Available Memory

The amount of free memory in the controller decreases when the controller has started up, and an empty project has been downloaded from Control Builder M.

The remaining memory is what can be used for application code, and is hereafter referred as to "Available memory".

The measurement results in Table 44 are made without IAC and any configured communication protocols and CEX units. Memory consumptions for used protocols and CEX units have to be added, according to Table 45.

Table 44. Available RAM Memory and Performance in AC 800M Controller (without Protocol Handlers)

Controller	Execution Performance Factor	Total RAM (kbytes)	Firmware and an Empty Project (kbytes)	Available Memory (kbytes)
PM851	0.5	8192	5570	2622
PM856	0.5	8192	5570	2622
PM860	1.0	8192	5570	2622
PM861	1.0	16384	8520	7864
PM861A	1.0	16384	8520	7864
PM864	1.5	32768	8531	24237
PM864A	1.5	32768	8531	24237
PM865	1.5	32768	8526	24242
PM865 SM810	0.9	32768	8918	23850
PM865 SM811	0.9	32768	8964	23804
PM866	2.1	65536	13864	51672
PM891	4.5	262144	61029	199650

Execution Performance

Cyclic CPU load is calculated as a percentage using the following formula.

Cyclic CPU load (%) = 100(Total execution time / Total interval time)*

Depending on the amount of code and requested task interval times, applications may demand up to 70% of CPU capacity (never more)[1]; the execution of IEC 61131-3 code is called *Cyclic Load*. Should an application require more than 70% of CPU capacity, the task scheduler automatically increases the task interval times to re-establish a 70% load.

Load balancing can be disabled (see the manual *AC 800M Configuration (3BSE035980*)*. In a HI Controller, load balancing is always replaced by overrun and latency supervision.

It is important to consider CPU load if communication handling is vital to the application. Running at the maximum cyclic load will result in poor capacity and response times for peer-to-peer and OPC Server communication.

Communication handling has the lowest priority in a controller. It is therefore important to consider controller CPU load if the communication handling is vital to the application. Running close to 100% total load will result in poor capacity and response times for peer-to-peer and (OPC Server for AC 800M) communication. It is recommended that peak total load will be kept below 100%.

Among the communication protocols, the IAC MMS protocol will be the last to be affected if there is a communication data starvation.

CPU load is also influenced by other factors, such as Modulebus scan interval and the number of modules on Modulebus (AC 800M), or the scanning of ABB Drives.

The PM860 and PM861/PM861A processor units have the same internal design and the same performance when execution application program.

The PM851, PM856 and PM860 processor units have the same internal design. They differ only in performance when executing an application program. The execution time in PM851 and PM856 is approximately two times the execution time in PM860.

1. This is **not** true if load balancing is set to false or if you run with an AC 800M HI. The controller will run until it is forced to stop.

More than One Application in the Controller

Less spare memory is needed when there is more than one application in the controller.

The on-line changes are done to one application at the time. This means that if changes are done to more than one application in the controller, these changes will not take effect in a synchronized way.

Example: One application requires 50% used memory and 70% maximum used memory. If you split this application into two equally smaller applications, it will still require 50% used memory, but only 60% maximum used memory, since the extra memory needed for the on-line changes will be half.

Comparing Memory Allocations Made with Different Versions

From the discussions above, you can see that the "used memory" value provided by the *SystemDiagnostics* function block cannot be used to compare different versions.

The amount of available memory in the controller varies between versions for a number of reasons, one being the number of functions implemented in the firmware.

Memory Consumption and Execution Times

Memory is reserved for each function block type defined. When another instance is created, the amount of memory reserved for the instance is very small in relation to the type. This means that the memory consumed by the type itself is of great importance.

The following tables show memory consumption and execution time for AC 800M PM864/PM866/PM891 controllers, for a number of common function blocks and control modules.

In the tables the *First Object* column shows the required memory for the object type and one function block or control module and *Next Object* column shows the required memory for every further function block or control module.

Table 48 and Table 49 list the execution times and first scan execution time for a number of common SIL classified Function Blocks and Control modules.

Table 46. AC 800M Memory Consumption and Execution Time for Function Blocks and Control Modules

Object	First Object (kbytes)	Next Object (kbytes)	PM864 (µs)	PM866 (µs)	PM891 (µs)
Function Blocks					
ACStdDrive	82.5	18.0	584	404	228
AlarmCond	6.9	1.9	34	26	15
AlarmCondBasic	5.0	1.4	21	16	9
Bi	60.0	13.1	316	279	133
McuExtended	110.8	29.8	566	360	240
MotorBi	71.8	16.2	386	284	164
MotorUni	60.8	11.9	311	223	133
PidCascadeLoop	67.7	12.7	490	330	147
PidCascadeLoop3P	71.6	12.9	481	351	141
PidLoop	55.6	6.2	269	189	88
PidLoop3P	59.6	6.3	298	214	96
PidSimpleReal	9.0	1.8	63	51	16
SDBool	22.9	5.7	115	97	48
SDInBool	23.8	5.8	85	66	35
SDInReal	42.9	14.4	262	187	92
SDLevel	23.3	5.7	77	55	35
SDOutBool	26.4	7.6	114	83	52
SDReal	38.8	13.2	224	159	82
SDValve	28.5	6.6	159	138	60
SignalBasicBool	4.3	0.7	8	6	3
SignalBasicInBool	4.2	0.8	9	6	4
SignalBasicInReal	10.0	1.5	62	43	18
SignalBasicOutBool	4.2	0.8	8	6	4

Table 46. AC 800M Memory Consumption and Execution Time for Function Blocks and Control Modules (Continued)

Object	First Object (kbytes)	Next Object (kbytes)	PM864 (µs)	PM866 (µs)	PM891 (µs)
SignalBasicOutReal	6.1	1.1	15	11	5
SignalBasicReal	3.9	0.9	17	12	5
SignalInBool	23.3	4.4	64	47	28
SignalInReal	59.2	12.1	166	121	65
SignalOutBool	22.7	4.2	43	32	18
SignalOutReal	53.1	10.9	107	76	39
SignalSimpleInReal	22.4	3.7	64	47	24
SignalSimpleOutReal	16.4	3.3	29	24	14
StatusRead	10.8	3.5	37	27	13
Uni	52.4	9.5	196	155	82
ValveUni	50.9	8.8	188	134	83

Table 46. AC 800M Memory Consumption and Execution Time for Function Blocks and Control Modules (Continued)

Object	First Object (kbytes)	Next Object (kbytes)	PM864 (µs)	PM866 (µs)	PM891 (µs)
Control Modules					
ACStdDriveM	87.6	18.4	582	451	229
AlarmCondBasicM	5.9	1.2	36	26	17
AlarmCondM	5.5	1.4	24	17	11
AnalogInCC	21.3	4.0	119	89	47
AnalogOutCC	19.1	4.1	91	65	34
BiM	64.9	13.9	419	253	149
CascadeLoop	214.6	64.3	-	-	-
Detector2Real	81.1	14.6	373	290	155
DetectorBool	35.5	6.7	159	114	69
FeedForwardLoop	208.1	50.7	-	-	-
Level2CC	21.7	5.5	125	75	46
Level4CC	29.8	7.7	153	109	72
Level6CC	38.1	10.0	206	153	94
McuExtendedM	114.5	30.4	509	420	239
MidRangeLoop	208.0	51.2	-	-	-
MotorBiM	73.3	16.0	400	289	169
MotorUniM	63.8	12.0	309	227	136
OverrideLoop	283.3	118.8	-	-	-
PidAdvancedCC	206.4	26.7	833	602	280
PidCC	94.5	15.7	483	349	168
PidSimpleCC	12.4	2.7	102	71	33
SignalInBoolM	26.1	4.6	59	47	25
SignalInRealM	67.2	12.0	248	201	103

Table 46. AC 800M Memory Consumption and Execution Time for Function Blocks and Control Modules (Continued)

Object	First Object (kbytes)	Next Object (kbytes)	PM864 (µs)	PM866 (µs)	PM891 (µs)
SignalOutBoolM	26.0	5.0	76	55	33
SignalOutRealM	61.9	11.8	231	155	88
SingleLoop	184.8	37.7	-	-	-
ThreePosCC	20.6	4.5	153	110	51
UniM	55.1	10.2	208	152	90
ValveUniM	53.6	9.6	251	158	104

Table 47. Execution Time for a Number of Standard Operations and Function Calls

Operation/Function	Data Type	PM864 (ns)	PM866 (ns)	PM891 (ns)
a:= b or c	bool	156	105	9
a:= b and c	bool	155	106	9
a:= b xor c	bool	150	142	9
a := b	string	7237	4671	507
a := b + c	string	4041	2921	328
a := b + c	string[10]	2455	1790	204
a := b + c	string[140]	9924	7243	773
a := b + c	dint	148	109	10
a := b + c	real	1210	869	12
a := b - c	dint	148	107	10
a := b - c	real	1308	927	13
a := b * c	dint	152	108	9

Table 47. Execution Time for a Number of Standard Operations and Function Calls (Continued)

Operation/Function	Data Type	PM864 (ns)	PM866 (ns)	PM891 (ns)
a := b * c	real	1199	854	11
a := b / c	dint	319	236	65
a := b / c	real	3426	2481	58
a:= b <> c	dint	172	128	13
a:= b <> c	real	1016	735	18
a := real_to_dint(b)	dint	7203	5161	116
a := dint_to_real(b)	real	1252	902	63
a := real_to_time(b)	time	18355	12977	953
a := time_to_real(b)	real	5493	3915	210

Table 48. AC 800M Execution Times in PM865 HI

Object	First scan exec time (µs)	Continuous exec time (µs)
Function Blocks		
AlarmCond	1075	63
AlarmCondBasic	827	36
Bi	1534	520
BiSimple	287	253
ForcedSignals	299	190
MotorBi	1708	725
MotorUni	1444	511
SDBool	1189	157
SDInBool	1168	120

Table 48. AC 800M Execution Times in PM865 HI (Continued)

Object	First scan exec time (µs)	Continuous exec time (µs)
SDInReal	1410	346
SDLevel	662	109
SDOutBool	1735	161
SDReal	1349	285
SDValve	1281	227
SignalAE	561	36
SignalBasicBool	14	13
SignalBasicInBool	25	14
SignalBasicInReal	82	78
SignalBasicOutBool	24	14
SignalBasicReal	25	22
SignalBool	112	78
SignalInBool	1595	87
SignalInReal	299	271
SignalOutBool	109	71
SignalReal	241	226
SimpleEventDetector	166	57
StatusRead	43	42
Uni	1004	345
UniSimple	173	158
ValveUni	982	341

Table 49. AC 800M Execution Times in PM865 HI

Object	First scan exec time (µs)	Continuous exec time (µs)
Control Modules		
AlarmCondBasicM	910	59
AlarmCondM	911	35
BiM	1591	592
BiSimpleM	275	265
CCInputGate	56	50
CCOutputGate	57	54
CO2	946	309
Deluge	855	245
Detector1Real	1436	400
Detector2Real	1570	490
DetectorAnd	74	73
DetectorAnd4	81	80
DetectorAnd8	122	116
DetectorBool	1361	220
DetectorBranch	78	74
DetectorBranch4	96	85
DetectorBranch8	125	124
DetectorLoopMonitored	1505	455
DetectorOr	81	73
DetectorOr4	78	79
DetectorOr8	114	112
DetectorRemote	1718	221
DetectorVote	77	72
ErrorHandlerM	33	22

Table 49. AC 800M Execution Times in PM865 HI (Continued)

Object	First scan exec time (µs)	Continuous exec time (µs)
Control Modules		
FGOutputOrder	913	299
ForcedSignalsM	317	207
GroupStartObjectConn	138	91
MotorBiM	1682	685
MotorUniM	1558	503
OrderMMSDef16	581	715
OrderMMSRead16	1147	1159
OrderOr	56	56
OrderOr4	51	50
OutputBool	232	157
OutputOrder	895	290
OverrideControlInterface	46	45
OverviewConversion	263	266
SDLevelAnd4	39	38
SDLevelBranch4	46	30
SDLevelM	1209	164
SDLevelOr4	38	33
SignalBoolCalcIntM	1224	129
SignalBoolCalcOutM	1270	124
SignalInBoolM	1749	103
SignalInRealM	4239	384
SignalOutBoolM	1659	106
SignalRealCalcInM	3874	456

Table 49. AC 800M Execution Times in PM865 HI (Continued)

Object	First scan exec time (µs)	Continuous exec time (µs)
Control Modules		
SignalSimpleInReal	107	104
SiteOverview	206	198
SiteOverviewMMSDef	1273	1283
SiteOverviewMMSRead	1138	1181
SiteOverviewOr4	74	74
Sprinkler	808	195
UniM	1415	410
UniSimpleM	184	168
ValveUniM	1406	398
Vote1oo1Q	1105	559
VoteBranch4	58	50
VotedAnd4	42	38
VotedBranch4	41	32
VoteXoo2D	1269	662
VoteXoo3Q	1297	635
VoteXoo8	1329	651

Online Upgrade

Stoptime

The total stop time of a controller when doing online upgrade is defined as the longest time the I/O is frozen. Generally, the following factors will have large impact on the total stop time.

- The stop time will increase with increased memory usage in the controller.