The following is a list of commands for ASC3, a procedural programming language used in the games Starcraft and Starcraft: Brood War for AI scripting.
Credits are located at the bottom of this file.

Notice: Whenever you see the word "call" in the text below, I probably meant to say "the use of [some] command", and not the "call (block)" command specifically. Comments regarding the call command will be highlighted with the colour red.

AI Flags and header commands

farms_notiming
Sets a flag that makes the computer player build necessary farms (Overlords, Supply Depots, Pylons - only of the AI Player's respective race) only when it hits exactly the maximum supply available. [1]

farms_timing
Sets a flag that makes the computer player build necessary farms with a correct timing, so that the AI never gets supply capped.

start_areatown
Starts the AI Script for area town management. Uses the trigger location (NOT regions) as its bounds for assigning buildings to the town, but the state % regions themselves are created in relation to building placement, not the location. Might cause issues with town management if used mid-game, mostly intended for preplaced bases, ran at the time of town init.

start_campaign
Sets the campaign flag which affects town management significantly. Redundant in UMS, since the UMS game setting forces it onto all players.
Generic start_campaign header structure for towns:
start_campaign
wait 1

start_town
defaultbuild_off
default_min (byte)
wait 1

For areatowns:
start_campaign
start_areatown
defaultbuild_off
default_min 0
wait 1
Waits in both headers should NOT be removed, start_campaign can.[2]
start_town

[1] yes this means that the AI can get stuck if it's waiting to train a 2 supply unit but is missing 1 supply to get capped
[2] removing the waits desyncs building assignment

Starts the AI Script for town management.[3] All of the computer player's buildings that are not inside of an areatown get assigned to the last called start_town.

start_town is also needed for every block that an expansion opcode uses, but notably there are no race checks for assigning a worker that starts the new town

transports_off
Computer will not train transports and observers on its own.[4]

check_transports
Makes the AI player train transports on its own beyond their base values. Trains transports or ovelords up to 5. The first transport is built at priority 80 and subsequent transports are built at priority 50. Protoss also trains up to 5 observers at priority 80. The numbers might change depending on the amount of preplaced transports and whether the AI player has the campaign flag set or not.

Build-Tech-Upgrade/Attack/Defense/Train order commands

default_build
If the AI has more than 600 minerals and 300 gas and no other requests, it will continuously train race specific units until it reaches the define_max value or gets supply capped.
If it has other requests, it instead requires at least 1500 minerals and gas to train default_build units.
Terran: marine, ghost, siege tank, goliath, wraith, battlecruiser.
Zerg: hydralisk, mutalisk.
Protoss: zealot, dragoon, reaver, scout, carrier.

Note: "default build" is on by default. To turn it off, use defaultbuild_off

In the vanilla game, using train/wait_force/do_morph for (unittype) while the AI is requesting a guard replacement of (unittype) will instead call default_build that lasts until the train/wait_force request is filled.

defaultbuild_off
Turns off default_build. default_build should realistically always be turned off.

attack_add (byte) (military)
Add %1(byte) %2(military) to the current attacking party. %1 (byte) can be at most 64, the units beyond the 64th one are ignored and not added to the attack.
Multirunning multiple attack forces in a way that keeps units as already attacking can be used to bypass the 64 size limit.
In the vanilla game, Zerg AIs ignore units inside cocoons or eggs, so any training command including attack_add with attack_prepare will cause zerg AIs to overtrain certain units, and in the case of hydra->lurker and muta->guard/devo transitions, use them in the attack.

---

[3] buildings need to be assigned to a town to be used at all by the AI
[4] It will still use existing ones

Trains units at priority 50 if attack_prepare was used, requires separate train uses for special attacks like target_expansion or eval_harass.

Unit IDs of Defiler, Queen, Science Vessel, High Templar will arrive in the grouping region for the attack, but don't follow the attack force after grouping is finished.[5]

train (byte) (military)
Train %2(military) until at least %1(byte) of them are being trained[6], at priority 50 - doesn't execute if the AI has other requests while simultaneously having less than 1500 minerals and gas. The request is checked every 30 frames, and the command waits for 16 frames every time the requirement is not filled.

do_morph (byte) (military)
Attempt to train enough %2(military) units so that the AI commands %1(byte) of them. Doesn't re-apply the train request and doesn't wait for anything.

wait_train (byte) (military)
Wait until computer commands %1(byte) %2(military).[7]

wait_force (byte) (unit)
Wait until computer commands %1(byte) %2(unit), while training up to %1(byte) %2(unit) as long as it's waiting. The submitted train request is re-checked every 30 frames, and the command waits 16 frames every time the requirement is not filled.

attack_clear
Clear the attack data - reset the existing state 8 regions to 0, and clear the attack list. Some types of attacks can't start unless the attack data is cleared first.

attack_do
Prepare the attack and attack the enemy with the current attacking party afterwards.[8] Waits for grouping to finish and for the attack to launch before resuming script execution.

If a special attack type or attack_prepare already has been used, this command only waits. Should not be used with attack types that don't perform grouping.

attack_prepare
Prepare the attack and attack the enemy with the current attacking party afterwards.
[9]
Generic attack structure
attack_add(...)
attack_do
attack_clear

set_attacks (byte)

---

[5] other casters and hero casters follow properly, though all units without a weapon will retreat after getting attacked
[6] please note that this waits and stops script execution until the condition is fulfilled
[7] does not actually request training, just waits
[8] Captain unit note: SC1 AI has no captain units, instead region centers are used for rallying
[9] Identical to attack_do but doesn't stop code execution

Sets the number of attacks possible to execute with target_expansion. Cannot be used if the campaign flag is set.

With enough set_attacks left, target_expansion can re-prepare the same force and execute the attack again.

target_expansion
Executes an expansion attack with a deadline of 1 second (And thus requires the units to be trained in beforehand and practically has no grouping)
Expansion attacks never choose bases built around start locations before 1500 in-game seconds. They can acquire those bases as targets afterwards, but they will still prefer to target expansions.

Attack note: You should never call attack_prepare or target_expansion before 16 frames have passed or else the attacks will have a broken timeout timer and will possibly never fire. Preparing the attack has a hard-cap on it's length of 120 (melee) or 180 (campaign) seconds - after this time, the attack will commence even if the AI didn't manage to group all the units.

The default target is picked semi-randomly for campaign scripts, or is the closest possible target for melee scripts. Grouping occurs within 5 pathfinding regions of the target for campaign scripts, for melee scripts there's an additional check to see whether the area's safe before picking a grouping region. Some commands check force values in the region where the target is located at the time of their usage.

Attacking uses 3 different states. More on region states under **"Region states"**

quick_attack
Sets the starting attack point back far enough in time so that the deadline goes down to 5 seconds. Never use this if it would set the starting point to before the first 16 frames of the game.[10]

eval_harass (block)
Jumps based on whether the Player is currently attacking or not

harass_factor (word)
Used after attack_prepare, or otherwise during the grouping stage. Takes the total force of enemy units in the targeted AI region, and then compares it to (word) - if the result is big enough, it duplicates or triplicates the attack force. The formula is as follows:
((word+enemystrength-1)/(word))-1; division is rounded down, multiplication is in range of 1 (no change) to 3.

wait_finishattack
Waits until there are no state 1/2/8/9 regions existing for the owner of the thread.

build (byte) (building) (byte)
Build %2(building) until it commands %1(byte) of them, at priority %3(byte). The maximum value of %1(byte) is 30 per town. Max worker values can be lower and are dynamic, based on the number of mineral patches etc

[10] so, for example, don't use it with normal attacks in campaigns before 2976 frames have passed, never use it in conjunction with eval_harass or target_expansion

player_need (byte) (building)
Adds a request to the town that makes it so that if the player does not own %1(byte) number of %2(building) anywhere, then he rebuilds them in this town at priority 80.

wait_build (byte) (building)
Wait until computer commands %1(byte) %2(building).

wait_buildstart (byte) (building)
Wait until there are %1(byte) %2(building) that are either finished or their construction has started.

build_bunkers
Builds up to 3 bunkers around the base (Terran only). Seems to build them in regions where there are no static defenses.

build_turrets
Builds up to 6 missile turrets around the base (Terran only). Seems to build them in regions where there are no static defenses.

Terran AIs will always place siege_tank and ghost guards around newly built bunkers, and a ghost guard around newly built turrets.

wait_bunkers
Waits for the command build_bunkers to finish. Takes no parameters.

wait_secure
Waits until an AI spend cycle happens without it needing to train defenses. (Seems to wait until the AI has more Friendly Force of units than Enemies do near its Town regions.) Seems to refresh every about 7 seconds early into the game, or once every 37 seconds later in the game (~1500 frames), which may make the check delayed vs what's actually happening.

wait_turrets
Waits for the command build_turrets to finish. Takes no parameters.

Defense note: The maximum byte value of all defense calls of a given type (for example _aa) can reach 20 maximum. Calls beyond this will be ignored.

defensebuild_aa[11] (byte) (military)
Train %1(byte) %2(military) to defend against enemy air force or enemy air + ground force in unpathable regions.

defensebuild_ag (byte) (military)
Train %1(byte) %2(military) to defend against enemy air force in pathable regions.

---

[11] defensebuild order is as follows:
1. Check regions that need only air strength and train from _aa/_ag until force requirements are met
2. Check regions that need ground strength but have a nonzero air strength and train from _aa/_ag until force requirements are filled, however air force values of trained units are used to calculate vs ground force region requirements
3. Check regions that need ground strength and have no air force and train from _gg/_ga until force requirements are met

defensebuild_ga (byte) (military)
Train %1(byte) %2(military) to defend against enemy ground force in unpathable regions that have no enemy air units in them.

defensebuild_gg (byte) (military)
Train %1(byte) %2(military) to defend against enemy ground force in pathable regions.

defenseclear_aa
Clear all calls for _aa defense.

defenseclear_ag
Clear all calls for _ag defense.

defenseclear_ga
Clear all calls for _ga defense.

defenseclear_gg
Clear all calls for _gg defense.

defenseuse_aa (byte) (military)
Use %1(byte) %2(military) to defend against enemy air force or enemy air + ground force in unpathable regions.

defenseuse_ag (byte) (military)
Use %1(byte) %2(military) to defend against enemy air force in pathable regions.

defenseuse_ga (byte) (military)
Use %1(byte) %2(military) to defend against enemy ground force in unpathable regions that have no enemy air units in them.

defenseuse_gg (byte) (military)
Use %1(byte) %2(military) to defend against enemy ground force in pathable regions.

get_oldpeons (byte)
This command takes (byte) number of workers from the main to move to the expansion. Should be used after the expansion town center has finished building.

max_force (word)
Lets the AI use up to %1(word) worth of units to defend any given defensible region[12][13]. Unit value is based off the following formula:
Force = (sqrt((range / cooldown) * factor * damage + (((factor * damage * 2048) / cooldown) * (hp + shields)) / 256) * 7.58) / unit_specific_reduction * script_class_mod[14] | + ½*energy for casters (updated on spawn and whenever they're marked as active) (all weaponless casters have a force value of 1 by default, making their force value 26 when spawned without energy upgrades) - this second value is never used for casters in production, where it's 0 instead
Unit specific reductions:

---

[12] on melee, only state 3 and 4 regions will respect max_force values assigned for mirrored defense reqs
[13] on melee, the default max_force for ground units is 1500 and for air units is 1000, and there's a limited possibility of making max_force changes
[14] Script class modifiers: 1 for campaign 1,5 for melee

- Workers get 25% of default
- Interceptors, scarabs, mines 0%
- Firebats, mutalisks, zealots 200%
- Defilers, inf. terrans 1/16
- Reavers 10%

Factors are weapon factors from weapons.dat
In most commands I use the terms "force" and "strength" interchangeably.

guard_all
Sets all regions with self-owned units in them to state 5, identical on melee.

help_iftrouble
For every active computer player allied to this one, sets their regions to state 4 in every region the current AI player has a non-zero state.[15]

default_min (byte)
Sets the default minimal force value for town regions. Town halls and workers will have this value multiplied by 6.

default_min is 0 when not specified otherwise.[16]

capt_expand
Creates state 4 regions around the AI's state 5 regions and around other state 4 regions;

If default_min is greater than 0, causes the AI to train first of it's defensebuild_gg and _aa units until it hits a supply cap, define_max limit or has no resources left. The AI then tries to distribute all these units over to state 4 regions, as evenly as possible. The production of these units has a priority based on the regions they'd be sent to, which usually results in a priority of 10.

guard_resources (military)
Send units of type %1(military) to guard as many unacquired base locations as possible(1 per spot).

place_guard (unit) (byte)
Place one %1(unit) to guard town at strategic location %2, at priority 60.

Preplaced units in campaign scripts also become guards.
Guards don't get removed unless they die three times while still being a guard.
Guards are bound to a specific spot on the map: Transports, Carriers and Reavers will never travel to that point though, and morphing units tend to be displaced from the spot they should be sitting in after they morph (this includes both archons)
A guard unit can be assigned to defending or attacking - then it loses it's guard status and the AI will try retraining a guard for that spot every (unit training time + 5) in-game seconds
Morphing guards - behaviours:
High and Dark Templars can morph into Archons while still being considered guards.

---

[15] There are relatively few cases where campaign AIs end up defending state 4 regions, and while some of them involve default_min, it can also be consistently done if you concentrate 2+ state 4 regions that border each other and rely on state 7 switching
[16] in melee, the AI will have a default force value of 1000 ground force for regions attacked by air units, and of 1000 air force for regions attacked by ground units

This causes one of them to count as killed (raises kill count of the guard spot by 1) and the other to retain its guard status as an Archon - even though the guard spot would still be assigned to a Templar in theory.

Overall, most morphing units will cause noticeable overproduction of their composite units and be off-centered with where they stand compared to other non-morphing units.

There is a vanilla 1.16/1.20+ bug that crashes the game related to guards - more in " **Known bugs and non-command related issues** "
Location:
0 = town center
1 = mineral line
2 (and everything else) = geyser/refinery

prep_down (byte) (byte) (military)
Add all %3(military) to the current attacking party except for %1(byte) of them, however it must send, and trains, a minimum of %2(byte) of them.

tech (tech) (byte)
Research technology %1(technology), at priority %2(byte).

upgrade (byte) (upgrade) (byte)
Research upgrade %2(upgrade) up to level %1(byte), at priority %3(byte). The maximum supported upgrade level is 31.

Upgrade/Tech/Build note: Max of 100 U/T/B requests can be made per town. Crashes above 100.

wait_upgrades
Only works in campaign scripts - waits until all UPGRADE (not techs) requests have begun researching. Supposedly waits until level 3 has started researching if you call it immediately from 0. You should wait approximately 30 seconds (wait 480) before setting the requests and calling wait_upgrades, so that the request log has enough time to update.

wait (word)
Wait until %1(word) frames have passed.

Flow control commands

call (block)
Call %1(block) as a sub-routine. Only one call can be made in the game at all times from any computer player running a script.
In the vanilla game, if player 1 makes a call, and then player 2 makes a call, it will override player 1's call and control the flow of the return statement.
Can be used for multiple players to simultaneously execute the same code.
Be sure that there is no wait commands or blocking commands inside the call to have it function correctly with multiple players

return
Return to the flow point of the call command.

enemyowns_jump (unit) (block)
If an enemy player commands at least one %1(unit) (includes buildings in construction), jump to %2(block).

enemyresources_jump (word) (word) (block)
If the closest enemy player has at least %1(word) minerals and %2(word) gas then jump to %3(block).

goto (block)
Jump to %1(block).

groundmap_jump (block)
If the closest enemy player can be reached without using transports, jump to %1(block).

region_size (byte) (block)
Jump to block %2 if the town this command is used in has a pathfinder region count (meaning the sum of pathfinder regions connected and pathable by ground) below 32 times %1.

if_owned (unit) (block)
If the player owns a %1(unit) (includes incomplete) then jump to %2(block).

killable
Sets a flag that allows the current thread to be killed by another one. The killed thread always finishes whatever wait command it's currently stuck on before stopping.

kill_thread
Kills all threads that have the killable flag (including threads owned by other players).

notowns_jump (unit) (block)
If the computer doesn't have a %1(unit), jump to %2(block).

race_jump (block) (block) (block)
According to the race of the nearest enemy player, jump to %1(block) if the enemy is Terran, %2(block) if Zerg or %3(block) if Protoss.

random_jump (byte) (block)
Has a %1(byte) chance out of 256 to jump to %2(block).

resources_jump (word) (word) (block)
If the current player has at least %1(word) minerals and %2(word) gas then jump to %3(block).

time_jump (byte) (block)
Jumps to %2(block) if %1(byte) normal game minutes have passed in the game.

try_townpoint (byte) (block)
Jump to %2(block) if the AI **doesn't own** at least %1(byte) **expansions** (Includes areatowns) - ignored otherwise.

rush (byte) (block)

Will jump to %2(block) if the closest enemy Player %1(byte) fulfills conditions listed below, based on that Player's race. Clears any currently-prepared attacks regardless of whether it jumps or not.

Rush scoring system:

```
// Protoss Air Score
result = ProtossDragoons + ProtossScouts

// Zerg Ground Score
result = ZergHydralisks + ZergSunkenColonies * 2

// Zerg Air Score
result = ZergHydralisks + ZergMutalisks + ZergSporeColonies * 2

// Terran Infantry Score
if (bunkers * 4 <= marines)
        result = marines + bunkers * 4;
else
        result = 2 * marines;
```

Rush conditions:
TERRAN
0: Commands a Barracks
1: Terran Infantry Score > 16
2: Terran Infantry Score > 24
3: Terran Infantry Score > 5
4: Terran Infantry Score > 16
5: Terran Infantry Score > 6
6: Terran Infantry Score > 12
7: Commands a Siege Tank
8: Terran Infantry Score > 5
9: Terran Infantry Score > 9
10: Terran Infantry Score > 4

11: Terran Infantry Score > 10
12: Terran Infantry Score > 16
13: Terran Infantry Score > 24

ZERG
0: Commands a Spawning Pool
1: Zerg Ground Score > 10
2: Zerg Air Score > 10
3: Zerg Ground Score > 2, or commands a Hydralisk Den
4: Zerg Ground Score > 10
5: Zerg Ground Score > 6
6: Zerg Sunken Colonies > 1
7: Commands a Queen
8: Zerg Ground Score > 2
9: Zerg Ground Score > 4
10: Zerg Ground Score > 4
11: Zerg Ground Score > 10
12: Zerg Air Score > 5
13: Zerg Air Score > 10

PROTOSS
0: Commands a Gateway
1: Protoss Zealots > 6
2: Not used
3: Protoss Zealots > 1
4: Protoss Zealots > 8
5: Protoss Zealots > 3
6: Protoss Dragoons > 1
7: Protoss Zealots > 6
8: Protoss Zealots > 1
9: Protoss Zealots > 5
10: Protoss Zealots > 2
11: Protoss Zealots > 5
12: Protoss Air Score > 2
13: Protoss Air Score > 7
---

stop
Stop script code execution.

Multiple threads commands

expand (byte) (block)
Run code at Block (that has to begin with start_town) to create a new area town at a random unoccupied resource location. This is ignored if the Computer player has more than Byte towns.

allies_watch (byte) (block)
Expands at resource area number %1 (byte) using %2 (block). Note: If %1 is < 9 then it is a player's start location (unless it’s not placed in which case ->), otherwise it is a map's base location. Does nothing if the expansion is occupied (Resarea flag = 1). The maximum value for %1 (byte) is 250.

panic (block)
If AI has not expanded yet and total unmined minerals in the mineral line are less than 7500, then it will expand using (block). If the AI has expanded before, the command triggers every time there are less than 7500 unmined minerals total in all owned bases, or there are less than 2 owned Refineries that are not depleted.

multirun (block)
Run %1(block) simultaneously, as a new thread with the same town ID. There's a limit of 100 threads working on a map simultaneously.

Miscellaneous commands

create_nuke
Create a nuke. Requires a free nuclear silo for the player.

create_unit (unit) (word) (word)
Create %1(unit) at map position (x,y) where x = %2(word) and y = %3(word).

creep (flag)
A flag that controls the spread of cannons and creep colonies: Wide if %1 = 4, or narrow if it's anything else.

debug (block) (string)
Show debug string %2(string) and continue in %1(block).

define_max (byte) (unit)
Define the maximum number of owned %2(unit) to %1(byte). Byte 255 equals zero. Preplaced guards don't count towards the limit.

fatal_error
Crashes starcraft with a fatal error and the message "Illegal AI script executed.".

give_money
Refills minerals up to 2000 and ore up to 2000.

nuke_pos (word) (word)
Launch a nuke at map position (x,y) where x = %1(word) and y = %2(word). Needs an available Ghost with access to armed nuke silos.

nuke_rate (byte)
Sets the nuke timer to %1(byte), which causes the AI player to try using nukes every %1(byte) game minutes. By default, the nuke_rate is 0. The nuke timer resets only when a nuke's launching procedure has been started: otherwise the AI will continuously send ghosts one by one. (Or wait until a nuclear missile has finished production) nuke_rate 0 allows the computer player to launch multiple nuclear missiles at once, up to the number of full nuclear silos.

fake_nuke
Resets the AI nuke timer.

send_suicide (byte)
%1(byte) determines which type:

0 = Strategic suicide - Sends most military units on suicide missions. Will usually not work if constantly repeated, also will not work for players with no town management;

1 = Random suicide. Sends all units on random suicide missions. Will work for town management-less players, also suicides workers and heroes, will need to be repeated on a loop to work properly if the player has no town management

set_randomseed (dword)
Set random seed to %1(dword). Supposedly limits the number of random scenarios that can occur. Player inputs cause random seeds to be unpredictable, thus it'd only ever be useable in opening cutscenes.

implode
Changes the state of AI regions: For any region which has state 0, 1,2 or 3 that's connected to at most one pathfinding region by ground, the region's state is changed to 4.

For more information on regions, refer to "Region states" at the very bottom of the doc.

StarEdit commands

All staredit commands take a location to work.

clear_combatdata
Clears all AI attack data for all units in the location. (Removes units from attacks and clears current defense orders, does not remove guards)

disruption_web
Orders a Protoss Corsair to cast Disruption Web in the location.

enter_bunker
Orders units in the location to enter the closest bunker.

enter_transport
Orders units in the location to enter the closest transport.

exit_transport
Orders transports in the location to unload their units.

junkyard_dog
Orders units in a location to "Junk Yard Dog" (similar to a critter, attacks nearby enemies).

make_patrol
Orders units in a location to patrol to the Generic Command Target.

set_gencmd
Sets the Generic Command Target to the center of a location.

move_dt
Orders the computer to move all Hero Dark Templars and Dark Templars to the specified location. (Crashes if there are any Dark Templars in training)

nuke_location
Orders a computer player to nuke the selected location. Needs an available Ghost with access to armed nuke silos.

player_ally
Makes all players in a specified location an ally of the trigger owner.

player_enemy
Makes all players in a specified location an enemy of the trigger owner.

recall_location
Orders an Arbiter to use recall at this location.

sharedvision_off (player)
Orders the specified player to unvision (player), where the player number is 1 less than it'd be in staredit.

Can be used on self to allow building stacking.

sharedvision_on (player)
Orders the specified player to give vision to (player), where the player number is 1 less than it'd be in staredit.

value_area
Causes the computer to consider the specified region for defense. If the AI doesn't control any buildings in the region, it will only be defended once. Doesn't do anything if the region already has enemy units inside of it.
(Sets the region state to 5)
(The region is set in the center of the location, not over all covered regions)

switch_rescue
Switch the AI to a rescuable player.

Unused commands

if_towns
Does not exist in memory.

harass_location
Supposedly caused the AI to prefer attacking the selected region. Currently (1.16.1+) unused.

scout_with (military)
Does not exist in memory.

if_dif (byte) (byte) (block)
Jumps to block %3 if value %2 is different from an internal value related to an unused difficulty setting, which is now always 1 for active (ones that called start_town) players, 0 for inactive, using modifier %1(0 = greater than, 1 = less than).

easy_attack (byte) (unit)
Used only in conjunction with an unused difficulty setting.

AI limits

requests - 100 per AI town (each separate build, tech, player_need and upgrade call is a request)
results of breaching: corrupts other towns, almost always **results in crashes**

guard units - 1000 globally (both alive and requested)
result of breaching: place_guard doesn't work, units spawned via triggers or create_unit get frozen

military units - 1000 globally (only defenseuse/attack units matter for this limit)
result of breaching: new units that get trained get frozen

workers - 1000 globally, 30 hardcoded per AI
result of breaching: unknown for the global limit, workers work correctly for the single player limit

town buildings - 1000 globally (also includes workers, larvae, eggs and overlords)
result of breaching: unknown

threads - 100 globally (this includes town scripts, areatown scripts, and all multiran or otherwise existing threads)
result of breaching: unknown, might cause crashes

towns - 100 globally (start_town scripts and threads and start_areatown scripts)
result of breaching: unknown, might cause crashes

defenseuse_xy - 20 per defenseuse type, separate for each player
result of breaching: units beyond the 20th will not be used for defense of a given type

attack_add - 64 without overwriting attack target, also affects all other ways of adding units to an attack
result of breaching: if overwritten, the attack functions completely correctly

resareas - 250
result of breaching: cannot be breached

Region states

state 0 = Nothing: might “need” a force, but is not defended unless within (some) radius of a state 5 region. (it then can turn into state 6 or 3)

If defended, defense priority is 10

When a region is denoted as “being created” below, it usually means that it’s state is turning from state 0 into something else - since regions obviously can’t get created per-se midgame.

state 1 = Continue attack here. Not defended.

state 2 = Attacking: Always appears before state 1 and is used for the first attack target. Not defended.

state 3 = Enemy spotted: Can’t repair/rebuild and defend if made from a state 5 region. If there are neighbouring state 4 regions when it’s created, it can set one of them to state 7.

state 4 = This state is used by numerous commands:

It’s used for defending future towns, and a few temporary state 4 regions are created around expansions that are currently being constructed;

Multiple state 4 regions are created by capt_expand, surrounding town regions and filling in the blanks between state 5’s: they’re also used to distribute units made from capt_expand’s secondary function.

implode creates state 4 regions in state 0, 1, 2, 3 regions that are connected to at most 1 ground region.

help_iftrouble creates state 4 regions for allied AIs near and inside of the defended player’s towns.

It is important to know that state 4 regions are usually temporary: after they have been defended for the first time, they tend to either turn into state 0’s or into state 3’s: they also turn into state 6’s or 3’s immediately as they are attacked.

Defense priority 40

state 5 = Town, created around most non-supply and non-defensive structures and campaign beacons, also created with value_area script: turns into state 6 when attacked and turns back into state 5 when idle again. Might turn into state 3 and then into 6 and back into 5. Defended. Sometimes created for friendly AIs by help_iftrouble in vital town parts of the defended AI.

Defense priority 50

NOTE ABOUT STATES 0, 3, 4, 5: Their needed strength (force) value decays over time (even if the AI player has a default_min value) if (unknown conditions) occur. Usually just happens for abandoned and outlying regions.

state 6 = Actively defending. Each town has its town center highlighted by it being a permanent state 6 region. If the AI is sending units to a state 6 region it has no units in, it will change to state 3.

NOTE ABOUT STATES 4, 6: These states are the only ones to recognize default_min values: it tells them how much force do they need to defend at all times, though outer state 4's are still affected by strength decay. Ground regions only call for ground defenses and air regions only call for air defenses from default_min.

state 7 = Chase away - targets a defended region and uses its needed force value to prepare defenses in this region. If the AI is sending units to a state 7 region it has no units in, it will change to state 3, except if it was created by another state 3 region.

Defense priority for states 6, 7 is 100 or 110 if the region has a worker or a building inside

state 8 = Preparing attack. Not defended. The target region is used for harass_factor calculations.

state 9 = Preparing attack after state 8 region was deemed unreachable - occurs in the (nearest?) region with enemy units in it. Attacks going to state 9 regions will generally act the same as ones going to state 8, but their respective attack won't wait until attack preparation finishes, like it usually would. They also don't stop grouping after attack_clear.

Should never occur with vanilla attack region selection code.

Known bugs and non-command related issues

- The attacks will seldom stop it's attacking units midway of an attack preparation when the first target dies, especially in melee scripts
- If an upgrade/tech ID corresponds to the worker ID (i. e. ID 7 for SCV, irriadiate and t_inf_weapon) of the given race in an AI town, then the upgrade/tech level can never be higher than the town's worker limit (which is 3 with just a geyser: can be 0 if the town has no resources assigned)
- If an SCV is destroyed during the construction of a building, the AI tries to repair the building instead of renewing construction, and the given town's request log hangs up until the building is destroyed
- The AI will sometimes to attempt casting targeted spells while not having enough energy for them
- The AI has no checks for overkilling enemies with spells and will use multiples of them on a single target (area) if applicable and possible
- Train requests (Using train, do_morph, wait_force and possibly wait_train) used when the AI has a guard of the same type that you want to train, while the guard needs to be replenished, while the guard area is not in combat, causes the train request to be ignored and makes the AI train units as if you had defaultbuild enabled (fixed by AISE)
- Train->Morph requests (Archons, Guardians, Devourers...) can get deleted if the AI can't immediately act upon them: the AI should have a perfectly clear request log before training those units if you want them to accurately show up without any abnormalities
- Non-guard (military) High Templars the AI controls won't merge into Archons unless they are in a region with state 4, 5, 8 or 9
- Preplaced Creep Colonies can't be morphed into anything by the AI because their primary order is Nothing instead of Computer Script

- Attacks started at 0 frames have no expiration deadline
- Zerg Eggs owned by the AI killed in the last 10 frames of hatching (visually, this makes the unit completely disappear) can cause a crash if their town is not eliminated quickly
- AI won't build buildings near state 0 regions that were recently attacked (and "recently" means "forever" unless the region state changes)
- **The guard crash - steps to reproduce (as discovered by Neiv) (fixed by AISE)**

1) A guard needs to have been killed at least twice while still being a guard
2) Once the AI starts the request to replace the guard, it must be busy enough to take a few seconds before acting on it
3) One unit build time later, the AI needs to have another idle production building which can start creating the second guard.
4) The first guard spawns after the second guard has started production
5) The first guard dies before the second guard finishes
6) The second guard spawns, and stays alive long enough that its guard ai gets reused for some unrelated request
7) The game crashes

Credits:
Created by Nekron.

Neiv, for creating accurate descriptions of attack commands, and helping with scripting and documenting almost all of the previously undescribed commands.

Heinermann, for writing the original post which this was initially based on, and for writing descriptions of multiple AI functions.

Many others for in-game function testing.