zoom in and out of the schematic and press 'Escape' to end the insertion action.





Add all the Elements

Continue the add process to insert one output, one addition block and one multiply block to the schematic. You can find ADD and MUL in the MATH section of the library.



Add some text to the schematic to describe the function of the UDFB. Click 'Draw', 'Text' or use the 'Text' button **T** to open the text tool. Add the text: "Point Slope Calculator Y=MX+b". Press escape to close the tool.





Schematic description...

Assembly variants...

Name the Inputs and Outputs

Click 'Edit', 'Value' from the top left corner of the screen to enter the set value mode. Click the inputs and outputs one at a time to set the value (name) of each as shown.

Click the add wire button and connect the elements as shown. Click the end of a pin to start and end a wire.



Add UDFB to the Library

button

Use 'File', 'Save' to save your changes. Click the UDFB and choose 'Create' to add the UDFB to the library.

Close the Eagle edit window, the UDFB is ready for use.



😤 C&M_MC_Help	
Hide Back Forward Home Print Options	
Contents Igdex 201950 Totle Title Title Didux Didux	User derived Function Blocks An important concept is the User Derived FBs. With this concept, a user can generate his own libraries or identifies the reusable parts of an application program, created with the standard available FBs. By created save a tremendous amount of time in the next project. Moreover, the usage of own libraries enhance The process of creating a UDFB starts with creating the corresponding FBD. The new FB's name (e.g. Jaw will represent the input and output pins of the new FB. Once the MyFB sch is created the user needs to new FB symbol in the ProjectUDFB.lbr. (FBD Editor commands) From this point on it will be the <i>FBname help.html</i> help file with pin descriptions. The generated file will be placed into the projectname/Source/Docs then it will not generate that file. In this case projectname/Source/Docs/FBname.help.html will be listed by the Open HTML FB help file com The UDFB below was created from the JOG.sch FBD. UD1 JOG Avis_REF ERROR_ID ENABLE Forward
Collected Application	The internal schematic of a ODFB can also be opened by the right mouse click/ C&M Function Block Files

Additional UDFB information can be found in the C&M_MC_Help Overview section.

UDFB With CPP Overview

All function blocks have lines of code behind them. It is possible to create a UDFB containing your custom lines of CPP code.

To do this we will construct a UDFB containing only inputs and outputs. We build the project to have C&M create skeleton CPP files. We then edit the skeleton files adding our code.

The C&M_MC_HELP on this subject can be found in the Overview section under 'C++ Library Interfaces.



3 of 5 Averaging Problem

One of our customers needed a position averaging function. He wanted to average a set of 5 data points and not include the maximum and minimum samples in the average.

I came up with a solution based on the logic in this diagram. The function works but this might be better implemented in CPP.

Lets create a UDFB from CPP code to solve the same problem.



Repurpose Existing Project

Open the Integrator project from the examples folder and save it to the project folder as AVERAGE_DEMO.

Project name	
Project thumb view ON 🔽	
Integrator	•
Project parent directory	Details >>
C:\CandM\Working_5_2_6\Examples	
Project directory:	
C:\CandM\Working_5_2_6\Examples\Integrator	
	. 1
OK Cancel He	elo ale

Project name					
AVERAGE_DEMO					
Project parent directory					
C:\CandM\Working_5_2_6\Projects					
Project directory:					
C:\CandM\Working_5_2_6\F	Projects\AVERAGE_DEMO				
1	1	1			

Add an Skeleton UDFB

Add a User Derived Function Block to hold the CPP code. Add inputs and outputs shown in the image below. Use the exact names shown, they will become the names of the arguments and returned values for the CPP program. Update the library and save and close the UDFB. Rebuild the project to generate the skeletop files.



Add a Cpp folder to the project.



Create a folder named Cpp under the project\source folder. The new folder is shown in the image on the left. We must move the two files named CppAverageImpl.cpp and CppAverageImpl.h (Shown Below)from the project\Generated\Source\Cpp folder to the folder we just created. Be sure to move them, they must no longer exist in the Cpp folder under the Generated folder.





Add Code to the Header File

```
10
   eclass CppAverage : public CppAverageBase {
11
12
        public:
13
           CppAverage(Char const * instanceName, const
14
           ~CppAverage();
15
          void body(void);
16
        protected:
17
        private:
18
           // Object Copy Prevention
19
           CppAverage(const CppAverage&);
20
           CppAverage& operator=(const CppAverage&);
21
22
23
        // The next lines are my variables.
24
           Bool sampledEnable ;
25
           Bool sampledTake ;
26
           Float64 avgSum ;
27
           Float64 minVal ;
28
           Float64 maxVal ;
29
           Int sampleNum ;
30
31
    };
32
```

Add the hilighted line shown on the left to the CppAverageImpl.h file. Save the changes and close the file.

```
23
24
     void ImplFBsCM::CppAverage::
25 ⊟body()
26
     {
27
       Float64 tmp;
28
29
       if(*pin_Enable)
                                                 // Is the enable pin set to true?
30
       {
31
         if(!sampledEnable_)
                                                  // Are we detecting the rising edge of the enable pin?
32
                                                  // Take the first sample.
33
           sampleNum =1;
34
           maxVal_=(*pin_Input);
35
           minVal =maxVal ;
36
           avgSum =maxVal ;
37
           sampledTake_=(*pin_TakeSample);
                                                 // Note the state of the take sample pin
38
         }
39
         else
40
         {
41
           if(sampleNum <(*pin SampleSize))</pre>
                                                 // We keep taking samples until all requested samples are done.
42
43
             if(sampledTake_!=(*pin_TakeSample)) // Has the Take Sample pin changed state?
44
45
                                                 // Read the Input pin value
               tmp = (*pin_Input);
46
               avgSum_+=tmp;
                                                 // Add the new sample inut to my accumulator
47
               if(tmp>maxVal_)
                                                 // Copy the sample to my MAX and MIN registers if is an extreme value
48
               £
49
                 maxVal = tmp;
50
51
               if(tmp < minVal_)</pre>
52
53
                 minVal_ = tmp;
54
55
               sampleNum ++;
                                                 // Bump up my sample taken count
56
57
           sampledTake_=(*pin_TakeSample);
                                                 // Update the last known value of the take sample input pin
58
           3
59
                                                 // If we have taken all samples, calculate the average and signal Done.
           else
60
61
             src AverageValue=(avgSum - (maxVal + minVal ))/((*pin SampleSize)-2);
62
             src Done=TRUE;
63
64
         }
65
       }
66
       else
                                                 // If the enable pin is turned off, then turn off output Done pin.
67
       {
68
         src Done=FALSE;
69
       }
70
       sampledEnable_=(*pin_Enable);
                                                 // Update the last know value of the Enable input pin
71
    }
72
```

Add the lines of code shown on the left to the body function in the CppAverageImpl.cpp file. Save and close the file.

File P	roject Run Interface Virtual machine Collected	C&M application Tools Settings					Wind	
	Build (compile only latest changes) Rebuild (complete recompile)		U	虊	?	•	Targ	
	Delete C&M generated files							
	Open XML Delete generated files Validate all Delete all the files generated by the C&U This command is useful to reduce the s Open HTM to be e-mailed.	M Compiler size of the pr	oject	e.g. i	n <mark>cas</mark>	e it need	s	
	Open HTML FB help file							
	Replace FBD editor (Eagle) settings with defaults							

Rebuild the Project

Clean the project and then do a complete rebuild of the project.

File Project Run Interface Virtu	al machine Collected C&M application Tools Settings Window Help
🕒 🄊 送 - 🖿 🗀 🛤 🔺 📣	🏆 🔤 🕨 🏂 🍓 🕨 💓 🔯 🤹 🥐 🔹 Target platform : PC
Message window	Rebuild (complete recompile) Image: Second state of the seco

Add an Empty UDFB to the Project

To create the UDFB start with a new empty schematic. On the C&M Desktop click 'File', 'New FBD (schematic)



Add DROP_AVERAGE UDFB To the Project



Add the elements shown on the left to the empty schematic just created. Save the schematic as DROP_AVERAGE.

Delete the Integrator schematic



Open the Main schematic and delete all the logic blocks, wires, inputs and outputs.



Disconnect the Graphical HMI

Right click the INTERFACE object and choose Set Connect . Clear the field named "TYPE =" and click "OK"





Add Averaging function

Add the inputs, outputs, the DROP_AVERAGE UDFB, and the wires as shown in the image at left.

Set the default values for SAMPLE_RATE and SAMPLE_SIZE as shown.

Close the schematic and rebuild the project.

File Project Run Interface Virtual machine Collected C&M application Tools Settings Window Help Build (complete recompile) Delete C&M generated files Virtual Machine Point Point Point C:/F Open XML property file editor Point <	Click&Move Deskto	p- C:\CandM\Working_5_2_	6\Projects\AVERA	GE_DEMO		100		x		e
Build (complete recompile) Delete C&M generated files C: /P Open XML property file editor make Make Addide all property and config XML files Open XML project description file C: /P Open XML FB help file C: /P Open XML FB help file C: /P Open XML FB help file C: /P Options C' /AMEXIN-1/ Common/System/Tacs/PCW -1 Puburger /Cop/MXN / Options Beauger /Cop/MXN / Options Beauger /Cop/MXN / Options Beauger /Cop/MXN / Options C: /AmeM /MORKIN-1/ Common/System/Tacs/PMN -1 C: /CandM /MORKIN-1/ Common/System/Debugger / Ince /MN -1 C: /CandM /MORKIN-1/Common/System/Debugger / Ince /MN -1 <tr< td=""><td>File Project Run</td><th>Interface Virtual machine</th><td>Collected C&M a</td><td>application</td><td>Tools</td><td>Settings</td><td>Window Hel</td><td>р</td><td></td><td>2</td></tr<>	File Project Run	Interface Virtual machine	Collected C&M a	application	Tools	Settings	Window Hel	р		2
CC:/CandAr/WORKIN~L/Projects/AVERGeV - 1 C/CA129E~L/System/MakeFiles/SmemIntfCor Source/Cpp/MGW/PCW -I Source/Cpp/PCW -I C/CA129E~L/System/MakeFI-1 make.exe[1]: Entering directory `C:/Can make.exe[1]: Leaving directory `C:/Can C:/PROGRA~2/A-M-C/CA129E~L/System/Bin/A CC:/CandM/WORKIN~L/Projects/VERAG~1 - f C/CA129E~L/System/MakeFiles/XMLValidati make.exe[1]: Entering directory `C:/Can make.exe[1]: Entering directory `C:/Can make.exe[1]: Nothing to be done for `Bty make.exe[1]: Nothing to be done for `Bty make.exe[1]: Leaving directory `C:/Can make.exe[1]: Leaving directory `C:/ROGRA~2/A- Project is successfully built.	File Project Run Build (com Rebuild (com C:/P: C:/P: C:/P: Open XML make Validate all make Validate all make Open HTM C:/P: Open HTM C/CA C:/P: Open HTM C/CA C:/P: Open HTM C/CA C:/P: Open HTM C/CA C:/P: Open HTM C/CA C:/P: Open HTM C/CA C:/P: Open HTM C/CA C:/P: Options C:/PROGRA-2/A-M-C/ Debugger/Opp/MGW Generated/Debugger C:/CandM/WORKIN-1/ C:/CandM/WORKIN-1/ C:/CandM/WORKIN-1/ C:/CA129E~1/System/ make.exe[1]: Enter make.exe[1]: Leavi C:/PROGRA-2/A-M-C/ C:/PROGRA-2/A-M-	Interface Virtual machine pile only latest changes) mplete recompile) A generated files property file editor property and config XML fil L project description file L FB help file D editor (Eagle) settings with Click this menu item, select The topics in this chapter d Common/System/Debugge Common/System/Debugge Common/System/Debugge Debugger/Incs/NGW/PCW Debugger/Incs/NGW/PCW Incs/NSystem/Debugge Debugger/Incs/NGW/PCW Incs/NSystem/Debugge Debugger/Incs/System/Debugge Debugger/Incs/NGW/PCW MAKEFI~1 ing directory `C:/Cand CA1292~1/System/Bin/P	Collected C&M a es defaults a tab and press FI escribe various pre r/Incs -I C:/f -I C:/PROGRA- C:/PROGRA-2A- Project options - Tai Target platform De	Autosave	Tools To	Settings CM_TARGE /System// CM_TARGE I C:/PRO cs/PCW - on/Syste on/Syste /Debugg stem/Deb M_ ptions Targe Compro	Window Hel Target platform m/Incs -I MAKEFI~1 T_ID=PCW GRA~2/A-M-I m/Incs/MGW - m/Incs -I pp/PCW -I er/Cpp -I ugger/Incs/M tproperties Load pa ment selection for "Reference of the selection for the selection for "Reference of the selection	P : PCW X -I fGW ath un all' menu	item	
	C/CA129E~1/System/ Source/Cpp/MGW/PCW C/CA129E~1/System/ make.exe[1]: Enter make.exe[1]: Nothi make.exe[1]: Leavi C:/PROGRA~2/A-M-C/ CC:/CandM/WORKIN~1 C/CA129E~1/System/ make.exe[1]: Enter make.exe[1]: Inothi make.exe[1]: Leavi make: Leaving dire Project is success	MakeFiles/SmemIntfCog -I Source/Cpp/PCW -I MAKEFI~1 ing directory `C:/Car ng to be done for `Bu ng directory `C:/Can CA1292~1/System/Bin/A /Projects/AVERA6~1 -f MakeFiles/XMLValidati ing directory `C:/Can otory `C:/Can ctory `C:/PROGRA~2/A- fully built.		Debug Debug Projec Messa Build optic Auto With Auto	gger Jgger Viewer at viewer age window ns -generate pa -generate pa -generate pa -generate pack -upload pack	Ackage from pr MI C With C With	&M Application coad and run C Loa lebugger oject after compilation nout C&M-Min-HMI after compilation			

Set the project Options

Click File and Options from the C&M Desktop.

Un-check the C&M Hmi and check the C&M Min-Hmi.

Click Apply and OK





At startup the Min-Hmi will open with controls and indicators for each input and output on the main schematic.

The program samples a stream of input data at the sample rate. After collecting the number of samples indicated by sample size the average is calculated.

The calculated average will not include the maximum and minimum valued samples taken. For that reason the actual number of samples in the average is always two less than the value of sample size.

To test, use a 10 second sample rate and the debugger to monitor the action and set the input value manually from the mini HMI. The first sample is taken when the enable input goes true and subsequent samples are taken each time the take sample input changes state.