NTSTATUS DriverEntry(PDRIVER\_OBJECT DriverObject, PUNICODE\_STRING RegistryPath)

1. Объявить переменную status типа NTSTATUS;
2. Объявить переменную i типа int и инициализировать ее значением 0;
3. Установить функцию выгрузки драйвера DriverUnload в DriverObject->DriverUnload;
4. Начать цикл с условием i = 0, i < IRP\_MJ\_MAXIMUM\_FUNCTION, i++, при выполнении условия перейти к шагу 5, иначе к шагу 6;
5. Установить функцию обработки DispatchPass в DriverObject->MajorFunction[i], перейти к шагу 4;
6. Установить функцию обработки DispatchRead для IRP\_MJ\_READ в DriverObject->MajorFunction[IRP\_MJ\_READ];
7. Вызвать функцию MyAttachDevice с передачей DriverObject в качестве аргумента и сохранить результат в status;
8. Если статус выполнения не является успешным, вывести отладочное сообщение "Attaching is FAILING." и перейти к шагу 10;
9. Иначе, вывести отладочное сообщение "Attaching is SUCCESS." И перейти к шагу 10;
10. Вернуть статус выполнения;
11. Конец функции.

VOID DriverUnload(PDRIVER\_OBJECT DriverObject)

1. Инициализировать переменную interval типа LARGE\_INTEGER со значением 0;
2. Получить указатель на DeviceObject из DriverObject;
3. Задать значение interval.QuadPart равным -10 \* 1000 \* 1000;
4. Отсоединить устройство с помощью функции IoDetachDevice и передать LowerKbdDevice из расширения устройства DeviceObject;
5. Войти в цикл с условием пока pendingkey не равно нулю, перейти к шагу 6, если условие не выполняется, перейти к шагу 7;
6. Задержать выполнение потока с помощью функции KeDelayExecutionThread и передать режим KernelMode, флаг FALSE и адрес переменной interval, перейти к шагу 5;
7. Удалить устройство с помощью функции IoDeleteDevice и передать myKbdDevice;
8. Вывести отладочное сообщение "Driver is UNLOAD.";
9. Конец функции;

NTSTATUS MyAttachDevice(PDRIVER\_OBJECT DriverObject)

1. Объявить переменную status типа NTSTATUS;
2. Создать переменную TargetDevice типа UNICODE\_STRING и инициализировать ее значением "\\Device\\KeyboardClass0";
3. Создать устройство с помощью функции IoCreateDevice, передав DriverObject, размер расширения устройства sizeof(DEVICE\_EXTENSION), NULL, тип файла FILE\_DEVICE\_KEYBOARD, флаг 0, флаг FALSE и адрес переменной myKbdDevice;
4. Если создание устройства не удалось (статус выполнения не является успешным), вернуть статус выполнения и перейти к шагу 10;
5. Установить флаги DO\_BUFFERED\_IO и снять флаг DO\_DEVICE\_INITIALIZING у myKbdDevice;
6. Очистить память расширения устройства myKbdDevice с помощью функции RtlZeroMemory, передав размер sizeof(DEVICE\_EXTENSION);
7. Присоединить устройство с помощью функции IoAttachDevice, передав myKbdDevice, &TargetDevice и адрес переменной ((PDEVICE\_EXTENSION)myKbdDevice->DeviceExtension)->LowerKbdDevice;
8. Если присоединение устройства не удалось (статус выполнения не является успешным), удалить устройство с помощью функции IoDeleteDevice и вернуть статус выполнения, перейти к шагу 10;
9. Вернуть STATUS\_SUCCESS;
10. Конец функции.

NTSTATUS ReadComplete(PDEVICE\_OBJECT DeviceObject, PIRP Irp, PVOID Context)

1. Создать массив строк keyflag размером 4 и заполнить его значениями "KeyDown", "KeyUp", "E0", "E1";
2. Получить указатель на массив структур Keys типа PKEYBOARD\_INPUT\_DATA из SystemBuffer в Irp;
3. Вычислить количество структур в массиве Keys и сохранить результат в переменную structnum;
4. Инициализировать переменную i равной 0;
5. Если статус выполнения Irp равен STATUS\_SUCCESS, перейти к шагу 6, инече к шагу 15;
6. Начать цикл с условиями i=0, i<structnum, i++, если условия выполняются перейти к шагу 7, иначе к шагу 15;
7. Проверить, является ли нажатой клавиша Caps Lock с помощью условия Keys[i].MakeCode == 0x3A && Keys[i].Flags == 0;
8. Если клавиша Caps Lock нажата, изменить значение переменной CapsLockPressed на противоположное;
9. Если переменная CapsLockPressed равна истине, перейти к шагу 10, иначе к шагу 14;
10. Начать цикл с условиями j=0, j<46, j++, если условия выполняются перейти к шагу 11, иначе к шагу 14;
11. Проверить, совпадает ли Keys[i].MakeCode со значением scanCodeMs[j];
12. Если значения совпадают, вывести отладочное сообщение "The button is block, scan code = %x.", иначе перейти к шагу 10;
13. Восстановить память структуры Keys[i] с помощью функции RtlZeroMemory, передав размер sizeof(KEYBOARD\_INPUT\_DATA) и перейти к шагу 10;
14. Вывести отладочное сообщение "The SCAN CODE is %x (%s)\n", передав Keys[i].MakeCode и keyflag[Keys->Flags];
15. Если у Irp установлен флаг PendingReturned, вызвать функцию IoMarkIrpPending и передать Irp;
16. Уменьшить значение pendingkey на 1;
17. Вернуть статус выполнения Irp с помощью Irp->IoStatus.Status;
18. Конец функции.